Christian Pirchheim

Exploiting Constraints in Visual Localization and Mapping for Mobile Augmented Reality

DOCTORAL THESIS

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Thesis Supervisor

Dr. Dieter Schmalstieg

Graz University of Technology, Austria

Thesis Referee

Dr. Walterio Mayol-Cuevas

University of Bristol, United Kingdom

Graz, Austria, September 2015

Abstract

Visual localization and mapping are core problems that must be successfully tackled for the proliferation of mobile Augmented Reality (AR) applications in arbitrary indoor and outdoor environments on hand-held and wearable devices.

Localization and mapping are tightly interconnected problems, as localization refers to continuously estimating the motion of a moving camera with respect to a virtual model, while mapping refers to creating exactly these models. An AR system may benefit from the knowledge of an existing model such as a 3D city reconstruction, as this model can provide an advanced understanding of the scene, including object semantics, and annotations, which are registered in the model coordinate system that ideally has metric scale and is globally aligned. In general, however, an AR system cannot assume to operate in an a-priori modeled environment. In this latter case, without any world knowledge, an AR system must first build up such a model from scratch, online and in real-time, using Structure from Motion (SfM) and Simultaneous Localization and Mapping (SLAM) methods.

Most SLAM algorithms do not make any prior assumptions about the user-performed camera motion, or the geometry and the semantics of the observed scene. In many cases, however, the scene contains higher-level geometric primitives such as lines and planes. Additionally, it is possible to take advantage of a constrained camera motion model. We propose to approach certain limitations and problems of monocular visual SLAM by exploiting specific combinations of these constraints. These constraints allow for developing localization and mapping algorithms, which we integrated into efficient SLAM systems suited for hand-held devices such as phones and tablets. In particular, we have developed the following techniques:

- Two techniques for mapping and tracking of arbitrary camera motion and scene structure, depending on the environment knowledge. First, in unknown environments, we modeled hybrid maps that contained features with finite or infinite depth. Second, in partially modeled urban outdoor environments, we simulated a wide-range depth camera by rendering synthesized depth images.
- A geo-localization technique suited for the initialization of metric-scale 3D SLAM maps in urban outdoor environments. Given a single image and a coarse initial 6D pose prior as input, our geo-localization method employed computer vision techniques to estimate a refined 6D pose with respect the global coordinate system provided by an untextured 2.5D city model.
- A technique for mapping planar scenes by exploiting the homography motion model. Homographies provide strong constraints on both camera motion and scene structure, which we employed to implement a very efficient mapping algorithm especially suited for lo-fi mobile devices.

In our practical experiments, we found that, by handling parallax-free camera movements, we could considerably extend the total camera tracking time and, thus, the usability of AR applications. Our geo-localization algorithm was found very versatilely applicable and efficiently performing, and allowed for rendering geo-referenced annotations such as navigation hints in urban AR scenarios. Running at least a magnitude faster than bundle adjustment, our homography-based SLAM algorithm was suited for mapping high-level maps of planar scenes, which we found on numerous human-made structures. In summary, we developed multifunctional and practical localization and mapping methods, which could be successfully applied in mobile AR applications for registration in outdoor as well as indoor environments.

Keywords: augmented reality, mobile devices, SLAM, localization, geo-localization. tracking, mapping, initialization, metric scale, planes, lines, scene understanding.

Kurzfassung

Bildbasierte Lokalisierung und Kartografierung sind Kernprobleme, die für die weitere Verbreitung von mobilen Augmented Reality (AR) Anwendungen auf mobilen Geräten in beliebige Innen- und Außenbereichen erfolgreich in Angriff genommen werden müssen.

Lokalisierung und Kartografierung sind eng miteinander verschränkte Probleme. Lokalisierung bezieht sich auf das kontinuierliche Schätzen der Bewegung einer mobilen Kamera in Bezug auf ein virtuelles Modell, während Kartografierung sich auf die Erzeugung genau dieser Modelle bezieht. Ein AR-System kann aus der Kenntnis eines bestehenden Modells, wie zum Beispiel einer 3D Stadt-Rekonstruktion, profitieren, da dieses Modell ein erweitertes Verständnis der Szene liefern kann, darunter Objektsemantik und virtuelle Annotationen, die in einem idealerweise metrischen und global ausgerichteten Koordinatensystem registriert sind. Im Allgemeinen jedoch kann ein AR-System nicht annehmen, sich ein einer a-priori modellierten Umgebung zu befinden. In diesem Fall, ohne Wissen über die Welt, muss ein AR-System zunächst von Grund auf ein solches Modell erstellen, und zwar zur Laufzeit und in Echtzeit. Das ist ein Vorgang der in der englischen Fachliteratur mit den Begriffen Structure from Motion (SfM) und Simultaneous Localization and Mapping (SLAM) (dt. "Simultane Lokalisierung und Kartografierung") bezeichnet wird.

Die meisten SLAM-Algorithmen machen keine Annahmen über die Bewegung der vom Benutzer geführten Kamera oder über die Geometrie und Semantik der beobachteten Szene. In vielen Fällen jedoch enthält die Szene einschränkende geometrische Elemente wie Linien und Ebenen. Zusätzlich ist es möglich, aus einem eingeschränkten Kamerabewegungsmodell Vorteile zu ziehen. In dieser Arbeit schlagen wir vor, spezielle Kombinationen dieser geometrischen und semantischen Einschränkungen für das Lösen von bestimmten Problemen von visuellem SLAM zu nutzen. Die Verwendung dieser Einschränkungen erlaubten uns die Entwicklung von Lokalisierungs- und Kartografierungs-Algorithmen, die in effiziente SLAM-Systemem auf mobilen Geräten wie Mobiltelefonen oder Tablets integriert werden können. Im Speziellen haben wir die folgenden Techniken entwickelt:

- Zwei Techniken zum Kartografierung und Lokalisieren von beliebigen Kamerabewegung und beliebigen Szenen, in Abhängigkeit vom Wissen über die Umgebung. Erstens, in unbekannten Umgebungen, modellierten wir Hybridkarten, die Features mit endlicher oder unendlicher Tiefe enthalten. Zweitens, in teilweise modellierten städtischen Außenbereichen, simulierten wir eine Tiefenkamera durch das Rendern synthetisierter Tiefenbilder.
- Eine Geo-Lokalisierungstechnik für das Initialisieren von metrischen 3D-SLAM Karten in städtischen Außenbereichen. Gegeben ein einzelnes Bild und eine grobe initiale 6D Kamera-Pose, verwenden wir Computer Vision Techniken um eine genaue 6D Kamera-Pose bezüglich eines globalen Koordinatensystems, das durch ein untexturiertes 2.5D Stadtmodell gegeben ist, zu berechnen.
- Eine Technik zum Abbilden planarer Szenen durch Ausnutzung des Homographie-Bewegungsmodells. Homographien schränken sowohl die Kamerabewegung als auch die Szenenstruktur stark ein. Wir haben Homographien verwendet, um einen sehr effizienten Kartografierungs-Algorithmus zu implementieren, der besonders für leistungsschwache mobile Geräte ausgelegt war.

In unseren praktischen Experimenten haben wir festgestellt, dass durch die Berücksichtigung von parallaxenfreien Kamerabewegungen die Anzahl der lokalisierten Kamerabilder und damit die Nutzbarkeit von AR-Anwendungen deutlich gesteigert werden konnte. Weiters haben wir festgestellt, dass unser Geolokalisierungs-Algorithmus sehr vielseitig einsetzbar und effizient ist und die Darstellung von georeferenzierten Annotationen, wie zum Beispiel Navigationshinweisen, in städtischen AR-Szenarien erlaubt. Unser Homographie-basierter SLAM-Algorithmus stellte sich als mehr als eine Zehnerpotenz schneller als der übliche Bündelblockausgleichung heraus, und erlaubte planare Szenen zu kartografieren, die in zahlreichen alltäglichen Umgebungen zu finden sind. Zusammenfassend haben wir vielseitige und praktische Lokalisierungs- und Kartografierungs-Methoden entwickelt, die wir erfolgreich für die Registrierung von mobile AR Anwendungen in Innen- wie Außenbereichen eingesetzt haben.

Contents

1 Introduction

1.1 Mobile augmented reality

Augmented Reality (AR) aims to provide new forms of human-computer interfaces in which “the world becomes the user interface” [63]. More formally, AR refers to the idea of mixing the physical-real with digital-virtual elements in order to enhance the perception of a user [102]. While we are concentrating on sight, this mixture applies to all human senses including hearing, taste, smell and touch.

The ultimate goal of our research is to make mobile AR applications practically available everywhere (*e.g.*, in- and outdoors), anytime (*e.g.*, day and night, at any season of the year), and for everybody (*e.g.*, the “average” as well as the “expert” user). The realization of this broad and ambitious vision requires the availability of a set of hardware and software technologies including wearable computers, mobile networks, digital content providers, locations based services, and – finally – convenient AR applications.

The envisioned ubiquitous AR experience requires wearable or hand-held computing devices. In 2007, an important milestone was reached with the introduction of the Apple iPhone. Since then, “smart” phones have made steady progress in providing more and increasing-quality hardware and software features. In the course of these developments, ever more types of mobile devices become available on the consumer market: tablets (*e.g.*, Apple iPad), glasses (*e.g.*, Google Glass), and watches (*e.g.*, Apple Watch), just to name a few. While not all of these devices are equally suited or reap for AR applications, the current generation of camera phones and tablets (see Figure 1.1) provide all hardware components that are required for interactive see-through AR with real-time 3D graphics: megapixel camera sensors, satellite navigation receivers, motion and inertial sensors, multi-core central processing units (CPU), graphics processing units (GPU), and multi-

Figure 1.1: Selection of hand-held devices employed in this thesis: (a) Nokia N900 (2009), (b) Samsung Galaxy S2 (2011), (c) Apple iPad Air (2013), (d) Apple iPhone 6 (2014)

touch displays. However, compared to desktop and laptop computers, their computational capabilities naturally lag behind due to the form factor: CPUs are less powerful, memory capacity and bandwidth are restricted, cameras have low-quality sensors and narrow-field-of-view lenses, satellite navigation receivers as well as motion and inertial sensors are considered inferior compared to the state of the art, and general-purpose computing on mobile GPUs is still up in the air. Nevertheless, hand-held and wearable devices are considered to be the most widely available and promising hardware platform for the proliferation of mobile AR applications today.

Another important enabling technology for retrieving data from the internet and exchanging data between mobile devices are local and wide area communication networks. Local connectivity between mobile devices and other networked computers is enabled by technologies such as WiFi, Bluetooth and Near Field Communication (NFC). Wide-area mobile telecommunication networks are steadily improved in terms of coverage as well as data transmission throughput and latency and enable the communication with the internet and access to cloud services such as Geographic Information Systems (GIS).

Mobile AR requires digital content provided by location-based services. For example, Geographic Information Systems such as Google Maps and Microsoft Bing allow for retrieving information about the local environment. Similarly, community-driven services, such as OpenStreetMap, steadily make more and improved data available. Consequently, mobile AR applications require Application Programming Interfaces (APIs) and cloud storage infrastructure that allow for authoring and storing user-generated content.

To give a better idea of what we mean by mobile AR, consider the following application scenario:

> A user is sitting in a hotel lobby of a foreign city, browsing the news on his smartphone, and discovers an exhibition she wants to visit. Leaving the hotel towards the museum, she makes use of 2D map navigation to orient herself. At an ambiguous junction, however, she chooses to directly superimpose the correct walking directions over streets and buildings. Arriving at the museum, ticket and digital exhibition catalog can be purchased online with a few clicks. Moments later, she stands in front of an interesting piece and augments the object with information about its making and history in a perspectively correct manner via her device. After the visit, she can similarly visualize and explore a virtual version of the piece in an arbitrary environment, *e.g.*, at the airport gate waiting for her plane, or back home in her living room. In the public space at the airport gate, the visualization may be unregistered with the environment, providing a purely virtual view. In the private space of her living room, the visualization may be spatially registered, providing an augmented view.

In this example, the user employs the hand-held device for various purposes, including navigation, information browsing, and product purchase. The corresponding applications may employ different user interface and interaction metaphors, which fit best with the users' needs. For example, the navigation application may provide both a conventional 2D map as well as an AR interface. Any application that employs AR has to master a couple of tough requirements in order to provide a convincing AR experience, including geometric and photometric registration, visually coherent rendering, and intuitive and convenient user interfaces.

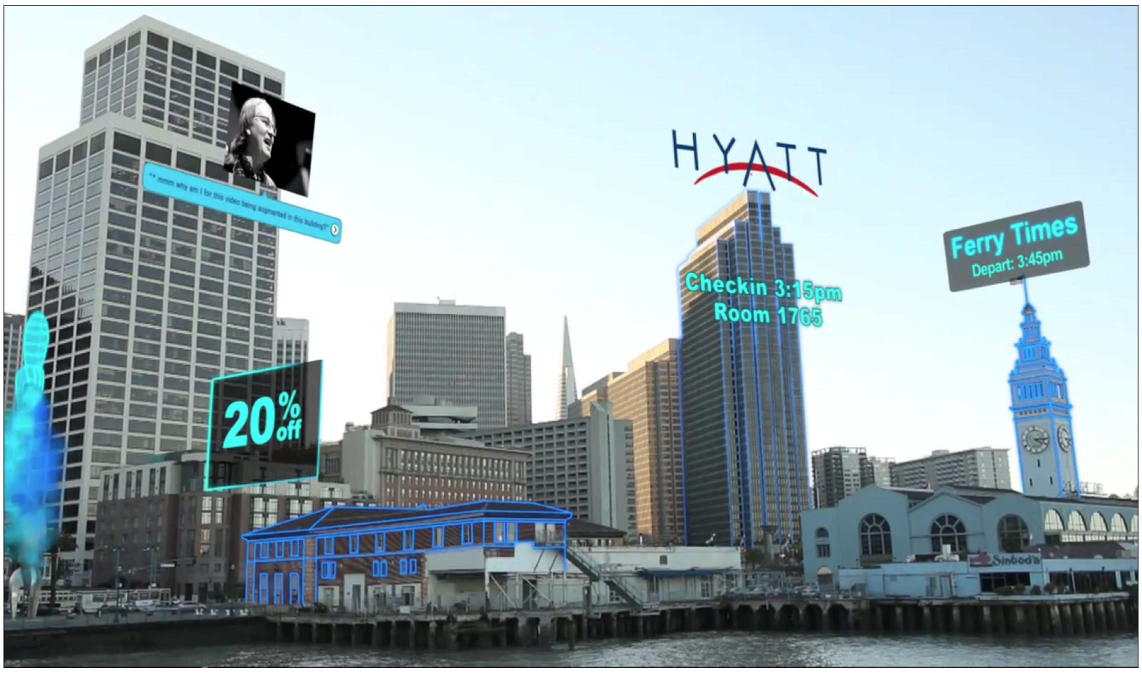


Figure 1.2: Illustration of an urban mobile AR user interface. © Layar

1.2 Registration problem

In this thesis, we concentrate on the geometric registration problem to enable mobile AR applications in indoor and outdoor, especially urban, environments. In particular, we aim to perform registration on hand-held computers using its built-in sensors and the camera.

At its core, a geometric AR registration system must solve the problem of aligning real-world physical objects with computer-generated virtual objects, for example, to augment real streets and buildings with virtual navigation hints. The requirements for convincing AR experiences are tough: registration must work at interactive frame-rates and, thus, process the incoming sensor input data and interaction events efficiently and robustly in real-time, producing a highly accurate registration output. These requirements regularly involve a tradeoff between different AR system components (*e.g.*, registration, 3D graphics rendering, interaction event handling), considering the available resources, in particular on restricted computing platforms such as hand-held devices. The requirements contrast with registration systems, which may neglect real-time constraints or have all input data available at once for batch processing (*e.g.*, offline Structure from Motion), or do not have such excessive accuracy requirements (*e.g.*, Virtual Reality).

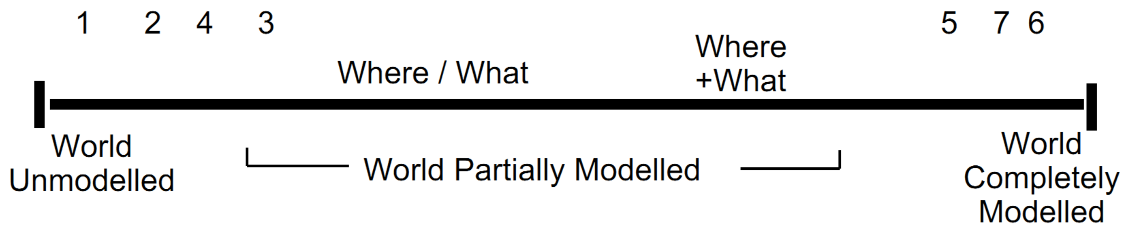


Figure 1.3: Extent of World Knowledge (EWK) continuum. © [102]

1.3 Extent of world knowledge

In order to analyze the AR registration problem in more detail, let us have a look at Figure 1.3: The Extent of World Knowledge (EWK) continuum by Milgram and Koshino [102] refers to the question about how much an AR system knows about the world being perceived (displayed or tracked) at a certain point in time. In particular, the continuum raises the following questions:

"Where is the real object?"
Refers, *e.g.*, to localization with respect to a geometric model, and to mapping of a geometric model of a real-world object.

"What is the real object?"
Refers, *e.g.*, to high-level geometric and semantic scene understanding.

The answers to the "Where?" and "What?" questions are independent of each other: On the one hand, the system may know the location of a real object (the "Where"), but not its semantics (the "What"). On the other hand, the system may know that within the environment a certain known object (the "What") should exist, but does not know its location (the "Where"). In the above example, in order to provide navigation hints in the urban environment, the AR system must localize itself with respect to the city map: based on the coarse GPS position, the system assumes that a certain building must be in the perceived environment ("What"), but has not yet localized its exact position and orientation with respect to the user ("Where"). Later on at the museum, the received virtual model of the exhibition piece provides both means for localization and semantic annotations ("Where" and "What"). Finally, the same virtual exhibition piece may be rendered on top of a table in the users' living room. In this case, the AR system may observe and reconstruct the living room including the table at runtime, and employs the corresponding geometric model for continuous localization ("Where"). However, while the system may not know that the geometric model contains a table object, it might know that the table provides a planar surface for placing the virtual exhibition piece ("What").

In general, models may provide information about geometry (for answering the "Where") and semantics (for answering the "What"). The geometric representation of a model can be very different ranging from sparse point clouds to dense surface reconstructions. Furthermore, many geometric models are merely untextured approximations of physical objects, lacking details or entire perspectives, *e.g.*, when only the street side of a building is modeled. Semantic modeling can well be considered an even harder problem compared to geometric modeling. For example, consider the extraction of planes or cubes from geometric models. Compared to points, these geometric primitives may already provide valuable semantic hints to AR applications. On the flip side, models may contain 2D semantic, but no 3D geometric information, *e.g.*, consider images which have been classified to contain "cars" or "sky". An important property of any virtual model is its coordinate system constituting the reference for localization, which can be a local (relative) or global (absolute) coordinate system, such as Universal Transverse Mercator (UTM) or World Geodetic System (WGS). In the latter case, the models have metric scale, compared to arbitrary scales in the case of local coordinate systems.

Overall, registration systems gravely depend on the quality of the virtual models and have to work around or deal with related problems. Where do these virtual models come from? The digitalization of the world is steadily ongoing, with models of outdoor (*e.g.*, Google Earth or OpenStreetMap models) and indoor environments (*e.g.*, the aforementioned digital museum exhibition catalog) being created or extended every day. However, there are further problems: For example, as the world is constantly changing (*e.g.*, due to construction activities in cities), virtual models, understandably, lag behind as they cannot be updated at the same pace. A specific problem of models captured from images are seasonal (winter vs. summer) and daytime (day vs. night) appearance changes, which lead to very different extracted geometric information.

Depending on the *a priori* world knowledge, an AR system may employ different approaches for registration. We can distinguish the following cases of world knowledge:

Completely or partially modeled environments

I.e., complete or partial world knowledge.

Unmodeled environments

I.e., no world knowledge.

While the distinction between complete and partial world knowledge is fuzzy, and the transition is fluent, we think that complete world knowledge is the absolute exception and hardly ever the case.

In the case of complete or partial world knowledge, an AR system perceives environments which are modeled or prepared. Prepared environments may be equipped with outside-in “open-loop” tracking systems (*e.g.*, optical, electromagnetic, ultrasonic) or – in outdoor environments – Global Navigation Satellite Systems (GNSS) such as the Global Positioning System (GPS). Global satellite navigation can be considered as partially modeled environment, because satellite signals cannot be perceived everywhere equally or not in sufficient quality for AR applications. Furthermore, inside-out “closed-loop” registration systems may operate in environments which are prepared with fiducial markers, or employ digital models which contain perceived real-world objects. The latter generally refers to visual model-based tracking and detection algorithms. Similarly, partial world knowledge implies that only some of the perceived world objects may be digitally modeled, or perhaps not modeled in sufficient quality for AR applications. Overall, in case of complete or partial world knowledge, an AR system can answer the “What” and “Where” questions depending on model completeness and quality. To extend that knowledge, an AR system may perform modeling at runtime, and thus attempt to extend the partial model.

It is important to note that a registration system may benefit from a-priori known virtual models of real-world objects and environments, as models may provide valuable information such as a global coordinate system, or metric scale, or object semantics. In general, however, an AR registration system cannot assume an a-priori modeled environment.

In the case of no world knowledge, an AR system perceives environments which are unmodeled, or at least no model is known a priori to the AR system. In this case, an AR system must first create a model in order to answer “What” and “Where” questions. Milgram and Koshino [102] suggest to “make quantitative measurements of the observed real world. With each measurement that is made, we are therefore effectively increasing our knowledge of that world, and thereby migrating away from the left hand side of the EWK axis, as we gradually build up a partial model of that world.” This statement anticipates localization and mapping algorithms.

1.4 Visual localization and mapping

When observing a real-world scene with a moving digital video camera, we can extract measurements from the rich information encoded in the individual images of the video stream, and use these visual measurements to reconstruct a virtual model that consists of the geometric *scene structure* as well as the *motion path* of the camera in 3D space. In the Computer Vision literature, this 3D reconstruction process is regularly called Structure from Motion (SfM), and includes the *localization* of individual camera images with respect to the scene geometry, as well as the *mapping* of the scene geometry.

Both, localization and mapping, rely on matching corresponding measurements (observations) of the same 3D landmarks within two or more images. Employing the underlying projective geometry, we can estimate the location and orientation (6D/6DOF pose) from where the camera images were taken and the 3D position of the measured image features (*e.g.*, point or line features). The reconstruction process is typically bootstrapped by establishing feature correspondences between a stereo pair of images that provide two views with sufficient parallax (*i.e.*, the ratio of the baseline between the camera centers vs. the observed scene depth) for robust estimation of motion and structure.

Visual localization and mapping approaches can roughly be classified in the following way: Approaches which are primarily interested in receiving a consistent model (*e.g.*, sparse or dense 3D reconstructions), rather than in concurrent localization and mapping in *real-time*, are referred to as Structure from Motion (SfM) [58]. Furthermore, Visual Odometry (VO) [114, 135, 50] approaches are laid out for real-time localization with respect to a temporary model, but neglect mapping a consistent model. And finally, Simultaneous Localization and Mapping (SLAM) [149, 32, 72, 109, 110, 41] approaches are interested in both, motion and consistent model estimation, concurrently and in real-time.

With MonoSLAM, Davison [32] introduced monocular visual SLAM, that is, SLAM using a single “agile” RGB camera as the only sensor input. MonoSLAM, as most of the SLAM work at the time, employed a probabilistic filtering framework, the Extended Kalman Filter (EKF). In contrast, Nister *et al.* [114] showed in their Visual Odometry work that is was possible to employ SfM methods for real-time localization and mapping. However, in contrast to SLAM, Nister *et al.* performed localization with respect to a temporary 3D model that was mapped from the most recent N camera frames only. Klein *et al.* [72] considerably extended these concepts in their Parallel Tracking and Mapping (PTAM) work that was based on the idea of separating the real-time localization and keyframe-based mapping tasks on multi-core CPUs, creating the first optimization-based – using bundle adjustment [163] – SLAM algorithm that eventually was found more robust and efficient compared to filter-based SLAM [152]. In the AR community, monocular visual SLAM systems such as MonoSLAM and, particularly, PTAM became popular for the registration of hand-held cameras in unmodeled environments.

1.5 Problem statement

In this thesis, we aim to shift the "What" and "Where" markers alongside the EWK continuum and increase the world knowledge for partially modeled and unmodeled environments by employing the SLAM approach.

Our own contributions on real-time localization and mapping presented in this thesis are based on Klein and Murray's work on PTAM [72, 73, 74]. We aim for even more "agile" monocular visual SLAM algorithms, which entirely and autonomously run on hand-held computers such as phones and tablets, using the built-in RGB camera of these devices as primary input sensor. In particular, we tackled several important limitations and problems of keyframe-based SLAM dealing with restrictions on estimating arbitrary camera motion and scene structure, providing geometric scene understanding (*i.e.*, high-level feature mapping), initializing metric-scale SLAM maps and geo-localization in urban outdoor environments.

Overall, SLAM is considered a "chicken and egg" problem. PTAM's camera localization relies on high-quality 3D features contained in the model, required for robust 6D pose estimation. In turn, mapping relies on high-quality keyframes poses, required for robust 3D feature location (depth) estimation. This mutual dependence is best reflected in bundle adjustment optimization, which iteratively refines both 6D poses and 3D features.

Furthermore, a SLAM system depends on the user input, especially on what kind of camera motion the user performs, and what kind of scene environment the camera observes. These runtime parameters heavily influence robustness and accuracy of both localization and mapping components. How are the connections between SLAM and user input? The 3D location (depth) of a feature is inferred from at least two camera frames providing corresponding 2D image observations of a real-world landmark, in a process called triangulation. Triangulation requires sufficient motion parallax, which is the angle between the rays going from the landmark to the camera centers (along the bearings of the 2D image observations). In theory, any parallax angle greater zero is sufficient for triangulation. In practice, however, triangulation error also depends on the image measurement quality, including noise and outliers.

Parallax itself is a function of camera motion and scene depth. In other words, there are two reasons why an observed landmark does not exhibit sufficient parallax: either because of insufficient camera motion or because of extreme depth of the landmark. For example, one can walk a long way and the stars in the sky will not exhibit parallax. Additionally, rotation-only movements, a prominent motion pattern often executed by hand-held device users [106], does not induce parallax and is considered a degenerate motion for depth triangulation (and 3D reconstruction).

Degenerate camera motion detection and keyframe selection are well-known problems in SfM [160], and similarly apply to visual SLAM. Monocular SLAM systems such as PTAM aim to avoid mapping camera frames which do not induce parallax to prevent the worst case, model corruption. For example, bundle adjustment optimization may converge to local minima, resulting in model corruption. This keyframe filtering comes at the price of mapping starvation and localization loss, because the observed scene regions are missing in the 3D model. This essentially means that SLAM systems such as PTAM cannot process arbitrary camera motion and scene structure.

The initialization of a 3D SLAM model requires – analog to the aforementioned triangulation problem – parallax-inducing general camera motion. Performing this kind of general camera motion is a tedious task for non-expert users, who often fail in initializing a SLAM system [106]. With increasing camera-scene distances, as it is often the case in outdoor environments, the problem only gets worse, as spanning the required parallax baseline forces the user to easily walk more than 10 meters in an awkward sideways motion [169].

Furthermore, two-image stereo initialization [45, 113] yields SLAM models with a local coordinate system and arbitrary scale. Arbitrary scales and coordinate systems are a problem for AR applications when displaying virtual objects: Where should it put the virtual objects and at at which size? The original PTAM initialization procedure employed a workaround by declaring the convention that the initial camera image pair should be manually taken roughly 10cm apart and perceive a planar scene structure such as a table surface, which was then extracted as the dominant plane from the initial 3D point cloud and provided the “playground” for top-level AR applications. Of course, these kind of assumptions and heuristics cannot be applied in general scenes.

In contrast, localizing the SLAM model with respect to known, ideally metric-scale and global coordinate systems such as UTM or WGS has many advantages. For example, location-based services such as geographic information systems can be queried using the same reference coordinate system, allowing for retrieving and uploading digital content. Visual wide-area localization, *i.e.*, self-estimation of the full 6DOF pose of a mobile camera device with respect to a given reference coordinate system, has been explored extensively. The literature includes approaches using motion sensors (*e.g.*, GPS, magnetometer and inertial sensors) [46], image databases [137], 3D reconstructions [66], textured 3D models [125], and digital elevation models [157]. However, wide-area localization algorithms that fulfill the practical requirements of AR applications in terms of ubiquitous availability, registration accuracy, and (near) real-time performance, are rare and still an open issue. For example, geo-localization algorithms for outdoor, and, especially, urban environments using hand-held camera devices require 3D reconstructions [6] and image databases [155], which are usually only available in selected regions. In contrast, untextured 2D city maps are available for nearly all urban environments of the world and can be retrieved from geographic information services such as OpenStreetMap.

Localization and mapping algorithms laid out for desktop computers often cannot be ported in a straight forward way to hand-held computers. For example, the PTAM algorithm defied trivial porting from the desktop to the mobile platform and required considerable modification and redevelopment. Still, the mobile version was considered "far less accurate and robust" [74]. As computational and sensorial limitations apply, SLAM on mobile devices is often restricted to low-level geometric features such as points and edges. However, high-level geometric map features such as planes would provide valuable information to AR applications for the generation of ortho-images and panoramas, as well as giving an advanced semantic understanding of the observed scene that allows for rendering meaningful annotations.

1.6 Approach

Most SLAM algorithms focus on the most general SfM case, that is, cameras which move in 6DOF and features which are located in 3D space. In other words, general SfM and SLAM algorithms do not make any prior assumptions about the scene structure or camera motion. As noted by Szeliski [154], in many cases, however, the scene contains higher-level geometric primitives such as lines and planes. These can provide information complementary to interest points and also serve as useful building blocks for SLAM. Furthermore, these primitives are often arranged in particular relationships, *i.e.*, many lines and planes are either parallel or orthogonal to each other. This is especially true of urban architectural scenes and models. Additionally, it is possible to take advantage of a constrained camera motion model. For example, when the camera moves in a fixed arc around some center of rotation, specialized techniques can be used to recover this motion and the observed structure.

We propose to approach the outlined limitations and problems of monocular visual SLAM by exploiting specific combinations of constraints on the performed camera motion, and on the observed scene structure and its semantics. In particular, we aim to exploit constraints on parallax-free camera movements, geometric primitives such as lines and planes, and semantic information extracted from images. These constraints allow us developing localization and mapping algorithms, which we integrate into efficient SLAM systems suited for restricted computing platforms such as hand-held devices.

Depending on the performed camera motion and the observed scene structure, features may exhibit a varying amount of parallax and can be modeled differently by a SLAM system. General visual SLAM approaches [32, 72, 110] assume parallax-inducing camera motion and perform 6DOF localization and finite 3D mapping. In contrast, panoramic visual SLAM approaches [104, 35, 90, 173] assume pure-rotation camera motion (or scenes with "infinite" depth). Thus, constrained panoramic SLAM models only allow to track the 3DOF rotation of the camera (and not the 3D location). We aim for the hybrid case, that is a SLAM algorithm which allows for models with mixed finite and infinite feature representations [25, 54, 119, 61]. These "unconstrained" SLAM systems truly allow for arbitrary camera motion and scene structure. Additionally, features may exhibit an increasing amount of parallax, when repeatedly observed over time. For example, a feature may initially be modeled with infinite depth, but later observed again with sufficient parallax to estimating a finite 3D location. This idea refers to inverse depth parameterization applied in filter-based SLAM [104] and deferred triangulation applied in keyframe-based SLAM [119, 61].

As one can observe easily, indoor as well as outdoor environments contain many human-made structures, such as walls, tables, billboards, and building façades, that can be geometrically modeled as planes. Observing these planar structures, the camera motion can be estimated with homographies, which describe the perspective transformation between two camera images via a plane. Homographies encode (up to scale) the relative Euclidean pose transformation (*i.e.*, the relative 6DOF pose) between the two cameras, as well as the corresponding plane. Thus, homographies provide strong constraints on both camera motion and scene structure, which we aim to employ for mapping planar scenes. Additionally, a homography can be decomposed [96] into a pose and plane equation, allowing to back-project 2D image features onto the estimated plane, as well as rendering orthographic images of the plane that can be used as localization templates by model-based tracking and detection algorithms.

Furthermore, certain architectural structures in urban environments, such as building façades, can be modeled as planes. These building façades usually provide many edge features located on windows, doors and other structures that can be modeled as vertical and horizontal lines. Vertical and horizontal lines allow estimating vanishing points and give strong constraints on the relative plane orientation. Again, this knowledge allows for rendering orthographic images of the façade. More than that, matching the extracted image lines and planes with a virtual reference model, possibly as simple as an untextured 2D city map, allows for geo-localization, that is, full global 6DOF pose estimation within the coordinate system of the reference model. Additionally, the matching of line correspondence between image and model can be facilitated by semantic knowledge retrieved from the image.

1.7 Contributions

With the outlined constraints at hand, we developed several visual SLAM algorithms especially for hand-held devices that enable – besides localization and mapping – applications such as geo-localization and high-level model generation (*e.g.*, orthographic and panoramic images). In particular, we describe the following contributions:

Homography-Based Planar Mapping and Tracking In Chapter 3, we present a real-time camera pose tracking and mapping system, which uses the assumption of a planar scene to implement a highly efficient mapping algorithm. Our light-weight mapping approach is based on keyframes and plane-induced homographies between them. We solve the planar reconstruction problem of estimating the keyframe poses with an efficient image rectification algorithm. Camera pose tracking uses continuously extended and refined planar point maps and delivers robustly estimated 6DOF poses. We compare system and method with bundle adjustment and monocular SLAM on synthetic and indoor image sequences. We demonstrate large savings in computational effort compared to the monocular SLAM system, while the reduction in accuracy remains acceptable.

Handling Pure Camera Rotation in Keyframe-Based SLAM Handling degenerate rotation-only camera motion is a challenge for keyframe-based simultaneous localization and mapping with six degrees of freedom. Existing systems usually filter corresponding keyframe candidates, resulting in mapping starvation and tracking failure. In Chapter 4, we propose to employ these otherwise discarded keyframes to build up local panorama maps registered in the 3D map. Thus, the system is able to maintain tracking during rotational camera motions. Additionally, we seek to actively associate panoramic and 3D map data for improved 3D mapping through the triangulation of more new 3D map features. We demonstrate the efficacy of our approach in several evaluations that show how the combined system handles rotation only camera motion, while creating larger and denser maps compared to a standard feature-based SLAM system.

Urban Outdoor Localization and SLAM Initialization In Chapter 5, we present a method for large-scale geo-localization and global tracking of mobile devices in urban outdoor environments. In contrast to existing methods, we instantaneously initialize and globally register a SLAM map by localizing the first keyframe with respect to widely available untextured 2.5D maps. Given a single image frame and a coarse sensor pose prior, our localization method estimates the absolute camera orientation from straight line segments and the translation by aligning the city map model with a semantic segmentation of the image. We use the resulting 6DOF pose, together with information inferred from the city map model, to reliably initialize and extend a 3D SLAM map in a global coordinate system, applying a model-supported SLAM mapping approach. We show the robustness and accuracy of our localization approach on a challenging dataset, and demonstrate unconstrained global SLAM mapping and tracking of arbitrary camera motion on several sequences.

1.8 Collaboration statement

The contributions of this thesis have been peer-reviewed and published as part of the papers listed below. We provide information on the collaborations which occurred during the work on these publications. For all papers, the author contributed to the development work, which involves design, implementation and analysis, as well as to the publication work. In particular:

- C. Pirchheim and G. Reitmayr. Homography-based planar mapping and tracking for mobile phones. In *Proceedings of the International Symposium on Mixed and*

Augmented Reality (ISMAR'11), pages 27–36. IEEE, Oct. 2011

Gerhard Reitmayr provided guidance on development and publication.

- C. Pirchheim, D. Schmalstieg, and G. Reitmayr. Handling pure camera rotation in keyframe-based SLAM. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'13)*, pages 229–238. IEEE, Oct. 2013

  Gerhard Reitmayr and Dieter Schmalstieg provided guidance on development and publication.

- C. Arth, C. Pirchheim, J. Ventura, D. Schmalstieg, and V. Lepetit. Instant Outdoor Localization and SLAM Initialization from 2.5D Maps. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'15)*, 2015

  The author and Clemens Arth equally contributed to development and publication, while the other authors contributed in varying degrees. Vincent Lepetit and Dieter Schmalstieg provided guidance on development and publication. The author, Clemens Arth and Jonathan Ventura contributed to the implementation. In particular: Clemens Arth provided the the semantic segmentation and the depth-based SLAM initialization. Jonathan Ventura conceived the original minimal solvers. Additionally, the BSP tree renderer was implemented by Christoph Klug. The author developed the final localization and evaluation frameworks, integrating all the above contributions.

The following people and institutions deserve specific mentioning, since a considerable part of this thesis would not have been possible without them:

- The author worked on this thesis as a member of the Christian Doppler Laboratory for Handheld Augmented Reality, which was supported by the following institutions: Qualcomm Technologies Incorporated, Christian Doppler Research Association, and Graz University of Technology.

- The research and development work at the Christian Doppler Laboratory for Handheld Augmented Reality was guided by its director, Dieter Schmalstieg, its affiliated professors, Gerhard Reitmayr (2009-2013) and Vincent Lepetit (2014-2015), and its deputy directors, Daniel Wagner (2007-2009), Hartmut Seichter (2009-2014) and Clemens Arth (2014-2015).

- The actual and former members and associated members of the Christian Doppler Laboratory for Handheld Augmented Reality were involved in discussions on the research work presented in this thesis (alphabetically): Raphael Grasset, Lukas Gruber, Jens Grubert, Andreas Hartl, Christoph Klug, Tobias Langlotz, Alessandro Mulloni, Thanh Nguyen, Qi Pan, Markus Tatzgern, Jonathan Ventura.
- Valuable support was provided by Daniel Wagner, Erick Mendez and Alessandro Mulloni from Qualcomm Research Austria.

2 Background

In the fictional Augmented Reality (AR) interface shown in Figure 2.1, the view of the world is enhanced with digital information about what the user (a humanoid robot, in this case) is seeing. This example foreshadows that AR is a interdisciplinary field, which draws from diverse other fields such as photogrammetry (making measurements from images and other sensor data), computer vision (estimating scene geometry, structure from motion), computer graphics (3D coherent rendering), human-computer interaction (having the user in the loop), and artificial intelligence (learning and interpreting the scene).

The classical definition of Augmented Reality comes from Azuma [7]: AR systems combine real and virtual objects, are interactive in real-time, and registered in 3D. This definition deliminates AR from offline and non-interactive computer graphics applications employed in movie and computer game industries. Furthermore, according to Feiner [47], Virtual Reality (VR) "aims to replace the real world", whereas AR "respectfully supplements it". AR is also related to Mediated Reality (MR) and Diminished Reality (DR) concepts, where the world is modified by adding virtual objects or removing real objects, especially by means of wearable computers or hand-held devices [97].

Figure 2.1: Augmented Reality user interfaces shown in the movie *Terminator 2: Judgment Day* (1992).

One core component of each AR system is registration, which aligns real and virtual worlds. The registration problem includes the following sub-problems:

Localization

Depending on community and context, also known as tracking, detection, re-localization, pose recognition, place recognition, self-localization, ego-motion estimation.

Mapping

Depending on community and context, also known as reconstruction, modeling.

The localization problem refers to continuously estimating the motion (*e.g.*, the 6D pose consisting of 3D orientation and 3D position) of the camera (or the computing device, the user, etc.) with respect to a coordinate system given by a virtual model. The resulting motion estimate is used to render a perspectively correct mixed-world AR view. The mapping tasks refers to creating the required virtual models, *e.g.*, reconstructing the 3D scene structure visible within a given set of images.

There is a vast amount of literature describing algorithms for localization and mapping problems, not only in the field of AR, but also in the fields of photogrammetry, computer vision, robotics, and others. The algorithms differ in many respects including the employed computer platform and sensors, the achieved real-time performance, the required user intervention, the intended application environment, and the assumed a-priori knowledge.

In the context of this thesis, we cannot provide a comprehensive literature overview. Instead, we are present a selection of relevant work with respect to the contributions of this thesis. In the following, we want to state more precisely the scope of this chapter.

Computing platforms We give a focus on algorithms that have been developed for hand-held devices or have been applied on hand-held devices. Algorithms may be laid out for such diverse platforms as computer clusters, desktop/laptop computers, and hand-held computers, as well as for network-connected combinations of these platforms. In Table 2.1, we show the specifications of two mobile phones, which also illustrates the rapid evolution of their hardware capabilities. However, in comparison to hand-held devices, desktop and cluster computers have a processing power that is several magnitudes higher.

Motion and inertial sensors In our contributions, we are partially using the motion and inertial sensors built into hand-held devices. Generally, localization and mapping algorithms may employ very different kinds of sensors, and each kind may range from

	Nokia N900	Samsung Galaxy S6
Year	2009	2015
CPU	600 MHz	Quad-core 2.1 GHz
GPU	PowerVR SGX530	Mali-T760MP8
RAM	256 MB	3 GB
Camera	5 MP, 2576 x 1936 pixels	16 MP, 2988 x 5312 pixels
GPS	A-GPS	A-GPS, GLONASS, Beidou
Display	800 x 480 pixels	1440 x 2560 pixels

Table 2.1: Comparison between Nokia N900 (2009) and Samsung Galaxy S6 (2015).

high-end state-of-the-art to low-end consumer-device quality. For example, hand-held devices mostly provide assisted GPS (A-GPS) receivers, whereas state-of-the-art differential or real-time kinematic GPS (D-GPS, RTK-GPS) provides much more accurate position estimates (*e.g.*, several meter vs. centimeter accuracy w.r.t. ground truth). This is similarly true for motion and inertial sensors, such as magnetometers, gyroscopes and accelerometers, as well as for camera hardware.

Camera sensors We give a focus on algorithms that employ monocular RGB cameras (*i.e.*, cameras with a single lens), preferably built into hand-held devices. Hand-held devices are typically equipped with comparably small field-of-view lenses and low-quality camera sensors, resulting in images with small dynamic range and heavy motion blur, even under relatively good lighting conditions. Catadioptric (*e.g.*, fisheye, omnidirectional) camera systems providing wide field-of-view or panoramic imagery are not available on hand-held devices without special externally mounted lens extensions. With a few exceptions, hand-held devices also do not provide stereo camera rigs.

Depth sensors We will also briefly describe algorithms using RGB+Depth (RGB-D) camera sensors, because it is likely that these sensors will be deployed to consumer hand-held devices in the future. Since the release of the Microsoft Kinect in 2011, depth and 3D motion sensors have become immensely popular in the research community, especially for real-time Simultaneous Localization and Mapping (SLAM). Since then, similar products were presented, either based on time-of-flight or structured light approaches. Depth sensors are usually combined with RGB cameras and, thus, called RGB-D sensors, providing synchronized RGB and depth buffers. Popular examples for RGB-D sensors are Apple PrimeSense and Mantis. The latter is installed in the experimental Google Tango device, making it the first hand-held device prototype with a built-in RGB-D sensor. The depth range depends on the parallax baseline, *i.e.*, the baseline between the structured infrared light projector and the corresponding infrared camera. For example, the Google Tango depth sensor gives values between 0.5 and 4 meters.

Real-time performance We focus on real-time localization and mapping algorithms. Another major distinction between registration algorithms is whether or not they can cope with real-time constraints. AR requires algorithms with real-time or near real-time performance, as they must process the incoming input and present an output within specified time constraints, *e.g.*, the live video stream of an camera must be processed online at frame-rate. In contrast, for example, certain structure from motion algorithms assume all input data available at once for offline batch processing, rendering these algorithms unusable for AR.

Interactivity We focus on algorithms which run automatic with minimal user interaction. In contrast, there are algorithms which require users to provide hints for localization and mapping, in particular interactive and semi-interactive modeling algorithms. However, an AR system is implicitly interactive, since the user is in the loop, moving the hand-held device and providing system input, *e.g.*, by operating the camera and observing a certain desired scene. Additionally, in certain situations, such as localization failure, the user is required to adopt its behavior and provide input that lets the system recover. During normal operation, however, our algorithms do not require any specific manual intervention. Moreover, our algorithms should adopt to arbitrary user input, leaving a maximum of freedom to the user.

Scene environments We focus on algorithms with practical value, *i.e.*, for usage in tough real-world conditions. Algorithms may be laid out for very different environments, *e.g.*, indoor or outdoor environments. The conditions outside of the laboratory are usually way more tough, including difficult lighting conditions (sunlight, shadows), dynamic objects (occlusions), larger camera-scene distances, etc.

A-priori knowledge The related work can be categorized by their assumed input, especially the a-priori knowledge about their runtime environment. We focus on the representation of the virtual models, which can be very different, with implications on the difficulty of the problem and, consequently, the real-time performance and practical applicability for AR. Following the Extent of World Knowledge (EWK) continuum by

Milgram and Kishino [102], we distinguish between algorithms which assume:

- **Completely or partially modeled environments** (Section 2.2)
- **Unmodeled environments** (Section 2.3)

We discuss localization and mapping approaches for the first category in Section 2.2 and the second category in Section 2.3.

Structure from Motion Since we focus on real-time mapping algorithms, offline Structure from Motion (SfM) approaches are beyond the scope of this thesis. However, 3D reconstruction and modeling may provide the virtual models which are employed by real-time localization algorithms. For instance, the interested reader is referred to Musialski *et al.* [107] which provide a recent and comprehensive survey of interactive and automatic reconstruction techniques from computer graphics, computer vision, photogrammetry and remote sensing communities. Furthermore, the survey of Yin *et al.* [182] covers manual reconstruction techniques, including the generation of 3D building models from 2D architectural drawings. We also want to refer to automatic and semi-automatic SfM systems, including offline batch SfM [150, 1, 49], interactive modeling based on offline SfM [167], incremental SfM [180], and real-time SfM [120]. The latter closes a circle, as Photogrammetry and Computer Vision research, in general, and Structure from Motion, in particular, provide the fundamentals for real-time localization and mapping.

2.1 Vision-based registration fundamentals

Overall, we focus on computer vision-based registration approaches that have been enabled by advances in feature detection and matching, feature-based alignment, structure from motion, and 3D reconstruction. Szeliski has written an excellent textbook covering these and further topics [154].

Feature detection and matching allows for establishing correspondences between two or more images. Feature detection refers to measuring so-called natural features or interest points, such as corners [57, 141, 129], edges [57, 18], line segments [56] and blobs [86, 98]. Corresponding features between two or more images can either be found using local or wide-baseline matching techniques. Local template matching involves the computation of image correlation such as Normalized Cross Correlation (NCC) [186], or direct image alignment techniques such as Kanade-Lucas-Tomasi (KLT) [94, 161, 11]. Wide-baseline matching usually involves computing feature descriptors [92, 13, 17]. Feature descriptors provide a condensed information about a feature location, and allow for descriptor matching which should be invariant to affine or even perspective image transformations, as well as to illumination changes. In this context, we recommend the surveys on image registration/alignment by Zitova [187] and Szeliski [153].

The resulting image feature correspondences are employed for feature alignment. Feature alignment involves the robust estimation of intrinsic and extrinsic camera parameter as well as motion models such as homographies and epipolar geometry. In this thesis, we employ a-priori internally calibrated cameras based on established parameterization and distortion models [59, 185], which we estimated using the GML Camera Calibration Toolbox of Alexander Velizhev[1], an extension of the well-known calibration framework of Bouguet[2]. Robust pose and motion model estimation refers to minimal solvers[3] such as the three-point absolute pose problem [48] and five-point relative pose problem [113]. Furthermore, we have adopted the mathematical notation introduced by Lepetit and Fua in Chapter 2 of their survey on model-based tracking [81]. This text covers fundamental tools for computer vision, including internal and external camera pose parameterization, as well as robust estimation and iterative minimization techniques such as RANSAC [48], robust least-squares using M-estimators [65], and bundle adjustment [163]. Additionally, we found the lecture notes of Drummond [37] on Lie groups and algebras very helpful, *e.g.*, for the parameterization of Euclidean transformations in 3D space.

Feature detection, matching and alignment are important fundamentals for Structure from Motion and 3D reconstruction. The corresponding standard textbook from Hartley and Zisserman [58] extensively covers topics such as projective geometry (*e.g.*, projective transformations), single view geometry (*e.g.*, camera models, pose estimation, vanishing points), two- and multi-view geometry (*e.g.*, epipolar geometry, homographies, triangulation, 3D reconstruction), and robust least squares minimization techniques (*e.g.*, iterative estimation, bundle adjustment). The survey papers on Visual Odometry from Scaramuzza and Fraundorfer [135, 50] cover similar topics.

[1]GML C++ Camera Calibration Toolbox: `http://graphics.cs.msu.ru/en/node/909`

[2]Camera Calibration Toolbox for Matlab: `http://www.vision.caltech.edu/bouguetj/calib_doc/`

[3]Minimal problems in computer vision: `http://cmp.felk.cvut.cz/minimal/index.php`

2.2 Localization in modeled environments

This section discusses registration in at least partially modeled environments, in particular localization methods with respect to some a-priori known virtual environment model. In the following, we want to distinguish between two sub-problems:

Initial localization
Also known as self-localization, tracking by detection, re-localization.

Continuous localization
Also known as tracking, recursive tracking.

Initial localization is performed from scratch without or with only a coarse prior, *e.g.*, referring to pose estimation using correspondences from wide-baseline matching. Continuous localization is performed after initialization, when already having a reasonable prior, *e.g.*, referring to iterative pose estimation using correspondences from local template matching with the previous frame or a keyframe. Generally, initial localization is considered a harder problem than continuous localization and, consequently, requires more computation effort and time.

In the following, we want to classify localization approaches with respect to the representation of the employed virtual model. The model representation can be very different, including sparse models (*e.g.*, consisting of point, edge features), and dense models (*e.g.*, consisting of image pixel or volumetric voxel features, and polygonal vertex meshes). Furthermore, models can be textured or untextured, and have two (2D), two-point-five (2.5D), or three dimensions (3D). In particular, we want to distinguish the following cases:

Localization in prepared environments (Section 2.2.1)
Prepared environments are equipped with localization infrastructure that can be used to register with respect to a given model coordinate system.

Localization with respect to untextured models (Section 2.2.2)
Untextured models consist of vertices, but have to texture, including 2.5D digital elevation models, 2D and 3D polygonal models.

Localization with respect to reference images (Section 2.2.3)
The reference images (a.k.a. keyframes) itself are registered with respect to a model.

Furthermore, the complexity of the localization problem depends very much on the *scope* of the model. By model scope, we understand the area or volume covered by of the model, up to scale. For example, localization with respect to a model covering an entire city (let's say, Paris) is considered a much harder problem than with respect to model that contains only a single object (let's say, the Eiffel tower). The scale of the model, however, makes no fundamental difference. Whether to localize the real-world Eiffel tower or its small-scale paper model is considered a similar problem, except for environmental issues such as lighting. Naturally, it will make a difference on the accuracy of the pose in metric scale (*e.g.*, localization with respect to the smaller model will likely give a higher accuracy than the large real-life model). For AR purposes, however, the major camera pose accuracy concern is the achievable quality of the real-virtual overlay, which will likely be the same.

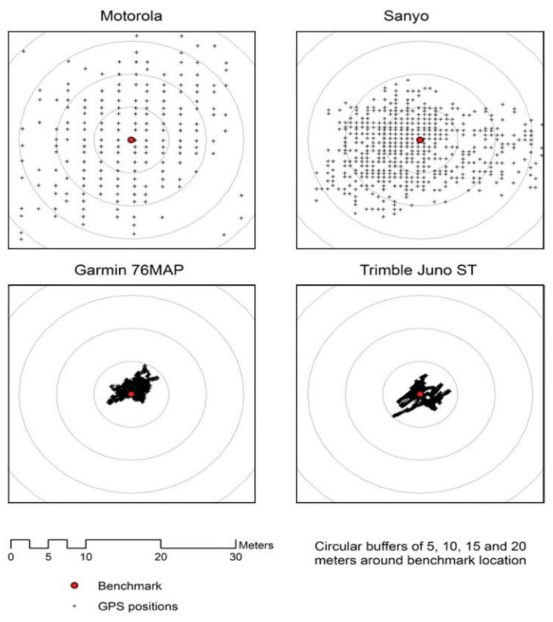
Consequently, in the following discussion, we will roughly distinguish between wide-area and local-area localization. However, the distinction is fuzzy, and the related work difficult to classify. One can say that wide-area localization expects the model to exist where the user is, while local-area localization requires the user to go where the model exists.

2.2.1 Prepared environments

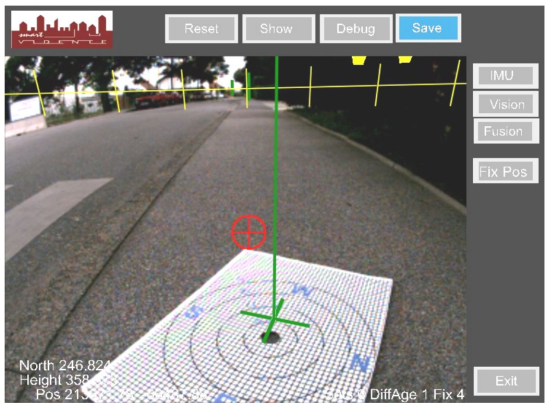
Localization infrastructure in prepared environments includes Global Navigation Satellite Systems (GNSS), registered WiFi access points, and mobile telecommunication base stations. This infrastructure provides coordinate systems to which an AR system can register, and which we take advantage of as priors for vision-based geo-localization in Chapter 5.

Several global and regional GNSS systems are in operation, including the United States' Global Positioning System (GPS) and Russia's GLONASS system. GNSS allow for determining the 3D position of a satellite signal receiver with respect to global coordinate systems such as Universal Transverse Mercator (UTM) or World Geodetic System (WGS). The precision and latency of the positioning depends very much on the receiver quality and on satellite signal availability. For example, satellite signals interfere with human-made structures such as buildings, and consequently can only be received with limited quality in indoor and certain urban environments (*e.g.*, urban canyons).

Early on, mobile AR registration systems have exploited GPS receivers in combination with other sensors, such as digital compass and inertial orientation sensors, to perform global 6D localization in outdoor environments. Being one of the first such systems, Feiner *et al.* [46] found that normal GPS readings were only accurate within about 100m, while differential GPS allowed to achieve about one-meter accuracy. Differential GPS is



(a) © [184]



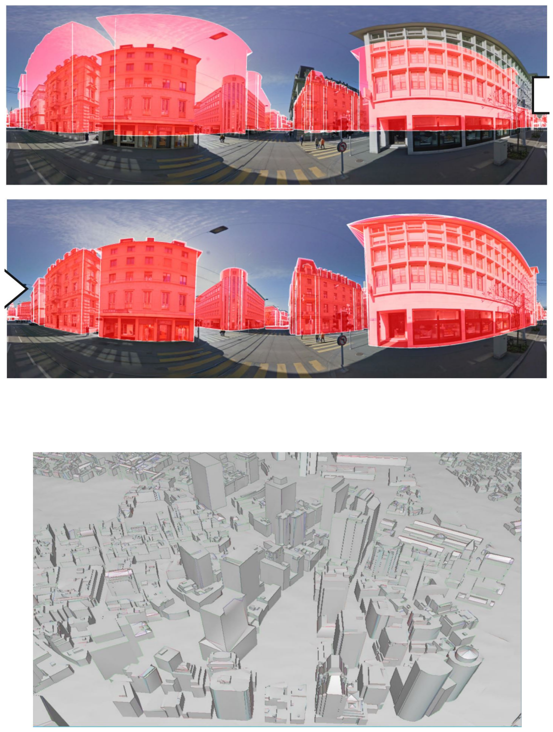
(b) © [136]

Figure 2.2: Outdoor localization using GPS: (a) Zandbergen *et al.* [184] evaluated A-GPS on off-the-shelf mobile devices and achieved accuracies within 5.0m and 8.5m. (b) Schall *et al.* [136] achieved sub-meter to centimeter-accuracies using high-end D-GPS and RTK-GPS sensors.

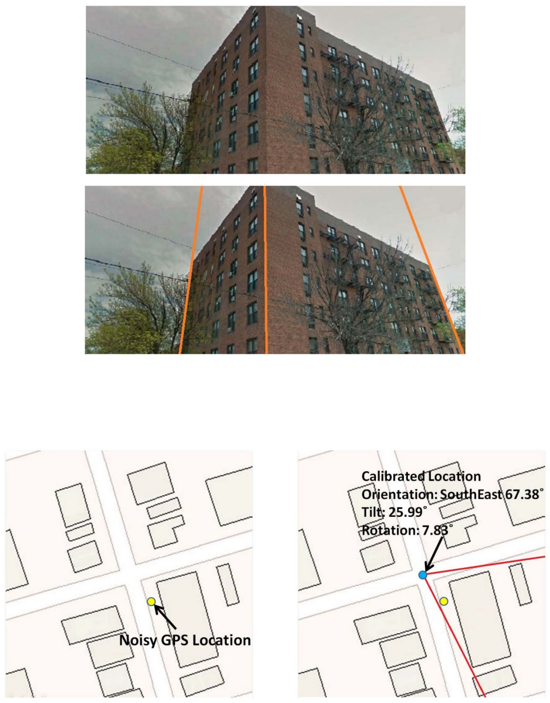
coupled with the correction signal of another receiver at a known location that contains information about how far it is off. More recently, Schall *et al.* [136] presented a system that employed high-end sensors and implemented Kalman filtering for fusion of Differential or Real-Time Kinematic GPS (D-GPS, RTK-GPS). They report sub-meter to centimeter accuracy position estimates. Current generation phones and tablets are typically equipped with Assisted GPS (A-GPS) receivers, which allow for accelerated 3D positioning. According to a recent study by Zandbergen *et al.* [184], using A-GPS techniques enables mobile phones to achieve positioning even in urban canyons and indoor locations. Outdoors, the median horizontal position error lies between 5.0m and 8.5m. Indoor errors are larger. However, very large errors are very uncommon and lie within 30m outdoors and within 100m indoors. See also Figure 2.2.

2.2.2 Untextured models

Another class of localization approaches relies on untextured models, such as 3D CAD models, or widely available 2D cadastral maps, which are additionally annotated with



(a) © [157, 123]



(b) © [24]

Figure 2.3: Urban outdoor localization with untextured models. (a) Taneja *et al.* [157] align panoramic query images with 2.5D elevation models. (b) Chu *et al.* [24] align query images with 2D city maps using building façade outlines.

per-building height information or digital elevation data (often from LIDAR), resulting in 2.5D models. Our contributions on geo-localization and SLAM initialization described in Chapter 5 employ such untextured 2.5D models.

Untextured models have been used for initial and continuous localization in both local and wide areas. In particular, we are going to discuss initial geo-localization in wide areas with respect to 2D cadastral and 2.5D digital elevation models which use single or multiple images. Furthermore, we will explain the restrictions made by previous work in this area. In particular, we observe that the methods either do not provide real-time performance or accuracy which is insufficient for AR purposes.

2.5D digital elevation models Ramalingam *et al.* [123] establish the registration between an image and a 2.5D model by computing 3D-2D line and point correspondences. However, an already registered second image is required to establish the 3D-2D correspondences by first matching 2D-2D SIFT features between the two images. Consequently, the first image of a sequence needs to be manually annotated. In contrast, our methods are fully automatic and only require a single input image.

Baatz *et al.* [8] use contour matching and refinement of sky silhouettes between a digital elevation model and mountain images. Similarly, Bansal *et al.* [12] verify pose hypotheses by matching the pictured image skyline with the model. In AR, users should not have to point the camera at the skyline. In dense urban areas, the skyline may not even be easily visible. Therefore, we do not rely on a visible skyline in our input images.

Another work from Baatz *et al.* [9] registers semantically labeled images with respect to labeled 3D digital terrain models. The idea of semantic segmentation is related to our method, but Baatz *et al.* process landscape images, whereas we process urban images. They state that "it is still extremely challenging to accurately identify the semantics" and restrict themselves to four of the easier classes (sky, water, settlements, other). This approach allows Baatz *et al.* to estimate only the orientation of the image with respect to the model. In contrast, our method computes both absolute orientation and 3D location.

With Taneja *et al.* [157], we share the idea of using semantic image segmentation as input for the registration with a 2.5D map (see Figure 2.3). However, Taneja *et al.* optimize the pose over a continuous 6D cost space, while we are verifying pose hypotheses at discrete positions within a 2D cost space, which makes our method arguably much faster. Taneja *et al.* use comparably detailed 2.5D models and Google StreetView images for their queries, which are high-resolution panoramas with a rather accurate initial geo-location. The quality of this data likely allows them to optimize all 6DOF simultaneously without getting stuck in local minima.

2D cadastral maps David *et al.* [31] register panoramic images with 2D maps using a building façade orientation descriptor. As shown by Arth *et al.* [6], the higher amount information contained in wide field-of-view images significantly increases the success rate of localization. Mobile devices have a narrow field of view, and a descriptor such as the one used by David *et al.* is not discriminant enough in such situations.

Cham *et al.* [21] also consider panoramic images and aim to detect vertical building outlines and façade normals, resulting in 2D fragments which are matched with a 2D map. Chu *et al.* [24] report to outperform Cham *et al.* [21]. Chu *et al.* compute a descriptor from vertical building outlines in perspective input images, which is then matched with a 2D map. They published their code and dataset online: Upon careful analysis, the dataset consist of only 11 scenes that mostly show free-standing buildings with rectangular footprint. Matching requires the exact detection of the left, middle and right building outline in the input image. To facilitate the detection of vertical edge and vanishing points, they partially annotated the input images manually. In contrast, we aim at a fully automatic method.

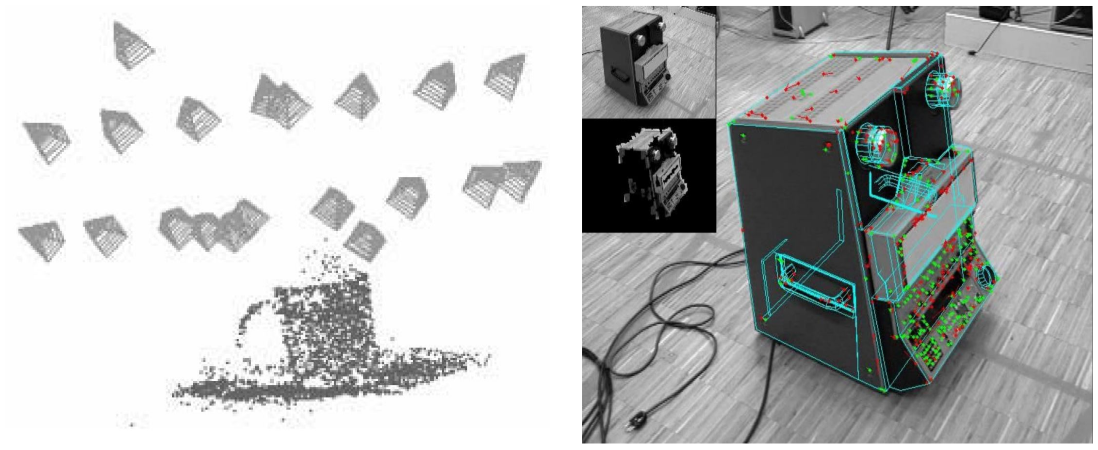
We share the assumption of horizontal and vertical image lines with Chu *et al.* [24] (see Figure 2.3). In contrast, however, we use a robust algorithm for orientation estimation, and consider a large number of potential vertical building outlines. The resulting pose hypotheses are verified based on a semantic segmentation of the image, which adds another layer of information to the pose estimation process. This allows our method to be applied to much more complex images compared to Chu *et al.*

2.2.3 Reference images and models

In this section, we review localization methods which employ a collection of reference images. The reference images (a.k.a. keyframes, image databases) are registered with respect to a model, such as untextured 3D CAD models or 3D models reconstructed from the reference images.

Image-based methods have been widely used for both initial and continuous localization in local as well as wide areas. In the AR context, initial and continuous localization usually refers to model-based detection and tracking, respectively. These techniques are also relevant for keyframe-based SLAM, where they are referred to as re-localization and map tracking, respectively. Initial localization in wide-areas with respect to a global coordinate system is often called geo-localization. In particular we describe the following techniques:

- Recursive homography tracking of manually selected planar scene models.
- Detection and tracking of keyframes which are registered with a 3D model.
- Planar target detection and tracking on mobile phones which uses collections of reference images as individual planar tracking targets.
- Tracking by synthesis which uses textured 3D models.
- Place and pose recognition which uses reference image databases, registered with 3DOF or 6DOF within a model, respectively. Consequently, the 3D position or 6D pose of a query images is returned.



(a) © [148] **(b)** © [83]

Figure 2.4: Model-based detection and tracking. (a) Skrypnyk *et al.* [148] first reconstruct a sparse point-feature model of the "mug" scene for later camera tracking by detection. (b) Lepetit *et al.* [83] use a CAD model and registered keyframes for point and edge feature-based real-time tracking.

2.2.3.1 Planar recursive tracking

Related to our contributions on planar SLAM presented in Chapter 3, the following methods perform recursive homography tracking with respect to a planar scene model, which is acquired at runtime. The model consists of one or more planar scene regions, which are manually defined by the user at runtime, *e.g.*, by selecting the vertices of a delineating polygon within the live camera view. These methods have several disadvantages: As no real-time mapping is performed, the selected plane has to stay within the camera frame, and recursive pose tracking starts to drift very quickly. Pioneering the real-time tracking of natural features and robust homography estimation, Simon *et al.* [147] presented a "markerless" recursive camera tracking system designed for unprepared environments which contain one or more planes. The actual 6D camera pose of the current frame is computed recursively by homography chaining, *i.e.*, multiplication of the previous pose with the current homography, which is estimated between the previous and current frame. Prince *et al.* [122] proposed a similar system. Later on, Simon *et al.* [146] and Lourakis *et al.* [88] suggested to employ the gathered knowledge about scene structure and camera motion for offline planar scene reconstruction.

2.2.3.2 Model-based detection and tracking

The following systems expect a 3D model with registered reference images to perform real-time camera detection and tracking. The applied techniques are important precursors for keyframe-based SLAM and wide-area image-based localization. We want to point out the fundamental difference between detection (initial localization) and tracking (continuous localization). Furthermore, we review two approaches for 3D model acquisition, one employing SfM models and the other Computer Aided Design (CAD) models.

Skrypnyk *et al.* [148] present a system for camera tracking by detection (initial localization) using SIFT that operated with 4-6 fps at the time. The system operates in two stages. First, in the offline stage, the manually selected reference images are registered with respect to a sparse point model using a "classic" SfM pipeline (see Figure 2.4). Second, in the online stage, the 6D camera pose is estimated with respect to the model by matching the current frame with the reference images (and implicitly, the 3D model). Both stages employ SIFT feature detection, description and matching to generate the required 2D-2D and 2D-3D feature correspondences.

Employing a 3D CAD model with registered reference images (they call them keyframes), Lepetit *et al.* [83] and Vacchetti *et al.* [166] combine initial detection with continuous real-time tracking (see Figure 2.4). For their real-time tracker, they detect corner features in the current frame, which are matched with both the previous frame and a selected keyframe, effectively combining recursive with keyframe tracking. For the latter, patches sampled from the keyframes are warped (with an affine transformation that sufficiently approximates a homography) into the current camera viewpoint, producing a synthesized image. The real-time tracker was reported to run with 15fps to 25fps. Later on, these pure point-based methods were extended with edge-feature tracking capabilities [165]. Thus, they achieved a tracking system which combines point and edge tracking and that can handle both textured and untextured models.

2.2.3.3 Planar target detection and tracking on mobile phones

Applying many lessons learned from desktop approaches, the work of Wagner *et al.* [174, 176, 175] pioneered real-time visual detection and tracking on mobile phone computing platforms. They use constrained models, that is, collections of reference images, where each image is registered onto a 2D plane, resulting in a database consisting of planar tracking targets. Essentially, Wagner *et al.* introduced three enabling techniques: First and second, for detection they adopted the original SIFT [93] and Ferns [82] wide-baseline matching

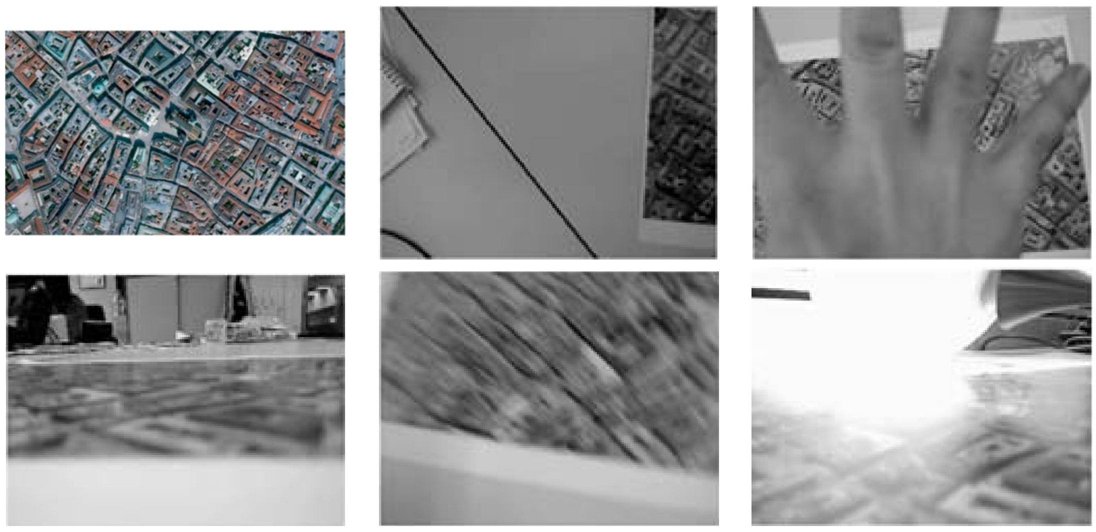


Figure 2.5: The mobile planar target detection and tracking method of Wagner *et al.* [175] is capable of handling considerable amounts of occlusion, tilt, motion blur, light reflection with respect to the template (top left). © [175]

frameworks for the mobile platform, resulting in the heavily modified PhonySIFT and PhonyFerns descriptors. Third, they presented a very efficient and robust template-based patch-matching tracker.

Using PhonySIFT in comparison to the original SIFT, the Difference of Gaussian (DoG) blob detector is replaced with the FAST corner detector [129] on an image pyramid, the descriptor length is truncated from 128 to 36 elements, and, instead of a k-d tree, a forest of spill trees is used as the feature database. Tracking by detection is performed by establishing feature matches between the current camera frame and the feature database, resulting in 2D-3D correspondences, which are geometrically verified with RANSAC using a homography motion model. The verified 6D pose can be immediately used as initial prior by the template-based tracker. Based on this prior, this tracker transforms known corner features from the reference image into the current frame. More specifically, it warps small patches using local affine transformations, followed by a search within a small window using NCC to match the best fit. To enable robust tracking fast camera motion, this warp/search strategy is performed on multiple image scales (see Figure 2.5). At devices of the time (*e.g.*, an ASUS P552W with an 624MHz CPU), this algorithm allowed for tracking up to six planar target at 17fps using a resolution of 320x240 pixels. For a single target, detection ran with about 15fps, while the template tracker was only limited by the frame rate of the camera.

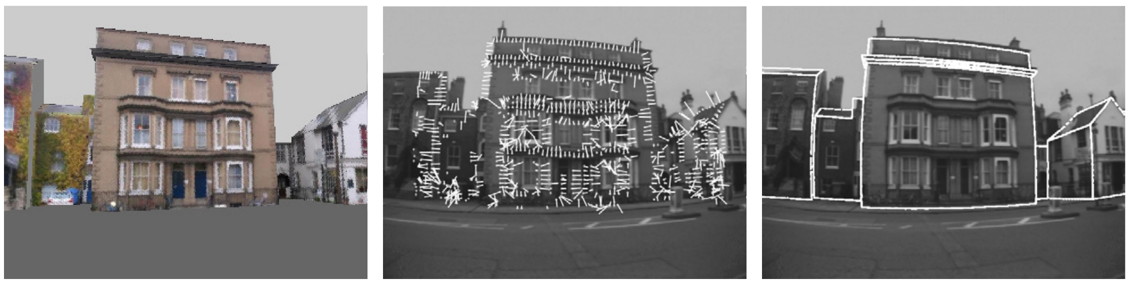


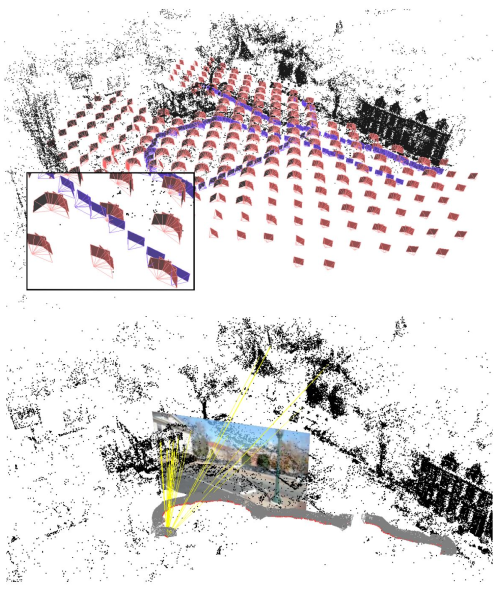
Figure 2.6: The tracking by synthesis method of Reitmayr and Drummond [124] renders a textured 3D CAD model using a pose prior, and performs edge matching with the current frame to estimate an updated 6D pose. © [124]

2.2.3.4 Tracking by synthesis

Tracking by synthesis methods employ a textured 3D model. Given a pose prior, the 3D model can be efficiently rendered with Graphics Processing Units (GPUs), resulting in depth and color buffers which are used for point [145] and edge [124] feature tracking. However, tracking by synthesis requires pose initialization. As part of our contributions presented in Chapter 5, we propose to employ rendered depth buffers of untextured models for 3D SLAM map initialization.

In contrast to edge-base tracking systems which employed untextured 3D CAD models [38, 70], Reitmayr and Drummond [124] presented a real-time visual registration system that combines inertial-visual sensor fusion [71] and edge-based tracking by synthesis of textured 3D models (see Figure 2.6). Given a pose prior, the textured 3D model is rendered with OpenGL, and both the frame buffer and the depth buffer are read back. First, an edge detector [18] is applied to the framebuffer. Then, the frame buffer is used to match the detected edge features with the current frame using NCC. Finally, the depth buffer provides the required depth information to establish the 2D-3D correspondences for 6D pose estimation. The system also integrates a point-based recovery mode, which was inspired by Lepetit *et al.* [83] (see above), but selects the required keyframes automatically.

Applied in urban outdoor environments, the system was later extended with a GPS-based initialization method [125]. The visual system is initialized with the 2D GPS position plus 1D average user height and the absolute 3DOF orientation from the inertial sensor fusion. However, since the visual tracker required a rather accurate initial camera pose (sub-meter position accuracy), they apply a search strategy in the local neighborhood of the GPS estimate.


(a) © [66]

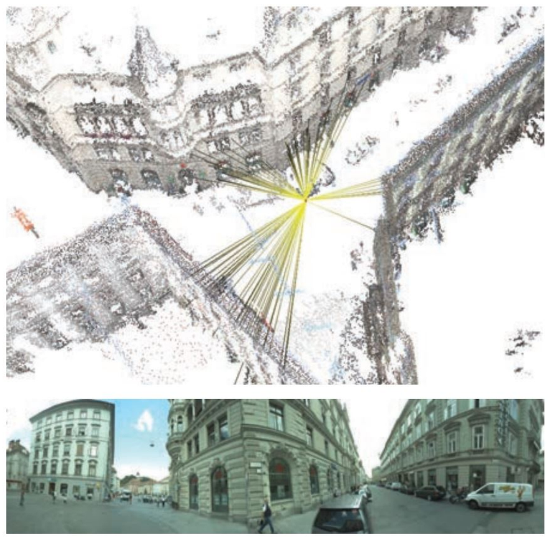

(b) © [2]

Figure 2.7: Image-based urban outdoor 6D pose recognition with respect to 3D point-cloud reconstructions and reference image databases. (a) Irschara *et al.* [66] use virtual views and small field-of-view query images. (b) Arth *et al.* [2] use wide field-of-view panoramic query images.

2.2.3.5 Image-based localization: place and pose recognition

Image-based localization, *i.e.*, place or pose recognition, refers to wide-area localization methods which take one or more query images and optionally a sensor prior (*e.g.*, location from GPS, orientation from compass and magnetometer) to perform appearance-based image retrieval with respect to large reference image databases. The reference images are pre-registered within models, *e.g.*, 3D reconstructions built from the reference images itself. The models allow for localization with varying degrees of freedom. Collectively, these methods employ content-based image retrieval, vocabulary trees and similar data structures that help to manage the enormous amounts of data, and query the databases efficiently.

The major disadvantages of image-based localization approaches remain that they do not scale well: Many images need to be captured for each new location, and, even with sufficiently dense sampling, it is still very challenging to match images under changing conditions due to illumination, season, construction activity and many other sources of change.

With respect to our contributions on wide-area localization presented in Chapter 5, we discuss place and, especially, pose recognition methods which allow for registering the full 6D pose of the query (and reference) images.

Place recognition Place recognition methods only allow for registering the 3D location of the query (and reference) images. Schindler *et al.* [137] demonstrated image-based 3D place recognition using databases that contain 20 km of urban street-side imagery, organized in a vocabulary tree [115] that is aimed to contain only the most informative features to improve the retrieval performance and to allow for increasing the number of reference images. Applying similar techniques, Zamir *et al.* [183] and, later, Vaca-Castano *et al.* [164] showed that it was possible to use existing image collections, such as Google StreetView, to perform city-scale place recognition. In the same spirit, Takacs *et al.* [155] implemented a distributed place recognition system by partitioning a server-based feature database, containing about 2500 images and covering roughly a city block, into "loxels". Per request, only the relevant loxels are transmitted to mobile phone clients based on a location prior received from GPS. Only the remaining reference images contained in the transmitted loxels are compared with the query image directly on the mobile client.

Pose recognition Being one of the first papers on image-based localization, Robertson *et al.* [128] performed pose recognition with respect to reference images showing building façades which are registered within a city model. They perform image-based content retrieval and pose estimation based on wide-baseline matching between the rectified façade views of the query and reference images.

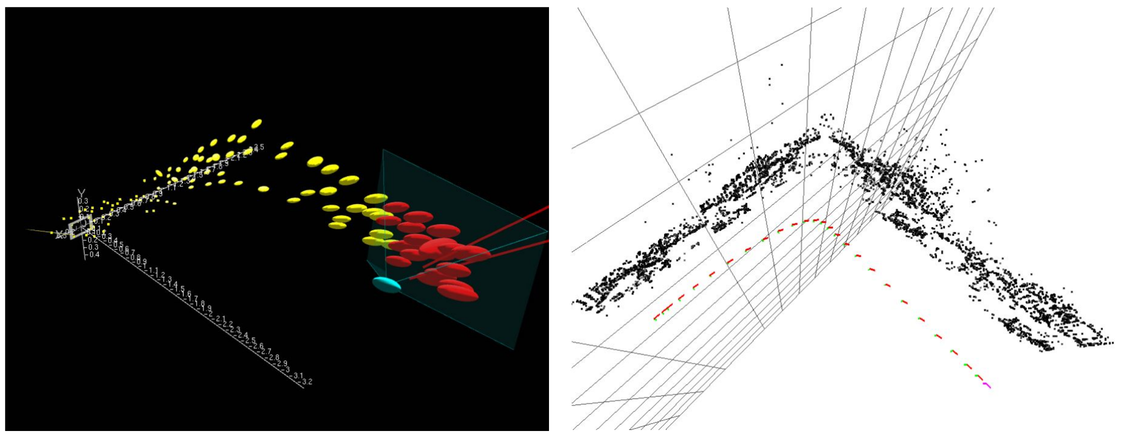
Depicted in Figure 2.7, Irschara *et al.* [66] presented a 6D image registration algorithm with respect to large 3D point-cloud reconstructions that performed in real-time on a powerful desktop computer. The algorithm is enabled by an efficient vocabulary tree-based search strategy that returns entire 3D model fragments instead of individual feature descriptors. Upfront, reference images are registered within a reconstructed 3D model, and complemented with synthetic views that represent 3D model fragments visible from the their perspective. Furthermore, the registration of the query image is accelerated by GPU-based extraction and matching of SIFT features.

Sattler *et al.* [134] and Li *et al.* [84] further improved the scalability of image-based pose recognition systems and, particularly, the performance and robustness of matching between 2D query image features and 3D features of large point-cloud models. Sattler *et al.* [134] proposed to combine expensive-but-accurate 2D-to-3D with efficient-but-coarse 3D-to-2D correspondence matching in an active tree-based search framework, achieving sub-second registration and rejection times on scenes comparable to Irschara *et al.* [66]. Similarly, Li *et al.* [84] propose a bidirectional matching scheme that scales with 10^5s of images and 10^7s of 3D points, covering several hundreds of landmarks spread throughout the world. In comparison, Irschara *et al.* [66] only processed 10^3s of images and 10^5s 3D points. However, the algorithm of Li *et al.* typically requires several seconds per query.

In several works, Arth *et al.* [6, 2, 3] investigated image-based self-localization on mobile phones. They partition pre-computed 3D point-cloud models into Potentially Visible Sets (PVS). In a first version [6], a single small-FOV query image is employed to retrieve similar reference images using a vocabulary tree of a single PVS, followed by robust pose estimation using SIFT features matched between the query image and the retrieved reference images. On a wide-area indoor dataset, their pose recognition system achieved real-time performance on mobile phones. Building upon this work [6], Arth *et al.* describe how localization accuracy and runtime behavior can be considerably improved by using wide-FOV panoramic query images acquired with a SLAM system [2] and by exploiting mobile phone sensors [3] (see Figure 2.7). In the latter case, sensor values are employed to assign gravity and normal vectors to SIFT feature descriptors, resulting in improved feature culling and descriptor matching performance.

Furthermore, Arth *et al.* [5] investigated pose recognition with respect to globally aligned models which contain reference images only registered with a known 3D location (in contrast to a full 3D reconstruction with reference images registered in 6D). Exploiting epipolar and planar homography constraints between a single query image and the reference images, they can, despite not in real-time, estimate the full 6D pose.

Most recently, SLAM systems have been used for wide-area localization. SLAM does not require an a-priori known model, but is capable of mapping and tracking arbitrary scenes at runtime. However, the local coordinate system of the SLAM map only allows for tracking “relative” poses. Ventura *et al.* [169] presented a localization method to align a local 3D SLAM map with a globally registered 3D model over time, and showed real-time SLAM on a mobile devices with globally registered 6D pose tracking in urban outdoor environments. A similar system was concurrently developed by Middelberg *et al.* [101]. However, before global localization can take place, a local 3D SLAM map needs to be initialized from a stereo image pair, which requires a translational camera motion. In urban outdoor environments, this requires the user to walk several meters to span the required baseline. By contrast, we show in Chapter 5 how to use the first keyframe



(a) (b)

Figure 2.8: MonoSLAM [32] and PTAM [72] point-feature 3D maps when observing the same scene. In comparison to the (a) sparse probabilistic map of MonoSLAM, (b) PTAM reconstructs far more features (black), including some outliers. Additionally, (b) depicts the keyframe trajectory and the initial dominant plane estimated by PTAM. © [72]

acquired by the SLAM system to perform geo-localization and SLAM map initialization.

2.3 Real-time localization and mapping

In the following, we discuss methods for real-time registration in unmodeled environments with a focus on monocular visual Simultaneous Localization and Mapping (SLAM), and, in particular, the Parallel Tracking and Mapping (PTAM) algorithm (Section 2.3.1), which plays a central role throughout this thesis. Furthermore, we review methods which employ constrained camera motion and scene structure assumptions, including planar and panoramic mapping and tracking (Section 2.3.2), which inspired parts of our contributions. Finally, we present unconstrained hybrid SLAM approaches, which aim to allow for arbitrary camera motion and scene structure (Section 2.3.3), particularly related to Chapter 4.

We begin with an overview of important milestones within the rich literature on real-time localization and mapping, including Simultaneous Localization and Mapping (SLAM), Visual Odometry (VO) and real-time Structure from Motion (SfM) methods. We focus on methods which use a single monocular camera, but also briefly discuss methods which employ other input sensors such as RGB-D cameras.

Monocular visual localization and mapping The first SLAM algorithms emerged from the robotics community in the late 1980s, predominantly for the task of autonomous self-localization of mobile robots [149]. These robotics systems, however, were typically using a rich array of sensors, including visual sensors.

The first monocular visual SLAM system has been shown by Davison *et al.* [32, 33]. In contrast to offline and batch SfM methods at the time, they achieved real-time performance on desktop computers by employing an Extended Kalman Filter (EKF) approach. The EKF allows for representing a probabilistic 3D map consisting of the camera pose and a number of real-world 3D landmark features in a state vector (plus covariance matrix). The state is continuously adapted to measurements taken in the incoming camera images in constant time. For each incoming camera image, a predict-measure-update cycle is traversed: first, a constant velocity motion model is used to predict the state (*i.e.*, the 6D camera pose and the 3D feature locations) and its uncertainty, followed by active feature search and matching within the camera image, and, finally, given the new image measurements, a state update is performed and the uncertainty reduced.

Approaching the SLAM problem from the Structure from Motion (SfM) perspective, Nister *et al.* [114] presented the first Visual Odometry (VO) system, which allowed for reconstructing the trajectory of a stereo or monocular camera in real-time. Their system builds upon efficient corner feature detection and template matching, minimal solvers, and robust RANSAC estimation [111] to reconstruct a temporary 3D point feature map and to estimate the camera pose. Initially, the system performs frame-to-frame tracking and estimates the relative camera pose with the five-point algorithm [113] to triangulate a 3D point feature map. The 3D map is used to estimate the absolute camera pose with a three-point pose solver [112] and to triangulate further 3D features. Inserting "firewall" frames into the incoming camera stream, however, the 3D map is intentionally kept small by regularly discarding 3D features and effectively using only frames between the latest "firewall" and the current frame for reconstruction.

Going further in employing SfM techniques in their Parallel Tracking and Mapping (PTAM) work, Klein and Murray [72, 73, 74] presented a keyframe- and optimization-based SLAM system, that, in particular, incorporated the following innovations: the selection of discrete keyframes for mapping, the separation of tracking and mapping on multi-core CPUs, allowing for full camera frame-rate tracking and keyframe-rate mapping, and the usage of bundle adjustment for local and global map optimization. In comparison to filter-based SLAM, PTAM was found more robust and more computationally efficient [152]. Since filter-based SLAM does not scale well with the size of the state vector, it only allows for a small number of (high quality) 3D landmark features (see Figure 2.8). In comparison to Visual Odometry (VO), PTAM was found less affected by drift due to the usage of global map optimization and keyframe-based map tracking. Having these properties, PTAM still represents the gold standard monocular visual SLAM approach on hand-held and mobile hardware platforms. We discuss the PTAM algorithm and its extensions in more detail in Section 2.3.1.

More recently, Newcombe *et al.* [108, 110] presented a Dense Tracking and Mapping (DTAM) algorithm, that, in contrast to the sparse point and edge feature maps of filter- and keyframe-based SLAM, reconstructed dense textured 3D surface mesh models consisting of millions of vertices. Real-time performance on powerful desktop computers is enabled though GPGPU parallelization on high-end commodity graphics hardware.

Most recently, the Large-Scale Direct (LSD) SLAM algorithm of Engel *et al.* [42, 41] builds maps consisting of a pose graph of keyframes associated with semi-dense depth maps, that, in contrast to DTAM, only contain depth values at pixels with non-negligible image gradients. Consequently, their method runs in real-time on the CPU of desktop machines, and – in a VO variant without consistent mapping capabilities – even on mobile phones [139].

All of the monocular visual SLAM algorithms presented above are *general* SLAM algorithms, since they implicitly assume parallax-inducing camera motion. The presence of parallax-free camera motion, however, causes inherent problems, because different kinds of tracking and mapping techniques would be required. This refers us to constrained and unconstrained hybrid SLAM algorithms, which are discussed in Section 2.3.2 and Section 2.3.3, respectively.

Computing platforms Over time, SLAM has been successfully explored in many different configurations, including varying computing platforms and sensor inputs. For example, in addition to the original desktop version of PTAM [72], a heavily modified and stripped-down PTAM version has been shown to run on Apple's iPhone 3GS [74]. On the other end of the computing platform spectrum, SLAM algorithms such as DTAM [108, 110] require a powerful desktop computer with high-end GPGPU support, which is not available on hand-held devices today. Consequently, localization and mapping workload has also been distributed between network-connected computers, such as a mobile device and a powerful server [177] for the purpose of dense 3D model reconstruction.

Most recently, Tanskanen *et al.* [158] and Kolev *et al.* [75] investigated dense metric

3D model reconstruction of small-scale objects using standalone SLAM on mobile phones. The system of Tanskanen *et al.* performs visual-inertial sensor fusion, combining feature-based tracking and mapping similar to PTAM with metric scale and pose estimation using the mobile phones' inertial sensors. Moreover, they compute filtered depth maps using a photometric criterion between a binocular stereo image pair, which takes 2-3 seconds for a 640×480 pixel image. Building upon this framework, Kolev *et al.* later described an improved method for the fusion of multiple depth maps into a consistent 3D model.

Sensors Besides single monocular cameras available on hand-held devices, SLAM algorithms have also employed more powerful sensors such as catadioptric (*e.g.*, fisheye, omnidirectional) cameras [159], stereo rigs [120, 100] and combinations of these [30]. The usage of Light Detection and Ranging (LiDAR) sensors also has a long tradition in SLAM research [78]. More recently, RGB+Depth (RGB-D) cameras based on Time of Flight (ToF) or structured light sensors have become very popular. These sensors work particularly well in indoor environments, but not outdoors and in sunny conditions. Additionally, they have a limited range.

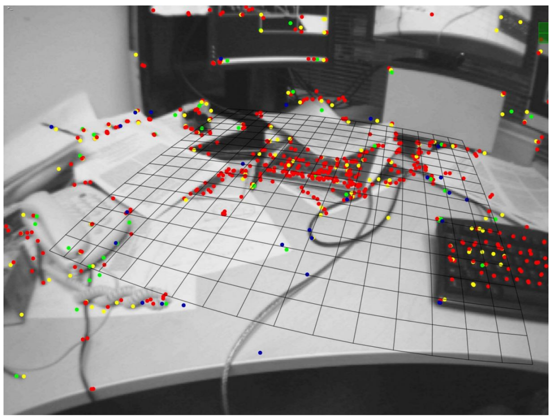
Employing the Microsoft Kinect, a commodity infrared structured light sensor, Newcombe *et al.* [109] describe a real-time dense volumetric surface mapping and tracking method dubbed "KinectFusion". They fuse the sensor depth maps into a global implicit surface model, which is directly used to estimate the sensor pose with an Iterative Closest Point (ICP) algorithm. Both, fusion and ICP, are accelerated with GPGPU, resulting in real-time performance on desktop computers.

In contrast to ICP tracking of KinectFusion, the RGB-D SLAM system of Kerl *et al.* [69] performs probabilistic tracking by both minimizing the photometric error of the RGB image as well as the depth error of the current RGB-D frame with respect to a reference keyframe. The keyframes are organized in a pose graph, where every vertex is a relative pose between two keyframes and has an uncertainty weight. The pose graph allows for loop closure detection and global map refinement with generic optimization frameworks such as g2o [77] or Ceres[4].

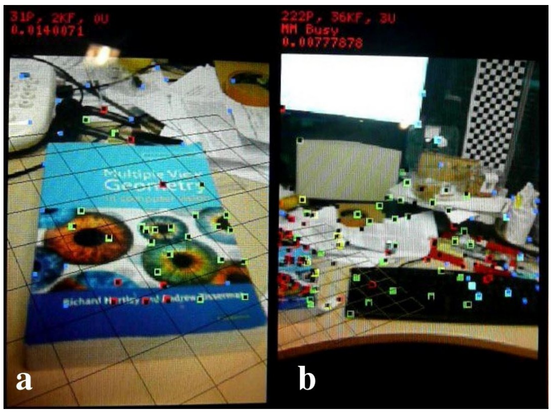
2.3.1 Parallel tracking and mapping

Since all our contributions presented in this thesis are related to the Parallel Tracking and Mapping (PTAM) algorithm, we explain its most important concepts and components,

[4]`http://ceres-solver.org`



(a) © [72]



(b) © [74]

Figure 2.9: User interface of the Parallel Tracking and Mapping (PTAM) system running on (a) the PC [72] and (b) the mobile phone [74]. In comparison, the 3D map of the PC version contains much more point features which can be tracked.

and some of its limitations. Additionally, we review the original design and a selection of extensions. The original method was introduced by Klein and Murray [72], and its robustness and agility were considerably extended in a follow-up work [73]. Furthermore, a heavily modified and stripped-down version has been demonstrated on a mobile phone [74], depicted in Figure 2.9.

The original PTAM algorithm is composed of tightly coupled tracking (a.k.a. localization) and mapping components, which are executed in parallel on different synchronized CPU threads, exchanging keyframes and a 3D model. The tracking component estimates the 6D camera pose of the incoming frames with respect to the given 3D model and discretely selects keyframes from the image stream. Keyframes are employed by the mapping thread to extend and refine the 3D model. Most importantly, the separation of tracking and mapping allows for two diverging processing paces. While the tracking thread runs at real-time camera frame-rate, the mapping thread is not subject to strict real-time constraints as it may run at keyframe-rate. Naturally, the threading concept can and has been extended, *e.g.*, to fully exploit quad-core CPUs. We understand PTAM as a very general framework, that has been interpreted and implemented in many different ways.

Map In general, PTAM allows for arbitrary map layouts, features, representations and parametrizations. Typically, PTAM maps consist of point and edge features, keyframes and corresponding 2D camera image observations, represented in 3D space (6D camera poses, 3D feature locations). Due to the separation of tracking and mapping, PTAM can usually manage maps with a much higher feature density compared to filter-based SLAM (*e.g.*, a few dozens vs. thousands of features). In comparison to the truly dense maps of DTAM or RGB-D SLAM, however, the PTAM maps are still considered sparse. In addition, PTAM has been shown to work with multiple submaps [19]. It is worth noticing, that without performing special computations, each map has an arbitrary and distinct scale.

Initialization Starting from scratch at system startup, SLAM must first initialize its map, typically from a stereo pair of keyframes. The bootstrapping process depends on the map representation. For example, the 2D panoramic SLAM maps can be initialized from a single keyframe. In general, however, the initialization of a 3D map requires parallax-inducing camera motion between two keyframes.

Between selecting a first and a second keyframe, 2D features are tracked and a motion model is estimated using techniques familiar from model-based tracking (minimal solvers plus RANSAC, and iterative refinement using M-estimators). For example, the 5-point algorithm of Nister *et al.* [113] allows for estimating the epipolar geometry (essential/fundamental matrices) and relative 6D poses. Another option is the plane-induced homography motion model, which can be decomposed into a relative pose and the plane equation using homography decomposition methods [45, 96]. Recently, Mulloni *et al.* [106] investigated the issue of stereo initialization in detail, presenting a novel stereo pose estimation method based on bundle adjustment, and also performing a user study. The study revealed that users regularly fail in performing the required parallax-inducing camera motion (*e.g.*, a simple sideways translation or general camera movement similar to an orbit). Instead, users often performed parallax-free pure-rotation movements.

Tracking and relocalization (recovery) After initialization, PTAM performs drift-free 6D camera pose tracking with respect to the given 3D map. Given the prior pose from the previous frame, the current pose is predicted using a motion model, *e.g.*, using second-order minimization [15] to estimate the inter-frame rotation. Consequently, 2D-3D correspondences are matched between the map and the current frame (on multiple image pyramid levels). The predicted pose is used to project 3D map features into the current frame. In the current frame, corners are detected using FAST [129] and Shi-Tomasi [141], which are tentatively matched with projected neighboring 3D map features using SSD scoring. The final matches are found using KLT-based [10] sub-pixel refinement between the current frame and reference keyframes. Alternatively, the template-based patch-matching algorithm of Wagner *et al.* [175] or wide-baseline SIFT matching [156] can be applied. Given the final 3D-2D correspondences, the 6D camera pose is computed with a robust M-estimator.

Despite the remarkable robustness of map tracking, tracking may fail due to effects such as rapid camera motion, motion blur, untextured or unmapped scene regions. For these cases, the original PTAM integrates a recovery component, which performs *immediate* camera relocalization. SLAM relocalization is related to model-based detection in the sense that the camera pose must be estimated from scratch (without having a pose prior) with respect to entire SLAM map. This capability is particularly important when switching between multiple submaps [19]. Furthermore, the classic SLAM "kidnapped robot problem" is related.

In contrast to immediate recovery, Eade and Drummond [40] proposed a fundamentally different approach which they called "*loop closing = recovery*". The term implies that after tracking failure, the system continues tracking, and eventually initializes a new submap, which is added to the overall map. Furthermore, the system continuously attempts to detect loops between submaps. Given that there is sufficient overlap, a loop closure is detected, and two submaps are merged, similarly resulting in recovery as with immediate relocalization. The obvious advantage of this approach is that the system does not stop tracking and mapping. The price is, however, that until a loop between two submaps is found, global map consistency is violated, as each submap has its distinct scale and coordinate system.

Relocalization and loop closure are related topics. Klein and Murray themselves present a simple but effective relocalization method based on small blurry images [73], which are matched with existing keyframes using direct second-order minimization [15]. Furthermore, Williams *et al.* [178] distinguish between map-to-map [28], image-to-image [29], and image-to-map [179] approaches. Map-to-map refers to matching the features two submaps, which reminds us of the estimation of the 7D relationship between two coordinate systems based on 3D-3D correspondences [64]. Image-to-image refers to image-based localization using wide-baseline matching, and content-based image retrieval (a.k.a. object detection). The image-to-map algorithm of Williams *et al.* [179] is based on the Ferns classifier [82] and is applied in the original PTAM.

Keyframe selection One important task of the PTAM tracking component is the selection of keyframes which are passed to the mapping component to extend and refine the

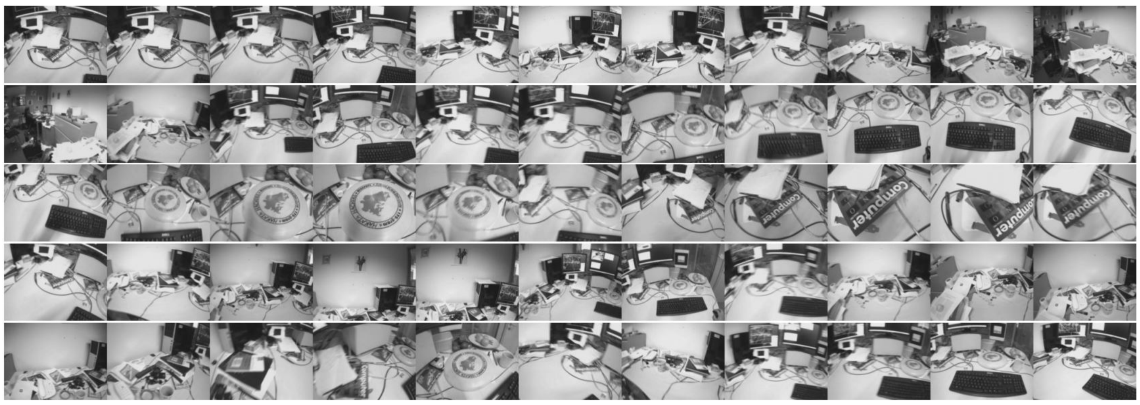


Figure 2.10: Keyframes selected by PTAM on an indoor "small workspace" scene such as depicted in Figure 2.9. © [72]

map (see Figure 2.10). Keyframes should either provide new perspectives onto already mapped scene parts in order to refine the model, or observe previously unmapped scene regions in order to extend the model. Keyframes are the building blocks for mapping, and their selection is a crucial component of the PTAM system. Even more, as keyframes might damage or corrupt the map, *e.g.*, when selecting keyframes which do not induce parallax.

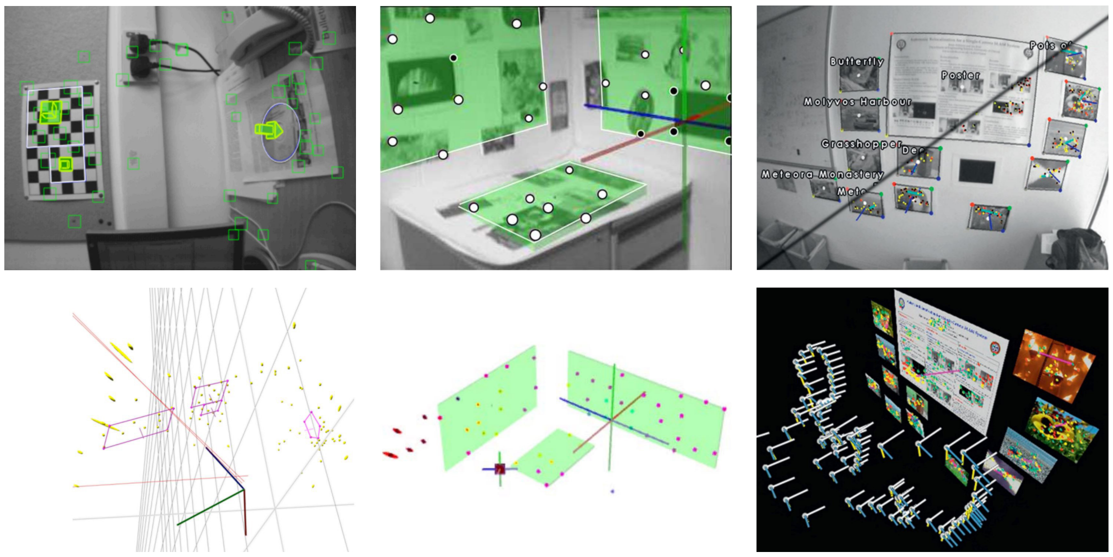
The keyframe selection problem is well known in the SfM literature [121, 160, 127]. One common approach to distinguish between parallax-inducing and parallax-free motion are model selection algorithms such as the Geometric Robust Information Criterion (GRIC) [162]. Due to real-time constraints, however, estimating multiple motion models for each frame is difficult for SLAM, at least on mobile devices. PTAM thus merely employs heuristics based on time, motion parallax, and frame coverage. In order to give the mapping component sufficient time for processing one keyframe, a (small) time offset must be awaited until selecting the next keyframes. Based on map tracking statistics, the motion parallax with respect to existing keyframes can be approximated. Similarly, the coverage of the current frame with map features can be estimated. Recently, Herrera *et al.* [61] presented a PTAM system which tracks both infinite 2D and finite 3D map features, and which allowed for implementing clear criteria for keyframe selection: (1) A given number of new 2D features can be created from areas not covered by the map. (2) A given number of 2D features can be triangulated. (3) A given number of 3D features have been observed from a significantly different angle.

Mapping The mapping component initializes, extends and refines the map based on the selected keyframes. Furthermore, and most importantly, the map is globally and locally optimized with bundle adjustment, providing a the best possible foundation for map tracking.

Given a new keyframe, mapping aims to create new 3D map features using triangulation. Triangulation requires at least two corresponding 2D observations in keyframes which exhibit sufficient motion parallax. Given the keyframe poses, corresponding 2D observations are found with local search along the epipolar line or wide-baseline matching. Similarly, mapping aims to add new keyframe observation to existing map features.

One core component of the PTAM mapping algorithm is bundle adjustment. According to the standard text of Triggs *et al.* [163], bundle adjustment is "the problem of refining a visual reconstruction to produce jointly optimal structure and motion parameter estimates". Optimal means with respect to some cost (objective, error) function, which is typically the reprojection error. Bundle adjustment is usually formulated as non-linear least squares problem, robustified against outliers with M-estimators, and typically carried out with Gauss-Newton or Levenberg-Marquardt approximations. In practice, bundle adjustment requires an approximate (as good as possible) initialization in order to converge to the global optimum. Additionally, it usually has cubic time complexity (*e.g.*, in the number of observations), and, thus, requires considerable processing resources, especially for larger reconstruction problems. In order to make it suitable for real-time applications such as SLAM, the runtime behavior of bundle adjustment has been improved by clever mathematical formulations and sparsifications [43, 68], and has been distributed over multiple CPU cores and GPUs [181]. However, it often remains as *the* computational bottleneck in optimization-based SLAM algorithms.

The runtime behavior and robustness of SLAM highly depends on the map data volume (number of features, keyframes, observations) and on its quality, referring to data association problems of filter-based SLAM. Given sufficient processing resources, the maps can be densified and outlier features somewhat ignored, since they are handled by robust estimators. With limited processing resources, such as on hand-held devices, however, a tradeoff between robustness and real-time constraints must be made, and issues such as map data volume optimization and data association considered. First and foremost, PTAM systems have to carefully select keyframes, measure features, and furthermore detect outlier observations and features as part of their mapping process. Furthermore, the large-scale SLAM reconstruction literature proposes several approaches for dynamically managing the



(a) © [126] **(b)** © [55] **(c)** © [20]

Figure 2.11: Detection of high level SLAM map features, providing scene understanding hooks for AR annotations: (a) polygons and ellipses [126], (b) planes [55], (c) planar object detection [20]. For (a)-(c), AR view (top) the map view (bottom) are depicted.

mapping problem, including employing local/windowed/relative bundle adjustment [143], active windows [105] or pose graph optimization [116, 78, 85, 151]. Essentially, these approaches aim for dynamically keeping the size of the "active" mapping problem small enough to meet the real-time constraint, while maintaining overall map consistency. In particular, loop closure is an important topic, using relocalization techniques presented above.

The estimation and integration of high-level map features provides valuable scene understanding hints for AR applications and the PTAM system itself. As part of map tracking, template matching benefits from a good approximation of the assumed local planar surface of the tracked 3D features. Molton *et al.* [103] employ a gradient-based image alignment method to estimate the surface normals. Aiming to provide a planar "playground" for AR applications, the original PTAM assumes to observe a planar scene during initialization, and places the origin of the coordinate system into the detected dominant (x, y) plane.

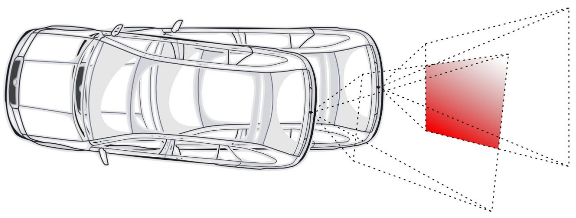
As depicted in Figure 2.11, the detection of further scene planes and planar scene mapping has been investigated in multiple works [126, 55, 170]. Reitmayr *et al.* [126] use a point-based SLAM system to simplify the online authoring of annotations in unknown environments. As part of the SLAM algorithm, they automatically detect and track high-level features (polygons, ellipses) indicated by the user. Gee *et al.* [55] integrated plane detection into filter-based SLAM. When a plane is detected, the corresponding low-level features are replaced with a high-level plane feature representation in the filter state. Based on this system, Chekhlov *et al.* [23] presented an AR game application that required the player to navigate a virtual character through a real-world planar scene by interactively creating a planar SLAM map of the environment, supported by systems' automatic plane detection. Ventura *et al.* [170] integrated RANSAC-based plane fitting into the PTAM mapping component and allow for computing synthesized views that can be used for occlusion rendering.

Integrating object detection techniques in the mapping component, the PTAM system of Castle *et al.* [20] contains a SIFT feature database, which is matched with each keyframe to detect known planar objects in the scene. Known objects are integrated into the map, providing hooks for AR annotations.

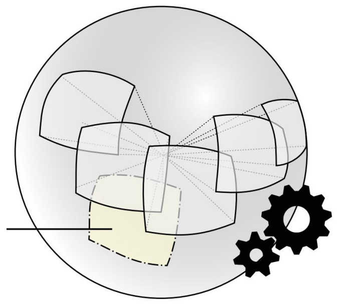
PTAM on the Apple iPhone 3G Adapting for the deficiencies of hand-held computing platforms in comparison to desktop computers, Klein and Murray [74] created a heavily modified PTAM algorithm, that compensated for poor camera (*e.g.*, rolling shutter, narrow field-of-view) and processing (*e.g.*, single-core CPU) capabilities of the Apple iPhone 3G. Since Bundle Adjustment (BA) optimization turned out to be the computational bottleneck, the map size required to be considerably limited by both minimizing the quantity and maximizing the quality of keyframes, point features and observations. In particular, Klein and Murray introduced keyframe culling, *e.g.*, by removing redundant keyframes, as well as measurement and point feature culling, *e.g.*, by employing the BA information matrix to improve the data association quality within the map. Similarly, the number of point features used for map tracking was reduced and template matching only performed on the first level of the image pyramid (having a 160×120px resolution) for efficiency reasons. In summary, while the modified PTAM system was capable of mapping and augmenting small maps, it was considered as "far less accurate and robust" in comparison to the desktop version.

2.3.2 Constrained SLAM

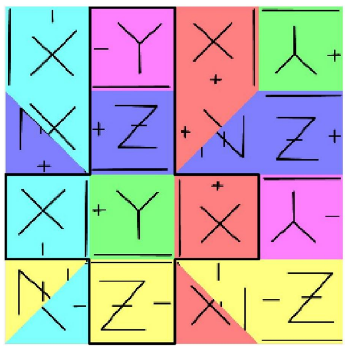
General SLAM and SfM algorithms do not make prior assumptions about the camera motion and scene geometry. Related to our contributions on planar and hybrid SLAM



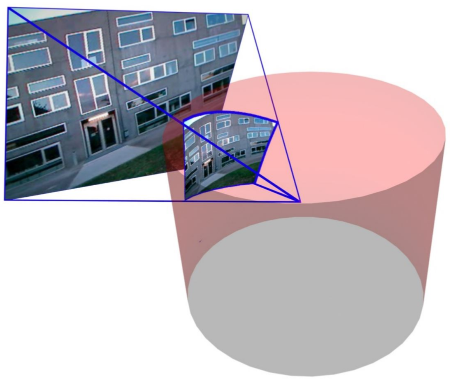
(a) © [91]



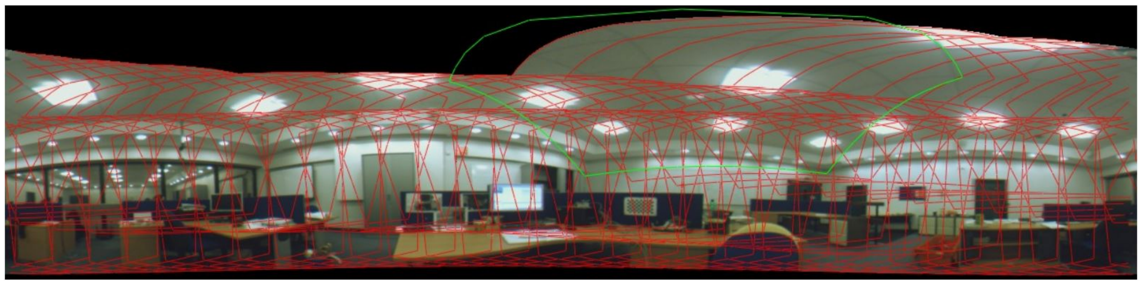
(b) © [90]



(c) © [35]



(d) © [173]



(e) © [90]

Figure 2.12: Constrained SLAM motion models and maps: (a) planar motion model [91], (b) spherical map [90], (c) cube map texture atlas [35], (d) cylindrical mapping [173], (e) spherical panoramic image mosaic [90].

presented in Chapter 3 and Chapter 4, we review a selection of recent mapping and tracking algorithms, which exploit constraints on planar scenes, and constraints on parallax-free camera motion, as shown in Figure 2.12.

2.3.2.1 Planar mapping and tracking

Based on single- or multi-planar scene assumptions, special mapping and tracking techniques can be applied. As already discussed in Section 2.3.1, this includes the automatic detection of real-world planes in 3D feature maps as part of SLAM mapping [55, 170] with the aim to generate high-level scene descriptions.

Similarly, interactive modeling systems [16, 168, 144] are concerned with creating high-level 3D models which are essentially composed of planar 2D polygons manually defined by the user. These in-situ AR systems use the moving camera as interaction device and combine image-based modeling with model-based camera tracking or SLAM.

Fraundorfer *et al.* [51] describe an automatic offline algorithm for the reconstruction of piecewise planar 3D models. In particular, they propose to automatically segment the planar scene regions observed within multiple images by inserting “seed points” which are verified with plane-induced homographies in a RANSAC framework.

Given a single camera image and user-selected planar patch location, the system of Lee *et al.* [80] uses the phones’ accelerometer to approximate the camera orientation and render a fronto-parallel view of the scene (assumed to be vertical or horizontal). Based on this view, the system learns the patch, using a perspective patch recognition algorithm [62]. After the learning stage, the patch can be detected and tracked in further camera images, but no larger 3D model is reconstructed.

Lovegrove *et al.* [91] describe an visual odometry method for a rear-parking camera, rigidly mounted on a moving car. The camera observes the planar road surface. For tracking the self-similar texture of the road surface, their visual odometry method employs a GPU-enabled whole image alignment algorithm based on direct second order minimization [95, 99]. It is shown that the raw GPS-trajectory is considerably improved when being fused with the camera trajectory resulting from the visual odometry tracking.

Our planar SLAM system for mobile devices described in Chapter 3. We solve the planar reconstruction problem of estimating the keyframe poses with an efficient image rectification algorithm and plane-induced homographies.

2.3.2.2 Panoramic mapping and tracking

Panoramic mapping and tracking algorithms assume parallax-free camera motion (also known as panoramic camera motion). Parallax-free camera motion is either the result of the camera rotating around its center (a.k.a. pure-rotation or rotation-only camera motion), or camera translation which is insignificant with respect to large or extreme depth of the observed scene. Both cases allow for panoramic mapping, which is reconstructing features with infinite depth in 2D on the plane at infinity. Map representations include sparse feature maps and dense image mosaics which are mapped onto geometric shapes such as cubes, cylinders and spheres. Panoramic mapping is a well-known technique in offline SfM, pioneered by Szeliski *et al.* Their tutorial [153] elaborates on methods for the creation of seamless panoramic image mosaics. In particular, they discuss direct (pixel-based) and feature-based alignment methods for pairs of images using different motion models, as well as and local and global optimization methods (*e.g.*, parallax removal, loop closure) for a larger number of input images.

Real-time panoramic filter-based SLAM has first been described by Montiel and Davison [104]. Their Extended Kalman Filter-based “visual compass” system assumes a pure-rotation angular velocity motion model and encodes map features as spherical coordinates with infinite depth in its EKF state. This system has later been extended by Civera *et al.* [26] to build up spherical image mosaics in real-time. The sparse point feature map is used to create a triangular mesh on a sphere, which is textured with patches taken from the camera images where the map features have first been observed. The mesh is synchronized with the EKF state over time, incorporating important refinements such as loop closures.

In their “Envisor” system, DiVerdi *et al.* [35] performed real-time panoramic mapping and tracking for the purpose of reconstructing environment maps which represent the local light distribution and allow for shading virtual objects. For the reconstruction of an environment map, the camera video stream is projected onto a cubemap using hardware-accelerated graphics operations. The orientation of the camera pose required for the cubemap projection is computed by frame-to-frame tracking, combined with map feature tracking to avoid drift.

The spherical mosaicing system of Lovegrove and Davison [90] combines direct whole image alignment tracking with global optimization of a spherical map, which consists of a set of keyframes. Orientation tracking employs direct image alignment between the current frame and a reference keyframe. The alignment is based on Efficient Second-order Minimization [95], which is implemented on the GPU to exploit parallelism and to achieve real-time performance on desktop computers. Similarly, global map optimization minimizes a photometric cost function between the set of keyframes.

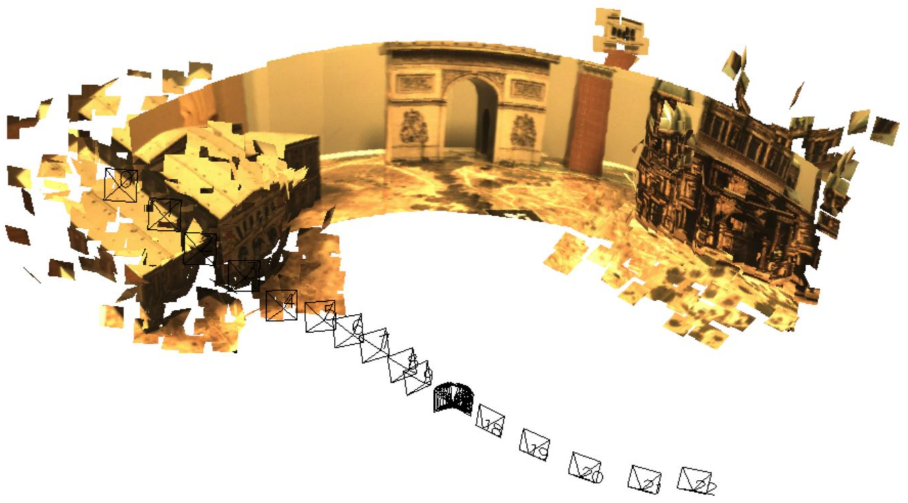
Panoramic mapping and tracking on mobile phones was demonstrated by Wagner *et al.* [173] using a cylindrical map that can be unwrapped into a fixed size 2D image (the vertical range of the map is thus limited). The image itself is split into a regular grid of square cells which are filled with pixels from the incoming camera images according to their orientation with respect to the map. Once pixels of a cell are filled, corner features are detected within the cell, and the features continuously used for camera tracking. The map is initialized from the first camera frame which is projected with the canonical rotation. The system performed with 30Hz on mobile phones.

2.3.3 Unconstrained hybrid SLAM

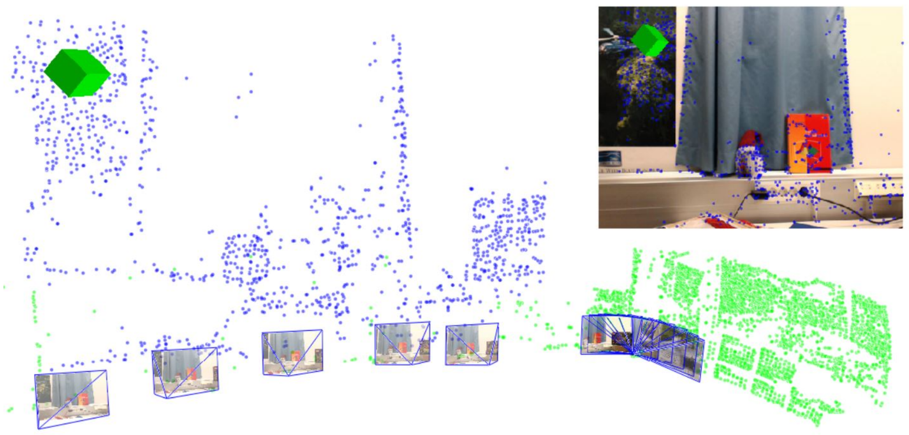
Unconstrained SLAM systems are capable of processing arbitrary camera motion and scene structure. For that purpose, hybrid SLAM systems have to detect and differentiate between parallax-inducing and parallax-free camera motion, and apply appropriate mapping and tracking techniques. This includes the management of hybrid maps consisting of features with both finite and infinite depth, either in a single consistent global map, or in multiple non-consistent submaps.

Civera *et al.* [25] integrated a Bayesian motion model framework and inverse feature depth parameterization [27] into their EKF-based SLAM system. The detection of the current camera motion (stationary, parallax-free, parallax-inducing) allows for adopting feature depth and confidence parameters in the filter such that smooth transitions are possible. Features are restricted to infinite depth until parallax-inducing camera motion is detected. With features becoming finite, cameras become estimated in 6DOF rather than in 3DOF.

Gauglitz *et al.* [53, 54] presented a keyframe-based real-time SLAM system that applies a generalized GRIC model selection algorithm [162] to explicitly distinguish between parallax-inducing and parallax-free motion models. Depending on the availability of a map, a specific motion model pair is estimated in the tracking component: (1) 6D pose and 3D rotation during tracking phases with respect to an available map. (2) homography and epipolar geometry during frame-to-frame tracking phases with respect to the previous keyframe. Depending on the selected model, either panoramic or 3D mapping is performed. With each model selection context switch, a new panorama/3D submap is



(a) © [54]



(b) © [61]

Figure 2.13: Hybrid SLAM feature maps. (a) One panoramic map topologically aligned with two 3D feature maps by the system of Gauglitz *et al*. [54]. (b) The maps of Herrera *et al*. [61] contain features with finite (blue) and infinite depth (green).

created (see Figure 2.13). The system attempts to merge submaps in the background, applying a "loop closing = recovery" strategy.

In Chapter 4, we propose to employ parallax-free keyframes to build up local panorama maps registered in a global consistent 3D map. Thus, our system is able to maintain tracking during rotational camera motions. Additionally, we seek to actively associate panoramic and 3D map data for improved 3D mapping through deferred triangulation. Due to the availability of a global map, our system can perform robust camera tracking and immediate relocalization.

Most recently, Herrera *et al.* [61] described a hybrid SLAM system which manages multiple submaps containing features with both finite and infinite depth (see Figure 2.13). They employ all features for camera tracking, matching them independently of a specific motion model. Based on the distribution of the matched features types (finite/infinite), only one motion model is estimated. In particular, an elegant motion model estimation cost function gives notion about the parallax of the matched features, allowing them to perform deferred triangulation, *i.e.*, triangulation of originally infinite features which exhibit sufficient parallax in later observations.

3 Homography-Based Planar Mapping and Tracking

In this chapter, we propose a light-weight mapping approach that assumes the scene is composed of a single plane only. A planar scene will induce homographies between images of the plane, which relates to panoramic orientation tracking approaches that assume pure camera rotation [173, 90]. In contrast to pure camera rotations, we allow for full 6D camera motion, which is tracked from the planar scene. While panoramic approaches work best with large camera-object distances, *e.g.*, in outdoor scenes, our system is primarily intended for indoor use, where we can find many human-made planar structures such as tables and walls.

Our system is a tool to rapidly map and track a planar surface within the user's reach. Examples include a background to a board or construction game that is created every time from scratch and extended during the game play: a poster, painting or layout drawing on a wall that needs to be refined with further annotations. In addition to registering virtual content, we also envision a combination with detection and tracking of known real-world objects such as game cards in front of a static planar background that is mapped with our system. To reconstruct the planar environment, we adopt an efficient planar image rectification algorithm. Note that image rectification in its common meaning, *e.g.*, aligning stereo image pairs for point matching in a 1D search space [87], is not our primary goal.

Our system builds on an efficient non-linear optimization scheme for planar rectification [131]. The algorithm computes a 2DOF transformation, which relates a canonical world plane with a single reference camera pose, while optimizing a constraint on all keyframe camera poses (see Section 3.1). This optimization scales linearly in the number of inter-frame homographies, providing a more efficient method to estimate the camera poses than bundle adjustment. Traditional homography decomposition [45, 96] performs a

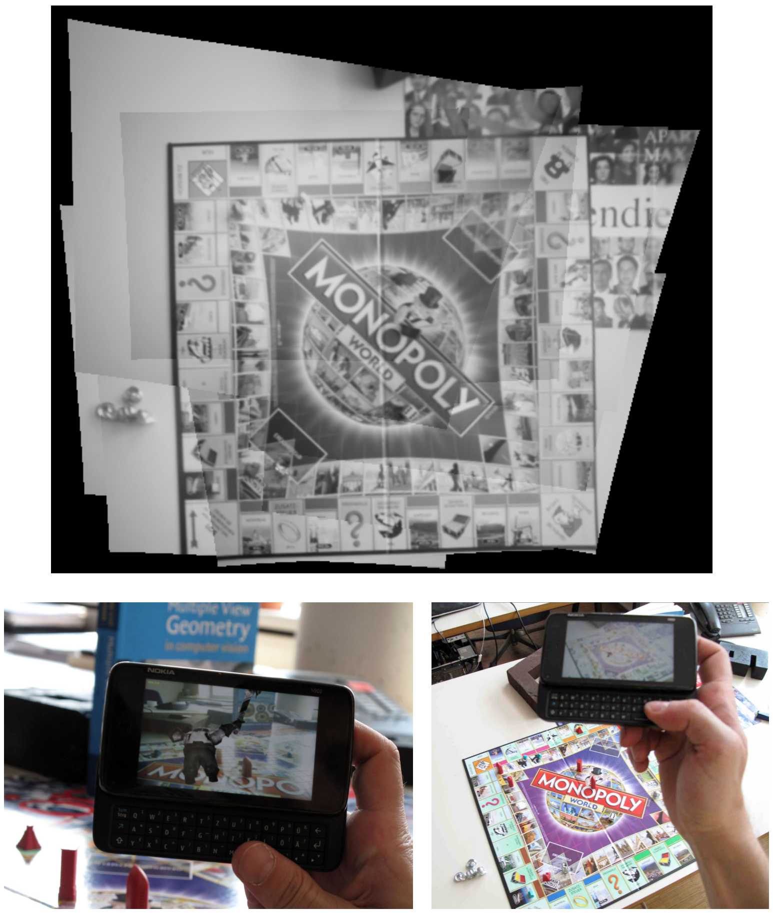


Figure 3.1: Orthoimage sampled from keyframe views of a planar indoor scene and created in real-time on the mobile phone.

similar task for pairs of cameras only and, therefore, does not consider all other keyframes.

Knowing the 2DOF plane-camera transformation, the remaining camera poses can be computed from the inter-frame homographies. Once all keyframe poses are known, we can create a planar map for tracking by back-projecting salient points in the images onto the canonical plane (see Section 3.2). The tracking component matches the map points with the current image and employs the resulting point correspondences in a robust 6-DOF refinement algorithm yielding the camera pose for the current frame (see Section 4.2).

The contribution of this chapter is a planar mapping and tracking system that operates essentially in linear time in the number of inter-keyframe homographies measured. We complement the original image rectification with a simple parameterization and explore various properties of the optimization problem. Furthermore, we developed and evaluated different approaches to robustly apply the optimization to a network of reference frames

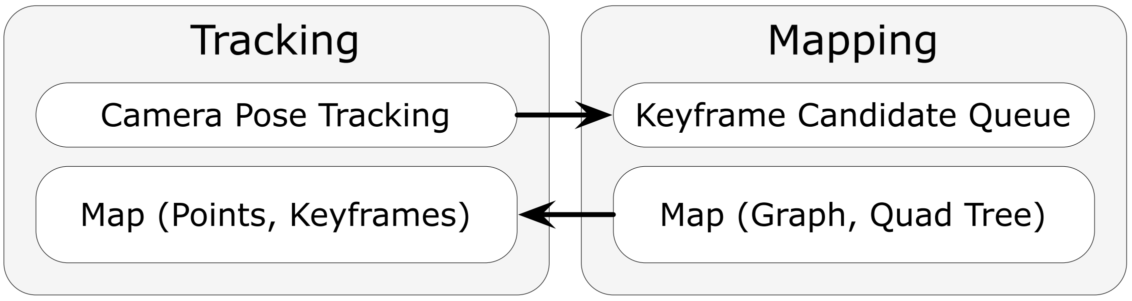


Figure 3.2: System Components. The pose tracking component pushes keyframes onto the mapping candidate queue. The mapping component processes keyframes and delivers enlarged and refined maps to ensure tracking.

connected with redundant measurements. We evaluate our system against PTAM and standard bundle adjustment to explore performance and accuracy of our approach (see Section 5.5). Finally, we demonstrate real-time operation on a mobile device (see Section 3.5).

3.1 Planar Mapping and Tracking

Our system is composed of tightly coupled tracking and mapping components (see Figure 3.2) similar to PTAM [72]. The tracking component processes the image stream of a calibrated monocular camera, computes 6DOF camera poses and selects keyframes, which are passed to the mapping component.

The mapping component establishes a persistent map representation of an unknown textured planar scene, which is continuously extended and refined, as new views become available (see Figure 3.3). Keyframes show diverse views of the 2D scene and are used to compute plane-induced homographies, mapping pixels from one keyframe to another. The resulting relations are managed in a pose graph. Maps are composed of 3D points, which sample the planar surface. Point observations originate either from salient keyframe corners or from feature matching as part of pose or homography estimation. We organize our planar map points in a quad tree resulting in fast point retrieval employed, *e.g.*, during search for unpopulated map regions on different scale levels.

3.1.1 Pose Graph

The main inputs to our mapping approach are a set of keyframes and the homographies measured between them. The camera pose corresponding to keyframe I_i is represented as

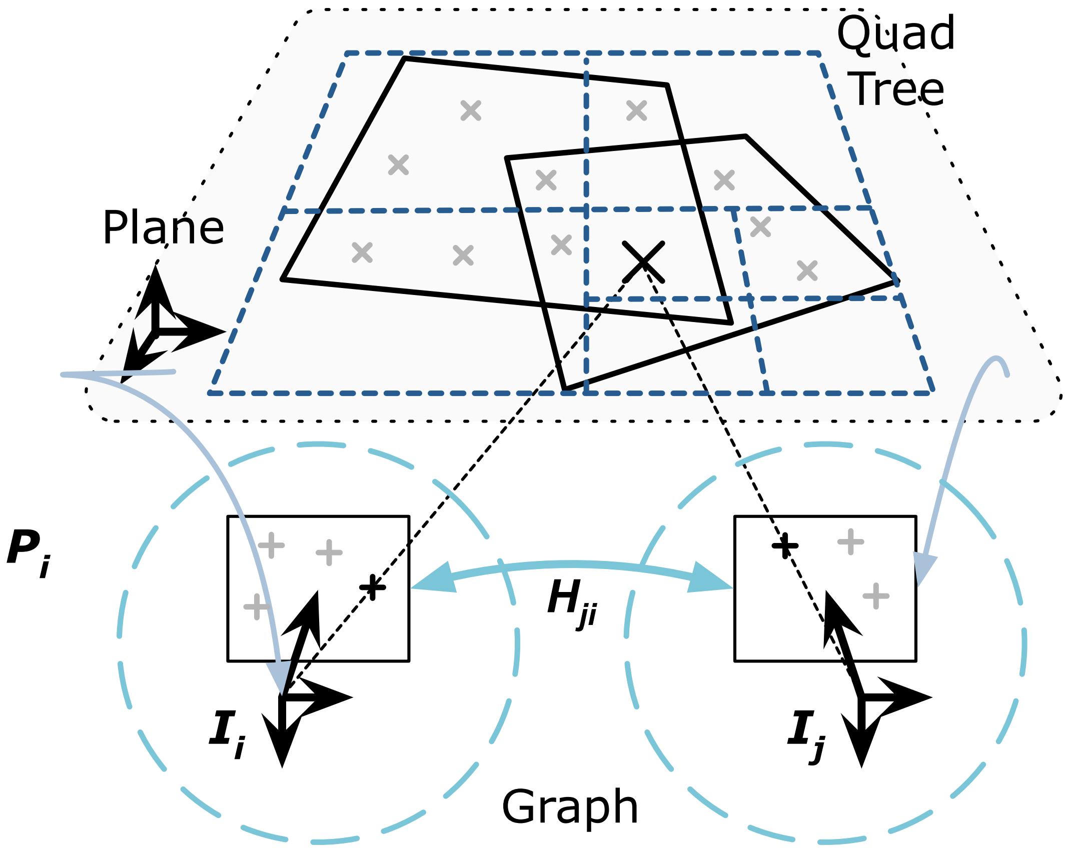


Figure 3.3: Planar map consisting of points organized in a quad tree. Keyframes provide salient image measurements and are related with the plane via camera poses P. Homographies H project points between keyframes I. Both are stored as nodes and edges in a graph.

a 3×4 transformation matrix $P_i = (R_i|t_i)$ representing the rigid transformation from the world coordinate system into the local camera coordinate system for subsequent projection. For some keyframe pairs (i, j), a homography $H_{j,i}$ is measured that maps points from keyframe I_i to keyframe I_j.

The keyframe-homography relations are managed in a directed graph, having keyframes as nodes and homographies as edges. For each homography $H_{j,i}$ estimate connecting a keyframe pair (i, j), we add a directed edge which connects the corresponding nodes I_i and I_j. Our graph is directed, since we estimate homographies between keyframe pairs in both directions to obtain independent measurements.

As new keyframes are generated, new homography pairs to neighboring keyframes are measured and added to the graph. To preserve our planar map assumption, we need to

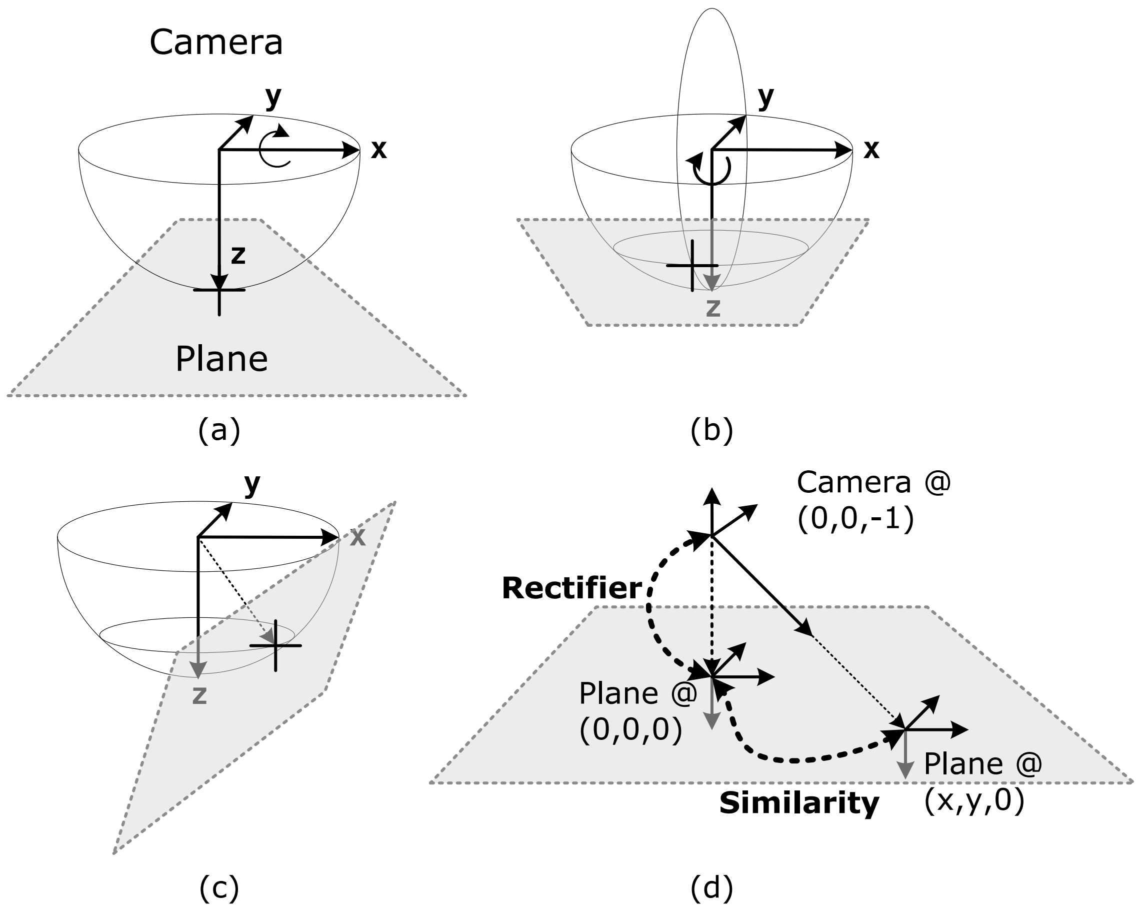


Figure 3.4: Geometric interpretation of the planar image rectification parameterization. The plane rotates on a unit half-sphere around the reference camera. The camera remains in canonical position after the rectifying transformation consisting of x- and z-axis rotations respectively. The final plane coordinate system is defined by intersecting the camera z-ray with the plane resulting in a similarity transformation.

ensure to only include keyframes that show the same plane. Most importantly, we aim to filter keyframes which predominantly show views of outlier regions (*e.g.*, regions depicting a different plane or a non-planar object), which would induce homographies that are not consistent with the main scene plane. Therefore, we employ a consistency check on new keyframes (see Section 3.2.2). Candidate keyframes which fail in this check are rejected. Candidates passing the consistency check are accepted and added to the graph.

3.1.2 Plane Estimation

To create a map of 3D points, we need to estimate the unknown camera poses P_i corresponding to the keyframes. For the calculation of unknown keyframe camera poses with respect to the plane we employ a rectification algorithm [131].

We found two general approaches for solving the problem of estimating the plane using known homographies between images. Homography decomposition refers to computing the camera motion, given a homography matrix between two images of a planar object [45, 96]. Image rectification algorithms compute projective transformations from known epipolar geometry (or homographies, in the planar case), which are applied to image pairs, aligning their epipolar lines, *e.g.*, for the purpose of point matching in a 1D search space [87].

In that spirit, Ruiz *et al.* [131] propose a computationally very efficient non-linear optimization scheme with only 2DOF, which uses a variable number of homography measurements between a dedicated reference camera and other cameras.

3.1.2.1 Cost Function and Parameterization

In the following, we describe the mathematical formulation of the optimization scheme given by Ruiz *et al.* [131] for completeness. We define the scene plane to be located in the canonical position $z = 0$, corresponding to the (x, y) plane. Thus, points on the plane have a z-coordinate equal zero and can be written as $(x, y, 0, 1)$ in homogeneous coordinates. The unknowns in the optimization are the camera poses P_i relative to this plane. Under the assumption that all world points are located on the plane, camera poses can easily be re-formulated as 2D homographies by eliminating the third column of the pose matrix P_i:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim (R|t) \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} = (r_1|r_2|t) \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{3.1}$$

The resulting pose homographies have the following important property based on the observation that their first and second columns are ortho-normal vectors, where r_1 and r_2 are the first and second column of R respectively:

$$C^T \cdot C = \begin{pmatrix} r_1^T \\ r_2^T \\ t^T \end{pmatrix} (r_1|r_2|t) = \begin{pmatrix} 1 & 0 & \cdot \\ 0 & 1 & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix} \tag{3.2}$$

Additionally, given a pose homography C_1 and the homography $H_{2,1}$ mapping from camera C_1 to C_2, the corresponding pose homography C_2 can be computed as follows:

$$C_2 = H_{2,1} \cdot C_1. \tag{3.3}$$

C_1 must observe the constraint (3.2). Moreover, by substituting (3.3) into (3.2) we obtain the following additional constraint for C_1:

$$C_2^T \cdot C_2 = (C_1^T H_{21}^T) \cdot (H_{21} C_1) = \begin{pmatrix} 1 & 0 & \cdot \\ 0 & 1 & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}. \tag{3.4}$$

We can formulate the constraint as a cost function on C_1 by enforcing that the off-diagonal entries are 0 and the diagonal entries have the same value. Thus, we define the following cost function for one homography $H_{i,1}$:

$$(H_{i,1}C_1)^T(H_{i,1}C_1) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdot \\ a_{1,2} & a_{2,2} & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}, \tag{3.5}$$

$$e_i(C_1) = (a_{1,2}/a_{1,1})^2 + (a_{2,2}/a_{1,1} - 1)^2. \tag{3.6}$$

The resulting cost function (3.6) exploits well-known orthogonality constraints over the image of the absolute conic [58] and holds for any homography $H_{i,1}$ mapping from the reference camera to another camera i. For a set of cameras C_i, all connected with individual homographies $H_{i,1}$ to a reference camera C_1, we construct a cost function by adding up individual costs, obtaining a single cost function for the unknown reference camera pose C_1

$$e(C_1) = \sum_i e_i(C_1). \tag{3.7}$$

Overall, the whole problem of estimating all camera poses C_i can be reduced to finding one camera pose C_1 that minimizes the total cost function (3.7).

A homography $H_{2,1}$ between two cameras has eight degrees of freedom, because it is defined up to scale. By fixing the unknown plane and allowing the second camera C_2 to move freely, the first camera C_1 has only two degrees of freedom left. Ruiz *et al.* [131] propose to fix the camera position and vary the camera tilt (x-axis) and roll (z-axis)

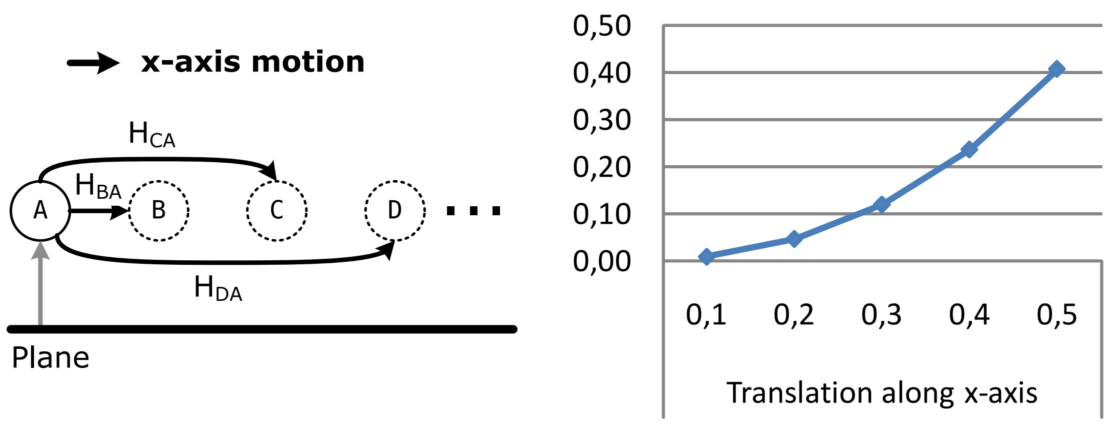


Figure 3.5: Motion encoded in single homographies and rectification cost surface depths. Translational motion along a single axis results in quadratically increasing depth values.

angles, but remain vague concerning the valid 2DOF parameter range. Geometrically, we interpret the parameterization as depicted in Figure 3.4. Plane and reference camera are defined to be located in canonical position, the plane aligning with the world (x, y) plane and the reference camera located at position $(0, 0, -1)$ such that world and camera coordinate systems align. We assume that the plane rotates and the camera stays fixed. The first rotation around the x-axis lets the plane move along a circle aligned with the (y,z) camera plane. The second rotation lets the plane move along another circle aligned with the (x, y) camera plane. Avoiding the plane to be rotated behind the camera, we define $(-\pi/2, \pi/2)$ as range for the x-rotation parameter. For the z-rotation parameter we define $[-\pi/2, \pi/2)$ as the valid range to avoid solution symmetry.

3.1.2.2 Properties

We analyzed the cost function (3.7) and the shape and depth of the resulting 2DOF cost surface. Depending on the selection of reference camera and motion between keyframes, we found various effects, which are described in the following.

We found the motion encoded in the input homographies to considerably influence the depth of the cost surface. The experiment illustrated in Figure 3.5 shows the cost range of the error function, when passing a single homography estimated between images of a fixed camera C_1 and camera C_2 gradually moving along the x-axis. The cost surface depth range increases quadratically with the magnitude of the translation.

In turn, cost depth and motion influence the solution multiplicity of the error function.

Generally, we aim to find the global minimum of the cost surface, which yields a unique 2DOF solution. However, the cost surface regularly shows two local minima, which correspond to solutions describing the camera pose in front and behind the plane. Increasing the amount of motion encoded in the homographies lets the local minima vanish in favor of the correct global minimum describing the camera pose in front of the plane.

We encountered degenerate motion cases, which do not change the cost function, and, thus, do not yield a valid solution. Such cases include pure rotation and translation along the plane normal, which are already described as degenerate in the homography decomposition literature [45]. For homographies encoding such motion, the cost function (3.6) yields values equal zero for all input parameters.

3.2 Homography-Based Planar Mapping

The mapping component processes keyframe candidates selected by the tracking component. Candidates provide new information about the planar scene, which is used to improve the planar reconstruction, and results in refined and extended maps.

3.2.1 Connecting Keyframes

For each keyframe candidate, we select a set of adjacent keyframe nodes from the graph with respect to the subsequent homography estimation. For that purpose, we compute pairwise frame overlap.

The overlap of a source/target keyframe pair is computed by projecting the frame corners of the source keyframe onto the target keyframe with a homography. The homography is derived from the known keyframe poses. The resulting four-point polygons are intersected and unified resulting in another two polygons. The desired overlap is the ratio $r(A, B)$ of these polygon areas:

$$r(A, B) = \frac{\text{area}(A \cap B)}{\text{area}(A \cup B)}. \tag{3.8}$$

A given candidate keyframe is paired with each of the existing keyframes in the graph, resulting in a corresponding list of overlap ratios. The actual selection is done by sorting this list in descending order and retrieving a limited set of keyframes (*e.g.*, five) from the front of the list, resulting in the adjacent keyframe set.

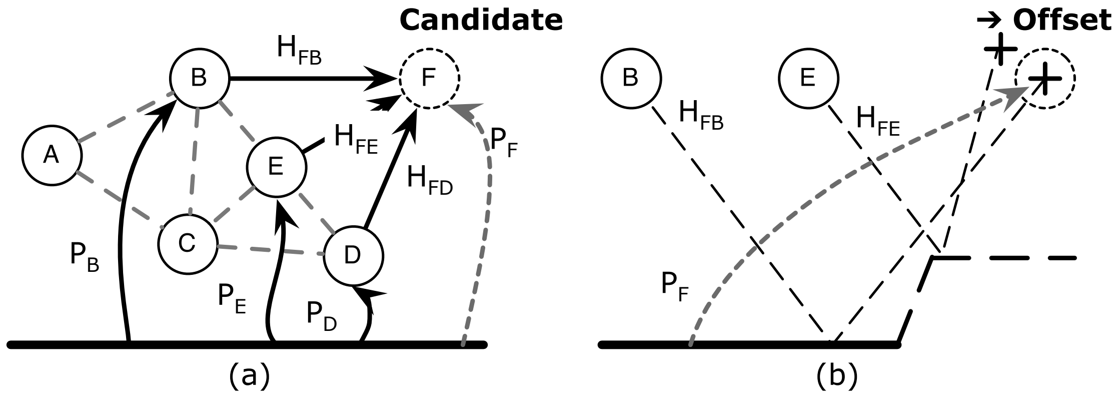


Figure 3.6: Pose consistency check of candidate keyframe F: Tracker pose P_F is compared with multiple homography-pose observations. Observations are computed by multiplying homography estimates between candidate and adjacent keyframes with adjacent keyframe poses (*e.g.*, homography H_{FB} with pose P_B). Position offsets between tracker pose and homography-pose observations result from homographies which are induced by different planes or outlier objects shown by keyframe images.

3.2.2 Pose Consistency Check

Combining the candidate keyframe with the previously selected adjacent keyframe set, we pairwise estimate homographies in both directions. We employ a RANSAC algorithm for robust estimation. Resulting homographies may be induced by planes which conflict with our currently mapped plane. Estimation errors might occur, *e.g.*, from poor frame overlap, uneven feature distribution, or low correspondence count (due to high noise or outliers).

We aim to detect candidate keyframes which feature erroneous homographies by comparing the tracker pose provided by the candidate with pose observations computed from the adjacent keyframes poses and estimated homographies (see Figure 3.6). For each adjacent keyframe I_i, we compute the pose O_i by combining the pose with the corresponding homography and measure the position difference to the candidate camera pose position P obtained from the planar tracker:

$$err = \sqrt{\sum_i \|O_i - P\|^2}\,. \tag{3.9}$$

If the resulting RMS error is below a certain threshold, the candidate and the estimated homographies are inserted in the graph as node and edges respectively. Otherwise, the candidate is rejected.

3.2.3 Reference Keyframe Selection

The selection of the reference keyframe has a considerable impact on the overall quality of the reconstruction, since we compute all keyframes poses with respect to the reference.

We propose two reference keyframe approaches based on observations of different camera trajectory types. As depicted in Figure 3.7, we consider exploration and orbit trajectory types which yield very different graph patterns and suggest different reference selections. For the exploration type, we assume the reference near the frontal (latest) keyframe as superior compared to the orbit type, which suggests the selection of the graph center.

For both reference types, we have implemented piecewise linear scoring functions, which are parameterized with values retrieved from breath-first graph operations. For the central reference node approach, we search the keyframe node with the minimum overall path depth. For the leading reference node approach, we search the graph node with the minimum distance from the latest keyframe minus some variable offset (*e.g.*, two). The evaluation of the scoring functions for each keyframe requires the knowledge of the minimum overall path depth and the minimum path depth to the latest keyframe.

3.2.4 Reference Pose Estimation

After the selection of a reference keyframe, we initialize the plane estimation algorithm with a set of input homographies and, subsequently, recalculate all keyframe poses with respect to the updated camera pose of the reference.

We propose two input homography retrieval approaches as depicted in Figure 3.8. The

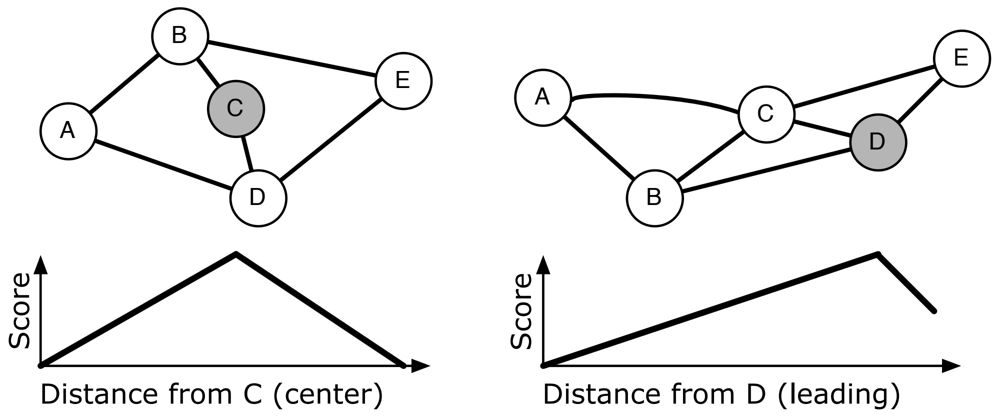
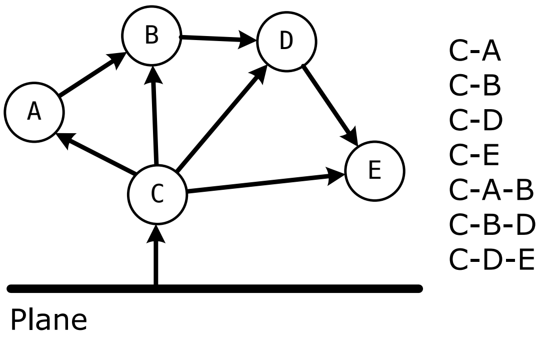
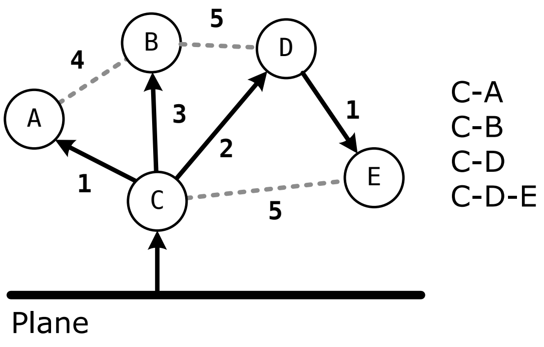
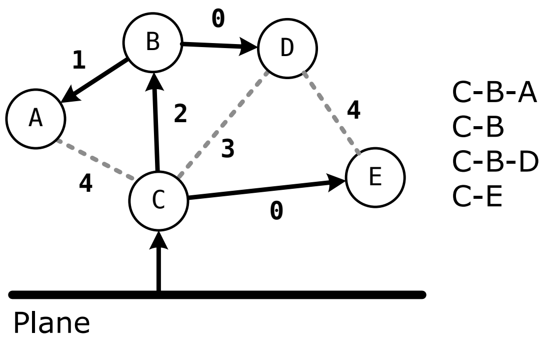


Figure 3.7: Reference keyframe selection using linear scoring functions: central graph node (left) and leading graph node near the latest keyframe (right).



(a)





(b) (c)

Figure 3.8: Homography retrieval approaches yielding different path sets given the same reference keyframe C: (a) Computing paths which contain all graph edges aims for redundant motion between keyframes. (b, c) Computing shortest paths along edges weighted with cost depth values from the rectification algorithm aims for (b) minimal or (c) maximal (using inverted weights) motion between keyframes.

first approach aims for redundancy to mitigate the risk that single erroneous homographies corrupt the rectification algorithm cost surface. For a given root node, we retrieve paths to all other nodes which contain all edges (in either direction) of the graph. Incorporating a maximum number of edges assures a high amount of motion and provides multiple paths to target nodes. Thus, we aim to compensate for composed homographies which contain individual erroneous homographies. The second approach retrieves the shortest paths from root to target nodes using the Dijkstra algorithm. Assigning uniform weights to the edges results in paths with minimum edge counts. Alternatively, we may assign weights which reflect the motion of individual homographies to the corresponding edges. These weights are calculated with the rectification cost function (see Figure 3.5). In the latter case, we aim to compute paths which maximize the motion between reference and other keyframes to achieve rectification cost surfaces which feature unique minima which in turn should result in reliable reference keyframe poses. Since the Dijkstra algorithm delivers the path with the minimum weight sum, we invert the edge weights by subtracting them with the overall maximum edge weight.

We obtain the pose of the reference keyframe by minimizing the cost function (3.7) with the homographies corresponding to the retrieved paths. For a given set of homographies we compute the least-cost global minimum solution within the valid parameter ranges (see Figure 3.9). We start by sampling the cost surface discretely (*e.g.*, each $10°$ in each parameter dimension) and compute a set of sample minima by comparing sampled cost values with their eight-neighbors. From the set of sample minima, we compute a set of local minima. Each sample minimum is refined with a Nelder-Mead downhill simplex solver to find the optimized parameters. Resulting local minima which feature parameters outside of the valid ranges are discarded to avoid symmetric solutions. Furthermore, we compare local minima pairwise and discard duplicates. From the remaining local minima, we select the least-cost global minimum as final solution. From the resulting 2DOF rotation parameters (x, z), we compute the reference keyframe pose by multiplying the two corresponding rotation matrices:

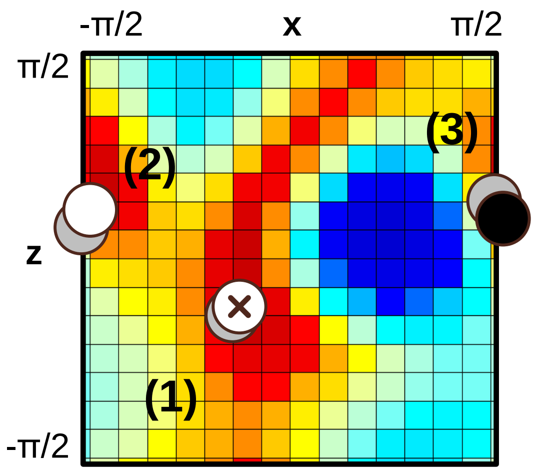
$$C_r = R_z R_x. \tag{3.10}$$

3.2.5 Map Creation

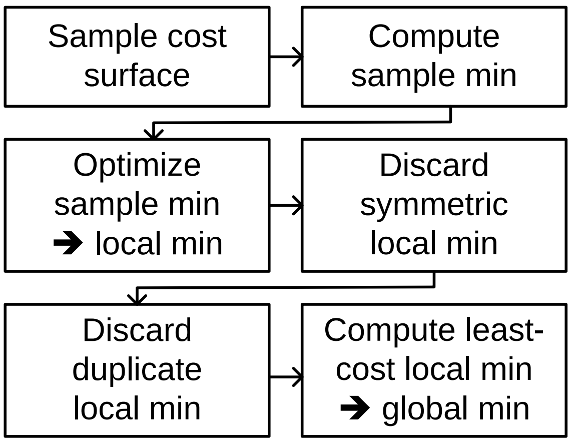
At this point of the reconstruction, we know the pose of the reference keyframe with respect to the plane. The remaining keyframe poses are computed by multiplying the reference pose with inter-keyframe homographies. The map creation is completed by recalculating the 3D positions of map points.

3.2.5.1 Keyframe Pose Update

Keyframe poses P_k are computed by multiplying the reference pose homography C_r resulting from the plane estimation algorithm with a homography $H_{k,r}$ to obtain the keyframe pose homography C_k. The full pose P_k is recovered through calculating the 3rd column of the rotation matrix. Additionally, we apply a similarity transformation S, which moves the plane coordinate system origin into the back-projected principle point of the first keyframe:



(a)


Sample cost surface
Compute sample min
Optimize sample min ➔ local min
Discard symmetric local min
Discard duplicate local min
Compute least-cost local min ➔ global min

(b)

Figure 3.9: Plane estimation algorithm (a) cost surface example and (b) outline for computing the global minimum. This cost surface features three sample minima (gray circles) and two valid local minima (white). Local minimum (3) is discarded, since it is located outside the valid parameter range. The least-cost global minimum (1) is selected as solution (crossed white).

$$C_k = H_{k,r} C_r S. \tag{3.11}$$

For the retrieval of the homography $H_{k,r}$ we compute the shortest path from reference to keyframe using the Dijkstra algorithm. Similar to the homography retrieval operation described in Section 3.2.4 and as depicted in Figure 3.8(c), we assign weights which reflect the motion of individual homographies to the graph edges. Here, we aim to minimize the motion encoded in the resulting path, since we assume that the corresponding homography is less error prone. Alternatively, uniform weights can be assigned to edges resulting in the path with minimum edge count.

3.2.5.2 Map Point Update

We recalculate the positions of map points on the canonical plane using the refined keyframe poses. Each map point is associated with a set of keyframe image observations. We determine the position of each map point by projecting its 2D image observations onto the plane using the updated keyframe poses and computing the centroid of the resulting 3D point observations (having $z = 0$ coordinates). Gross outlier observations are detected by calculating the mean distance of the observations to the centroid and discarding observations which are outside a certain threshold (*e.g.*, two times the mean distance). In such cases, the centroid is computed again using the inlier observations only.

3.3 Planar Tracking

The tracking component processes the incoming image stream from a calibrated monocular camera and computes 6DOF camera poses relative to the planar point map provided by the mapping component. At system startup, the map does not yet exist and is initialized from two keyframes. Subsequently, we estimate camera poses using a motion model and a robust pose refiner. Keyframe candidates are selected and passed to the mapping component, resulting in extended and refined maps.

3.3.1 Initialization

The mapping and tracking system is initialized by the user, who selects a first keyframe manually. Continuously, we estimate homographies between the keyframe and the current input frame. Additionally, we pass the single homography to the plane estimation algorithm (see Section 3.2) and calculate pose estimates for the first keyframe and the current frame.

3.3.1.1 Homography Estimation

Keyframes feature a fixed set of salient image measurements on different image scale levels. We generate a low-pass image pyramid with three levels to improve scale invariance. On each pyramid level, we apply a FAST-10 corner detector [129], generating a set of corner measurements. We additionally apply a measurement filter on the corner set, which distributes the corners into a fixed size (*e.g.*, 40x40) image grid and selects the corners having high FAST scores up to a maximum number of features (*e.g.*, 100/50/25 on the three image levels). Thus, we partially compensate for non-uniform texturedness within the keyframe and restrict the subsequent computational cost.

We use a robust RANSAC estimation algorithm for homography estimation from 2D-2D point correspondences between source and target keyframes. For each frame level (going from coarsest to finest), we iterate the salient image measurements provided by the source keyframe and perform point matching in the target keyframe. After point matching on the level image, the homography is refined with all available correspondences. The RANSAC outlier threshold is set in relation to the known internal camera parameters.

Point correspondences are computed with a sub-pixel accurate affine patch matching algorithm by active search over a search window with normalized cross correlation [186]. Given two frames and an initial homography, we establish a local 1x1 pixel coordinate system at the source point location in the source frame. We generate a local affine warping matrix by projecting the local coordinate system into the target frame. After adjusting the source frame level, we sample an 8x8 patch from the source frame which is correlated with the target frame over a search window with given radius (*e.g.*, 3-5 pixels) using NCC as the error measure. If the NCC value of a target image point is above a certain threshold (*e.g.*, > 0.9), source and target points are considered correlated.

3.3.1.2 Initial Map

With the automatic selection of a second keyframe, the mapping component computes an initial map from the resulting keyframe pair and corresponding homography. The second keyframe is selected based on frame overlap and the plane estimate quality. If frame overlap between first keyframe and current frame exceeds a certain threshold (*e.g.*, 0.75) and the rectification algorithm yields a valid and non-ambiguous plane estimate using the first keyframe as reference, we select the second keyframe and initialize a map.

Map initialization comprises the estimation of the homography from second to first keyframe, the pose estimation for both keyframes and the population of the map with points projected from the keyframes. Planar map point positions are defined by the centroid of the back-projected associated image observations. The world coordinate system origin is defined by back-projecting the principle point of the first keyframe image. Additionally, we setup the initial graph by adding two nodes and two edges.

3.3.2 Map Tracking

After the creation of the initial planar map, the tracking component estimates camera poses using 3D map points and matching them with the current frame. Additionally, keyframes are selected from the image stream and passed to the mapping component.

3.3.2.1 Pose Estimation

The tracking component robustly estimates a 6DOF camera pose relative to the planar map provided by the mapping component using the methods described by Wagner *et al.* [175].

A constant decaying motion model predicts the pose of the current frame based on previous pose observations. The motion model pose is used to initialize a camera frustum, which allows for discarding map points invisible to the current frame, achieving a considerable performance gain especially on large and densely populated maps. The resulting visible point set is filtered to obtain a fixed-sized (*e.g.*, 100/50/25 for the three image levels) uniformly distributed salient point set over the current frame image.

Point matching is consecutively executed for each image level, starting at the coarsest level. After completing a level, we add the matched level correspondences and perform a pose refinement step. The employed pose refinement algorithm aims to minimize the re-projection error using a robust Tukey M-estimator to discard outliers.

After completing point matching on all levels the validity and quality of the resulting pose is examined. Pose quality is defined by the ratio of salient vs. inlier feature counts. If the ratio is above a certain threshold (*e.g.*, > 0.75) it is considered *good*, if it is below a certain threshold (*e.g.*, < 0.5) it is considered *bad*, otherwise it is considered *medium.* If the pose estimator indicates an invalid pose (*e.g.*, because of too little point correspondences), we switch into the re-localization mode.

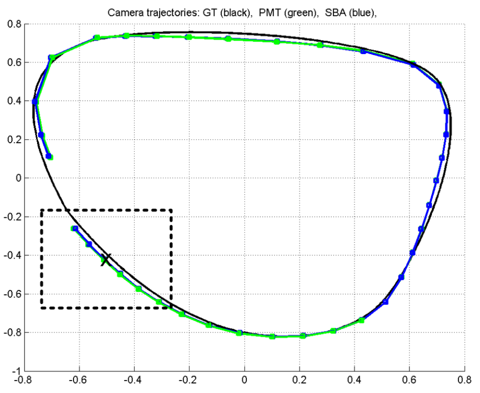
3.3.2.2 Relocalization

We provide a very basic camera pose re-localization approach based on a cache of sub-sampled keyframes images (*e.g.*, with 40x30 resolution). In case the camera pose becomes invalid, the current frame image is matched with the sub-sampled keyframe images from the cache using normalized cross correlation. If the NCC score is above a certain threshold, we initialize the motion model with the associated keyframe pose and try to track the pose.

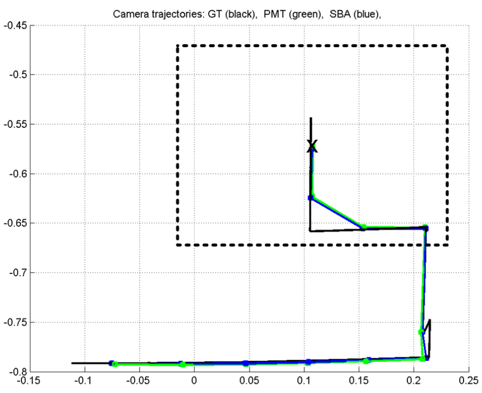
3.3.2.3 Keyframe Selection and Map Update

We select keyframe candidates based on current pose quality and frame overlap from the image stream. Only frames which yield pose estimates with *good* quality are considered as keyframes. If we find no existing keyframe (and candidate) which overlaps the current frame sufficiently (*e.g.*, the maximum overlap ratio is below 0.75), the frame is pushed onto the keyframe candidate queue processed by the mapping component.

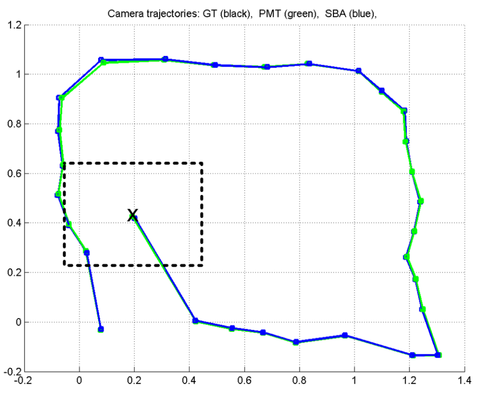
The mapping component indicates the completion of a refined map by setting a corresponding flag. Since refined maps might change considerably (*e.g.*, due to a change in the reference keyframe), we store the correspondences from the last two frames and



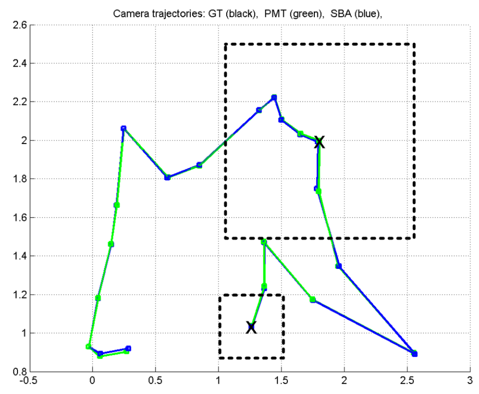
(a)



(b)



(c)



(d)

Figure 3.10: Camera trajectories of test image sequences. The sequences in detail: (a) exploration (synthesized motion, 500 frames, with GT), (b) robot arm (programmed motion, 500 frames, with GT), (c) closed loop (handheld motion, 1200 frames, no GT), (d) scale and rotation (handheld motion, 1400 frames, no GT). Scale is illustrated by dotted frames (size of selected images on the ground plane).

re-estimate their poses with respect to the new map. We update the motion model from these two new poses.

3.4 Results

We compare the proposed system and method with well-known bundle adjustment and 3D SLAM system implementations. We use synthesized and handheld camera image sequences to compare accuracy and performance observations. The evaluations have been performed on a standard PC equipped with a dual-core 2.66 GHz CPU and images with a resolution of 640×480 pixels.

Our test image sequences show well-textured planar indoor scenes and feature no gross outliers. The corresponding camera trajectories simulate motion patterns ranging from explorations to orbits (see Figure 3.10). Loop-closing exploration trajectories feature mainly translational motion, while orbit trajectories provide rich rotational motion and large differences in scale. Sequences comprise 500 to 1400 images and were recorded using rendering tools, robot arms and handheld cameras. Camera pose ground truth is available for half of the sequences (exploration, robot arm).

In the course of the evaluation, we vary a number of parameters of our system acting as independent variables in controlled experiments and observe a number of depending variables. We select the following system parameters for the evaluation:

Keyframe Selection Overlap The overlap ratio parameter influences the frequency of keyframe selection, which has a strong impact on the connectivity of the pose graph, reference keyframe selection and reconstruction, as well as the resulting map data volume and mapping performance. We compare the overlap ratio set $\{60\%, 70\%, 80\%\}$. For the evaluation, we disabled the pose consistency check to reduce the required number of adjacency keyframes to one (instead of two). However, smaller overlaps than 60% regularly result in candidate keyframe rejection, because the mapping component fails to estimate any homographies.

Reference Keyframe Selection We are interested in the effects of either using central or leading keyframes as reference for the reconstruction of the different trajectory types.

Reconstruction Approaches We compare four combinations of homography path retrieval approaches for plane estimation and pose update respectively. These are (plane estimation approach / pose update approach): (1) paths using all edges / shortest paths using uniform edge weights, (2) paths using all edges / shortest paths using homography motion edge weights, (3) shortest paths using homography motion edge weights / shortest paths using homography motion edge weights, (4) shortest paths using uniform edge weights / shortest paths using uniform edge weights.

We process the test sequences with all resulting 24 system parameter combinations and measure a number of dependent output variables for comparison:

Accuracy. We record structure and motion information including the keyframe camera poses, 3D points and corresponding 2D image observations. The resulting camera trajectories are registered with ground truth, where available, to compute re-projection and object space error. Re-projection error is defined as the RMS of pixel differences of projected 3D points with their corresponding 2D observations. Object space error is defined as RMS of camera position differences with the ground truth. The object space RMS is normalized using the trajectory size, which is defined as the 3D range of the volume spanned by the ground truth camera positions.

Performance and Data Volume. To assess the performance of our method, we monitor the timings of the individual mapping components. Additionally, we track the data volume kept in the map, including number of keyframes, 3D points and homographies. The amount of data to be processed naturally influences the overall system performance.

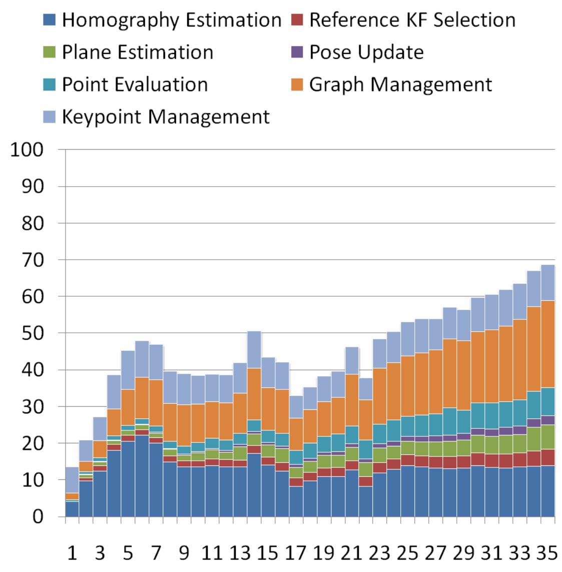
Besides our own system, we evaluate the publicly available PC-version of PTAM [72] on the same test sequences. We modified the software with logging and monitoring functionality. We run PTAM three times on each sequence to compensate for outlier tests and averaged the results for the evaluation.

Both PTAM and our system provide structure and motion information, which is passed to the SBA bundle adjustment software package [89] for refinement. We pass camera poses, 3D point estimates and 2D observations to the algorithm, which is configured to compute optimal results disregarding time constraints. For the two of four sequences where no ground truth is available, we use the SBA output as reference to compute the re-projection and object space error of PTAM and our system, respectively.

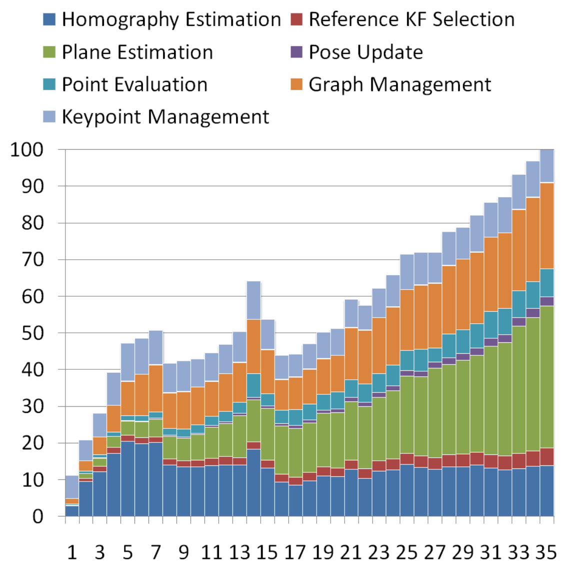
3.4.1 System Parameters

We evaluate the effects of individual parameters by varying one parameter and averaging the corresponding subset of all other parameter combinations and compare the resulting accuracy and performance results. For each test sequence, we performed 24 trials using all system parameter combinations. Our system successfully processed all trials on all test sequences except the "loop" sequence, where the system failed in five trials to close the loop resulting in a defunct tracking component, because of inconsistent map point observations. We cannot discover any specific parameter closely related to the failure cases.

We reviewed the computational complexity of our mapping method depending on the



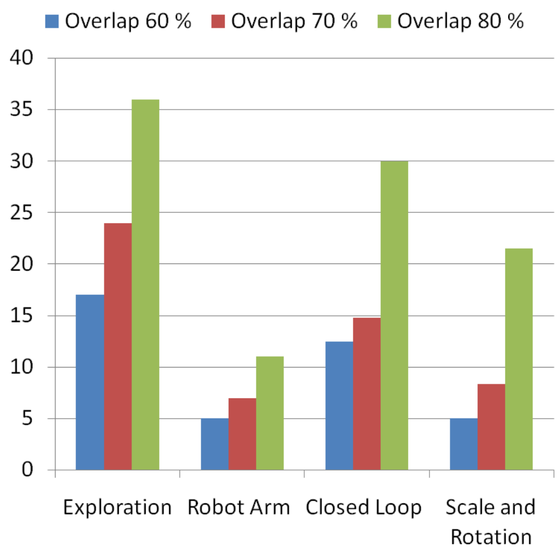
(a)



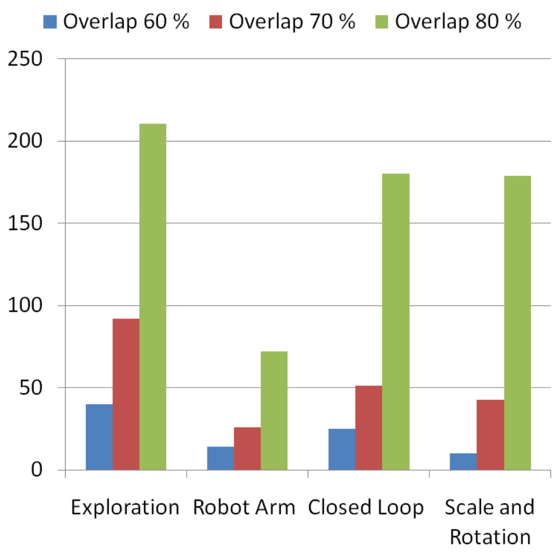
(b)

Figure 3.11: Times in milliseconds spent by our mapping method components for processing maps with an increasing number of keyframes. Separate diagrams for plane estimation approaches retrieving (a) shortest paths from reference to other keyframes and (b) paths using all graph edges. For these diagrams we employed the "exploration" sequence with the following system parameters: 80% keyframe selection overlap ratio and central reference keyframe. The maps having 35 keyframes both feature 210 homographies. Point counts are (a) 2391 and (b) 2395 respectively.

number of keyframes, map points and homographies, as depicted in Figure 3.11, which shows the timings of individual components in milliseconds. Homography estimation is expensive, but constant in the number of additional homographies which are computed for each keyframe candidate. Map point management refers to the addition of a constant number of new points and point observations for a new keyframe and includes checking the map region state for each point using the quad tree, a logarithmic operation in the number of map points P. Graph management incorporates the constant time for adding new keyframe nodes and homography edges as well as the computation of edge weights using the rectification algorithm. The graph operations (*e.g.*, breath-first searches), required for reference keyframe selection, can be done in $O(F(F+H))$ in the number of keyframes F and homographies H. Reconstruction comprises of pose update in $O(F)$ and point evaluation in $O(P)$ after estimating the plane. The expense for plane estimation, however, depends on the reconstruction approach: retrieving paths using all H edges (see Figure



(a)



(b)

Figure 3.12: Effects of varying keyframe selection overlap ratios on the number of (a) keyframes and (b) homographies.

3.11(b)) is significantly more expensive than retrieving solely the shortest paths from reference to other keyframes (see Figure 3.11(a)). Overall, we see mostly constant and linear asymptotic behavior in the number of keyframes, points and homographies.

The overlap parameter has considerable effects on the map data volume, thus on the overall system performance (see Figure 3.12). With increasing overlap, the number of keyframes steadily increases resulting in a denser sampling of the 3D space. In turn, candidate keyframes feature more adjacent keyframes allowing the estimation of more homographies and influencing the connectivity of the graph.

The impact of the overlap parameter on accuracy yields no significant results. Surprisingly, more data does neither improve object space nor re-projection error significantly. We conclude that already with 60% overlap, our mapping method receives sufficient data to solve the planar reconstruction in good quality with re-projection errors below 3 pixels and object space errors below 1.5% of the trajectory sizes. Similarly, the accuracy results for the different reference key and reconstruction approaches do not show clear trends. For both parameters, re-projection and object space errors remain within $\pm$ 10%, respectively.

3.4.2 Accuracy

We compare the accuracy of our method with the bundle adjustment implementations of SBA and PTAM. The structure and motion information of each test run of our system and PTAM, respectively is passed to SBA for refinement, resulting in two camera trajectories each. Registration to ground truth is performed for the resulting camera trajectories individually. For test sequences without ground truth (loop, scale and rotate), the system trajectories are registered to the respective SBA trajectory, resulting in the zero object space errors of the SBA trajectories. For the diagrams of Figure 3.13, we averaged the results of the 24 test runs of our system and the three test runs of PTAM, respectively.

Our system features object space errors between 0.5% and 1.5% of the trajectory sizes, thus performs similar to PTAM and SBA on the ground truth sequences. We consider the bad result of PTAM on the robot arm sequence as an outlier. On the handheld image sequences, PTAM performs with a significantly lower object space error. We conclude that our plane estimation and pose update components do not range on the same accuracy level as bundle adjustment, but keep up, with some trade-offs.

The re-projection errors of our system range between 0.5 and 2.5 pixels and are two to

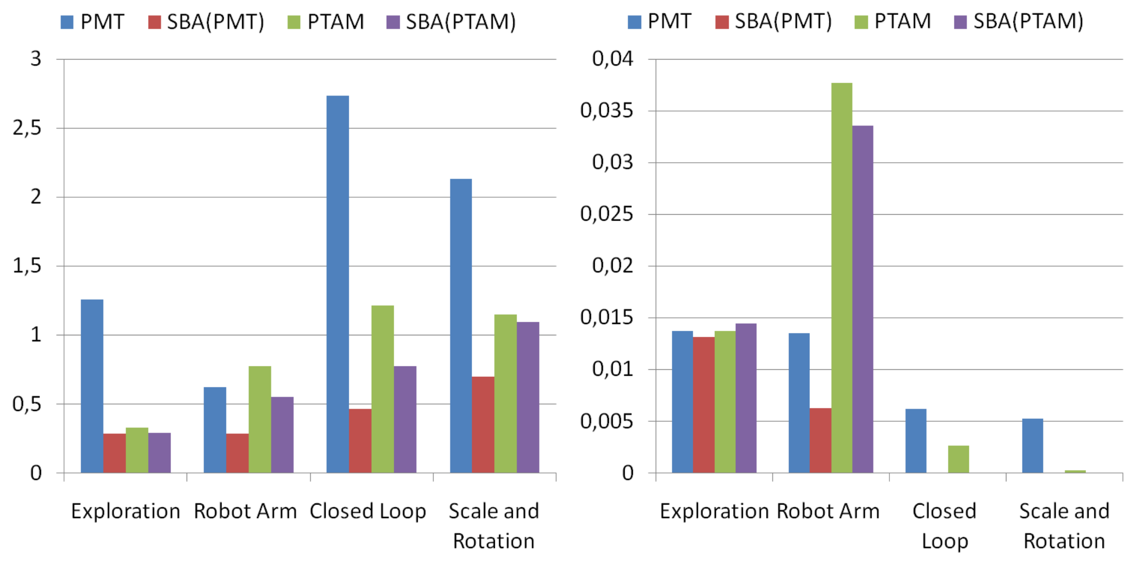


Figure 3.13: Accuracy comparison of our method with SBA and PTAM: (a) re-projection error as RMS of pixel differences, (b) object space error as RMS of camera position differences with ground truth (normalized using test sequence trajectory sizes).

five times larger compared to SBA and PTAM. Overall, SBA yields the least re-projection error as expected. We reason that this result is due to our optimization method, which minimizes an algebraic cost function (3.6, 3.7) rather than the re-projection error, as done by SBA and PTAM. We calculate the location of 3D points efficiently, but coarsely as centroid of back-projected 2D image observations. In comparison, the optimization of SBA and PTAM has additional degrees of freedom concerning the placement of 3D points with a variable z-coordinate. Thus, as our system places 3D points strictly on the canonical plane (with z-coordinates equal zero), we do not account for noise and non-planar outliers.

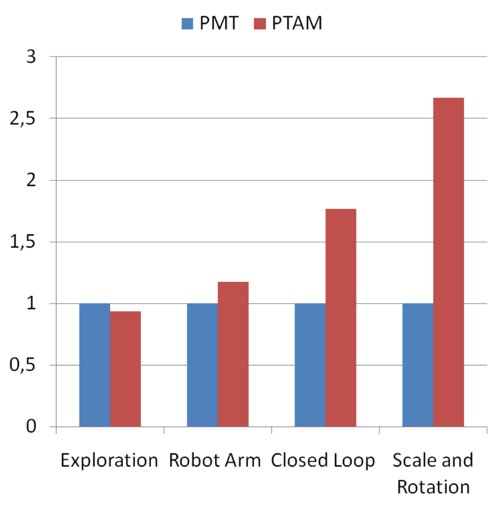
Overall, we see our method perform well in comparison to state-of-the-art bundle adjustment techniques on the selected indoor test sequences.

3.4.3 Performance and Data Volume

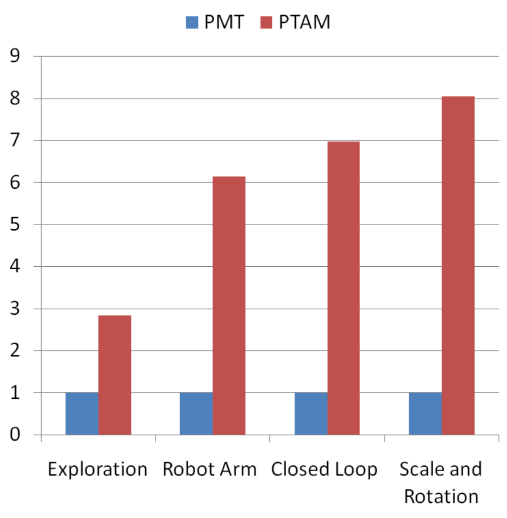
The performance of our system is compared to PTAM. We contrast the data volume used for map optimization and the total time spent for mapping when processing the individual test sequences as shown in Figure 3.14. The depicted values are averaged results of all test runs of our system and PTAM, respectively, and normalized with respect to our system.

Our system manages a considerably smaller amount of data concerning the number of keyframes and map points. The number of keyframes selected by PTAM is up to 2.5 times higher. We perform keyframe selection based on overlap, which both accounts for translational and rotational frame differences, while PTAM selects keyframes solely based on translational differences. The difference in the number of map points is even more obvious, with PTAM adding up to eight times more. We manage the points on the canonical plane in a quad tree, which allows to detect well-populated and unpopulated map regions efficiently. By performing these checks, our systems may considerably reduce the number of points inserted into the map.

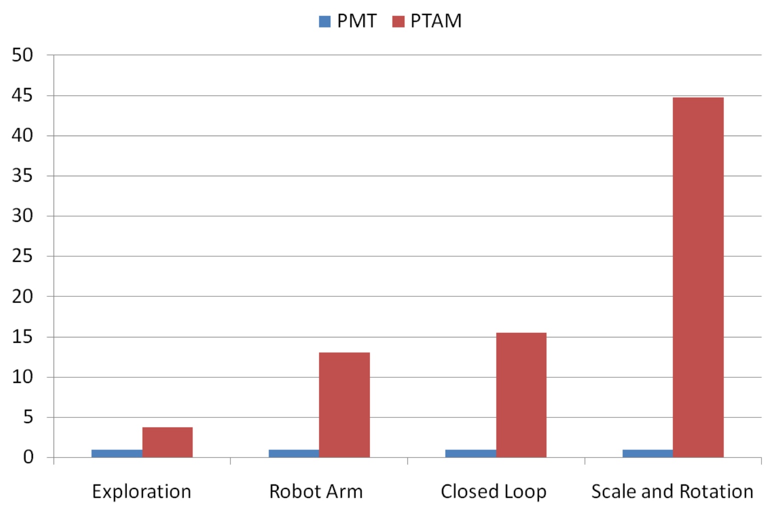
Comparing the total mapping times over entire test sequences, our system outperforms PTAM with factors between three and 45. Using optimized system parameters, *e.g.*, 70% keyframe selection overlap combined with a reconstruction approach using shortest paths, we may raise factors up to 15 on the "exploration" sequence, which stays behind in comparison to the other sequences. This result validates our system design that focuses on performance at the expense of reduced accuracy.



(a)



(b)



(c)

Figure 3.14: Data volume and performance comparison with PTAM: (a) Keyframe and (b) map point counts. (c) Total time spent for mapping over the entire test sequences. All values are normalized with respect to our system (having values equal one).

3.5 Mobile Phone Results

We employed the Nokia N900 phone to explore our systems' scalability, performance and robustness on a mobile platform. The Nokia N900 is equipped with a single-core ARMv7 600 MHz CPU and 256 MB memory. We processed camera images with a resolution of 320x240 pixels and configured the following system parameters: 70% keyframe selection

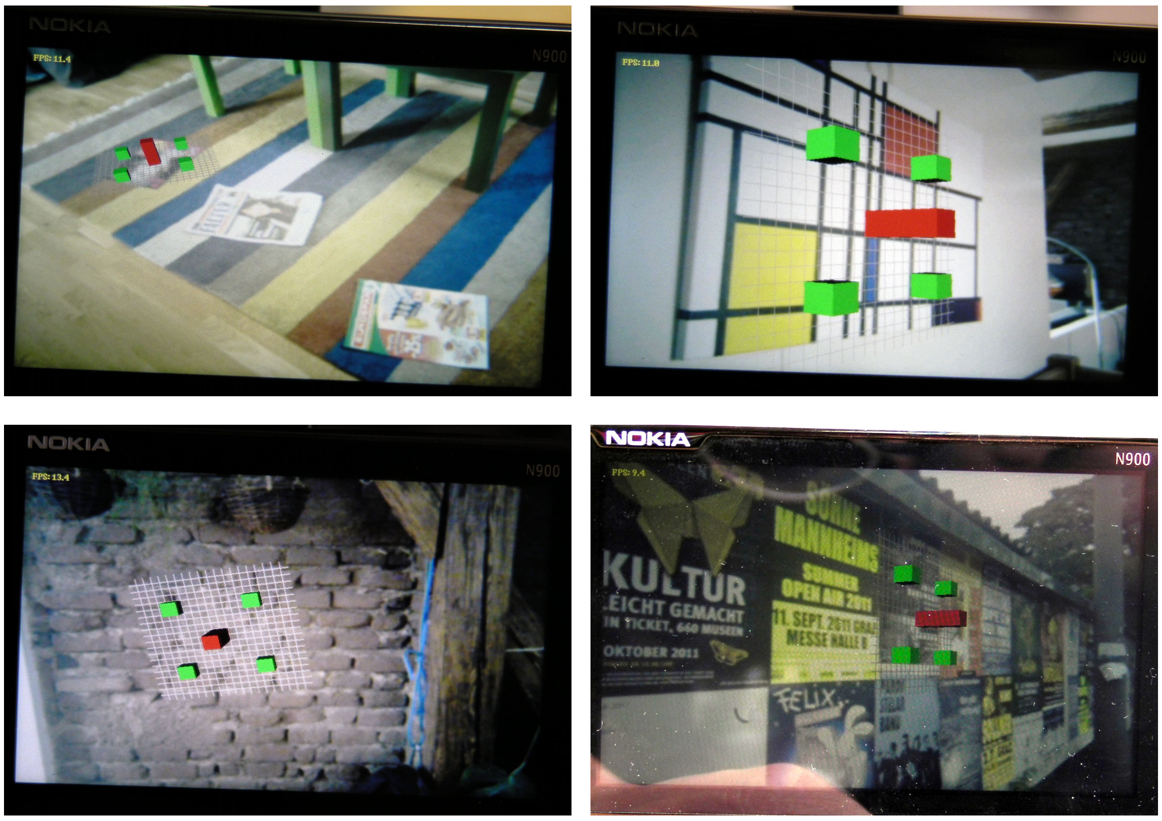


Figure 3.15: Examples of planar indoor and outdoor scenes mapped and tracked with our system (left to right): carpeted floor (with additional texture), wall with tableau, brick wall, outdoor billboard.

overlap, central reference keyframe and reconstruction using shortest paths with uniform edge weights.

Mapping and tracking table-sized indoor scenes such as depicted in Figure 5.1 at different scales, our system reports frames rates of 6-12 FPS on the Nokia N900 phone. One specific test with a map consisting of 15 keyframes, 1069 points and 76 homographies yielded 15-16 FPS during the initialization phase and 6-12 FPS during the map tracking and mapping phase while consuming roughly 16 MB of memory. In the initialization phase, our system performs homography tracking and estimates the plane using a single homography, resulting in a planar map. Subsequently, pose tracking ran with 10-12 FPS largely depending on the density of map points projected into the current frame. Mapping single keyframes took 250 milliseconds on average with timings increasing from 170 to 430ms. Tracking and mapping are executed in separate threads. However, the single-core

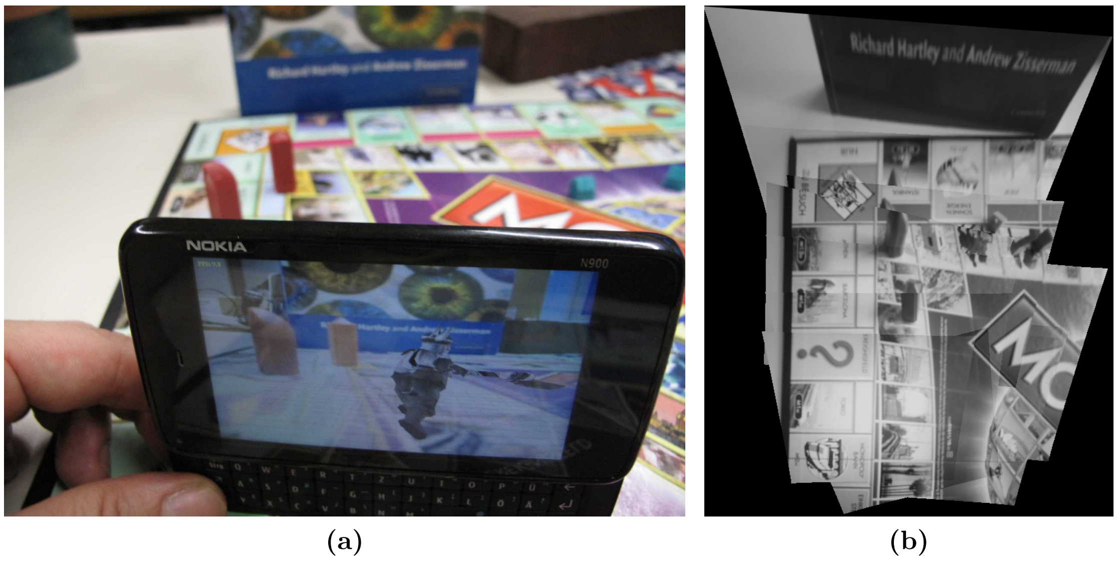


Figure 3.16: Robustness test on the mobile phone: (a) scene with outlier objects results in (b) distorted maps due to homography estimation errors.

CPU of the Nokia N900 does not allow for running these threads in parallel. Alternatively, we schedule mapping in a round-robin fashion and prioritize tracking. Keyframe mapping is performed in several time slices by suspending the thread after a timeout (*e.g.*, 20ms), thus switching to the tracking thread. This strategy preserves tracking and resulted in frame rates of 6-9 FPS. Yet, running threads in parallel on modern multi-core CPU phones should yield considerable performance gains.

We added outlier objects of various sizes to the scene to assess the robustness of our system. Outliers influence our mapping system via the homographies estimated between keyframe images showing these objects. The resulting maps become distorted which, in turn, degrades pose tracking. In Figure 3.16 we depict our system mapping and tracking a scene which contains several outlier objects. The book in the background defines a completely different plane. The depicted map image shows the effects of outlier objects on map quality. However, our pose consistency check rejected candidate keyframes showing larger portions of the background outlier plane.

3.6 Summary

This chapter presented a planar mapping and tracking system capable of operating robustly in real time on mobile phones by restricting the mapping problem to a single scene plane. Considered as one of the computationally most light-weight mapping algorithms in the literature, it provides a profitable trade-off between accuracy and performance. Our approach uses a fraction of the computational resources required by comparable monocular 3D SLAM systems, while incurring a moderate decrease in accuracy. Overall, the algorithm scales either linearly or quadratically in the number of keyframes, depending on the system configuration. Therefore, as we demonstrated, our system is also suitable for mobile devices with a single-core CPU and works on planar scenes in indoor and outdoor environments (see Figure 3.15).

Similar to PTAM, our system provides the dominant scene plane as a "playground" for the annotations of top-level AR applications. Additionally, by rendering the mapped planes as orthographic images, the system allows for scanning and generating persistent map representations. These image templates could either be employed online, *e.g.*, in combination with object detection algorithms, or saved and reused later on by other applications such as model-based tracking and detection, or image-based modeling.

To further increase the understanding of the environment, the system could be extended towards explicitly detecting different planes and starting to extend the map to these new planes. This could lead to an overall system that explicitly represents the environment as a set of planar surfaces instead of sparse clouds of point or line features. Meanwhile, Salas-Moreno *et al.* [133] have demonstrated a fusion-based SLAM system of this kind, but only on the desktop and by employing an RGB+Depth (RGB-D) camera.

Due to the constrained planar scene assumption, the mapping algorithm is not particularly robust against outlier measurements. While we attempt to detect violations of the planar map model, such as objects sitting on a table surface, the system cannot compete with the robustness of general SLAM systems, which perform explicit 3D mapping.

4 Handling Pure Camera Rotation in Keyframe-Based SLAM

In this chapter, we present a keyframe-based hybrid SLAM algorithm, which handles both general, that is, parallax-inducing, as well as rotation-only, that is, parallax-free, camera motion. In the past, SLAM systems have been presented which specialize in mapping and tracking either general or rotation-only camera motion. General SLAM systems with six degrees of freedom (6DOF) [33, 39, 72, 110] assume general camera motion and apply structure-from-motion techniques to create 3D feature maps. Robust triangulation of 3D map features from observations of multiple camera viewpoints requires sufficient parallax induced by translational or general camera motion. In contrast, panoramic SLAM systems [35, 26, 90, 173] assume rotation-only camera motion and track the camera in 3DOF. Because no parallax is observed, feature points are not triangulated and, consequently, can only be thought of as rays. In the following, we call such rays *infinite features*, while 3D points from 6DOF SLAM are called *finite features*.

Panoramic and 6DOF SLAM have complementary strengths and weaknesses: 6DOF SLAM cannot handle pure rotational camera movements well. Tracking may be lost, and in unfortunate situations, erroneously measured finite features may corrupt the map. In contrast, panoramic SLAM can only handle rotational motion. Any translational motion component may be encoded as additional rotation, also leading to a degradation of map quality.

The inherent problem of handling both general and rotation-only camera motion is partly created by the real-time processing constraint of SLAM systems. As incoming video frames can only be triangulated with respect to the map known up to the current moment, it is rather straightforward to discard frames that cannot be triangulated yet. However, keeping more potential keyframes around would increase the likelihood of match-

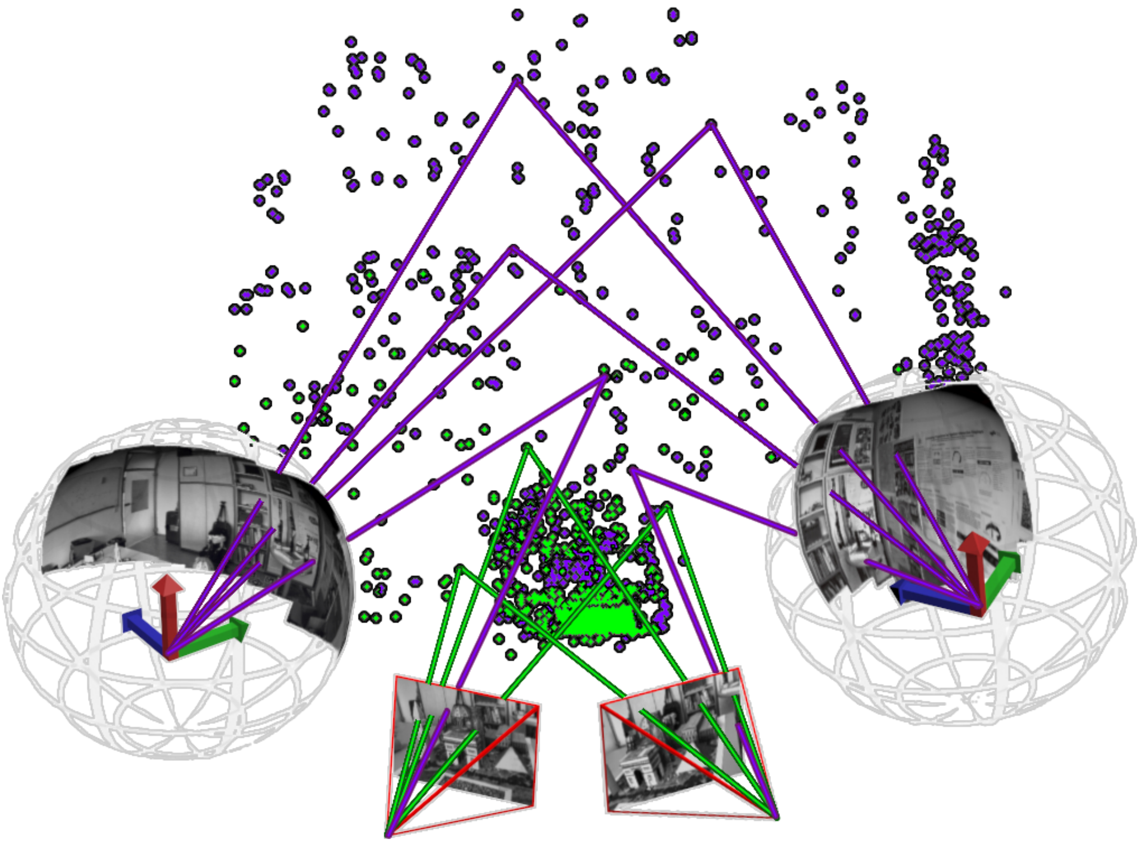


Figure 4.1: Rotation-only camera movements are handled by tracking and mapping local panorama maps registered within a global 3D map. The information contained in the panorama maps is also used for 3D reconstruction.

ing observations in later frames. This is in contrast to full structure-from-motion systems that have complete information available and, therefore, can match with both past and future frames.

We propose to combine the advantages of 6DOF and panoramic SLAM into a hybrid keyframe-based system that accepts both fully triangulated keyframes for normal 6DOF operation as well as keyframes with only rotational constraints. This combination contributes in several ways to an optimization-based SLAM system:

- Tracking can cope with pure rotation and provide a more seamless experience to the user.
- Mapping has more keyframes available to estimate new parts of the 3D SLAM map.
- As we extend state-of-the-art approaches, we obtain a system that performs as least

as well as a normal 6DOF SLAM.

The tracking component can dynamically and seamlessly switch between full 6D and panoramic tracking modes, depending on current motion performed by the user. It is designed to handle temporary rotations away from the mapped part of the scene that users often make in practice. We detect these rotations and select special "panorama" keyframes that are used to build up local panorama maps. The local panorama maps are registered in a single consistent 3D map. The observed finite and infinite map features allow for robust tracking of alternating phases of general and rotation-only motion with a unified pose estimator. Additionally, we support re-localization.

The transition from a panorama map to another 3D sub-map is not intended, since global map scale consistency would be lost. However, we explicitly exploit observations of infinite features measured in panorama keyframes in the construction of the global 3D map, if they can be combined with observations in later keyframes allowing for deferred full triangulation of the feature.

We demonstrate the efficacy of our approach in several evaluations that show how the combined system handles rotation only camera motion, while creating larger and denser maps, compared to a standard SLAM system (see Section 4.5).

4.1 System overview and map representation

The system architecture of our hybrid approach follows the standard two-component design of optimization-based SLAM [72]. Tracking and mapping components are executed in separate threads and synchronize via dedicated keyframe and map interfaces. The keyframe interface allows the tracker to submit keyframes, which are asynchronously processed by the mapper. The map interface allows the tracker to retrieve a snapshot of the current map.

The tracking component performs robust frame-rate map tracking of 6DOF or rotation-only camera motion and selects new keyframe candidates. In our system, keyframes can either be fully localized with a 6DOF pose or panorama keyframes that are described with a rotation relative to a reference pose. The mapping component adds keyframes to a single global map, comprising both types of keyframes and finite and infinite features. Additionally, it also refines the map through establishing new data associations and bundle adjustment optimization.

Our system builds and maintains a single global map that has a consistent scale. The

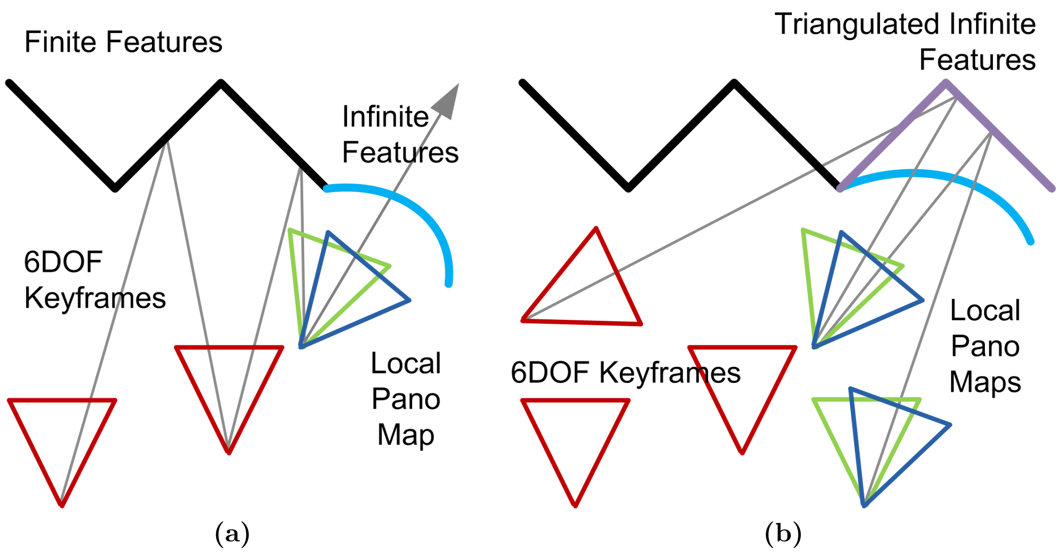


Figure 4.2: Relationships in our hybrid map representation between keyframes and features depicted in two stages. In stage (a), 6DOF keyframes observe finite map features. Local panorama maps are registered in the 3D map via reference panorama keyframes (green) that have finite and infinite feature observations, while the remaining dependent panorama keyframes (dark blue) observe infinite features only. In stage (b), infinite features are triangulated from corresponding observations matched between additional 6DOF keyframes and/or localized panorama keyframes from different local panorama maps. Note that the additional features enable the localization of further dependent panorama keyframes.

map is composed of finite and infinite point features, which have 2D image observations in regular 6DOF and panorama keyframes. Figure 4.2 shows the relationships within our hybrid map representation.

The map is represented as a collection of keyframes that store the camera pose C_i of the keyframe and an image pyramid for tracking points and finding new correspondences. We use the pose C_i to denote the keyframe itself. Keyframes generally fall into two categories, either full 6DOF frames or panorama keyframes. Full 6DOF keyframes are denoted by the set $K = \{C_k\}$. Panorama keyframes are organized in local panorama maps $P_j = \{C_{i,j}\}$ that are registered in the 3D map with its center of rotation. The center of rotation is determined by a dedicated reference panorama keyframe C_j^r which is both a 6DOF frame and part of the panorama map, thus $\{C_j^r\} = K \cap P_j$. The remaining panorama keyframes $C_{i,j}$ are dependent and effectively have only a 3DOF rotation pose relative to the reference keyframe C_j^r.

Point features are represented as homogeneous 4-vectors $X_i = (x, y, z, w)^T$, where

$w = 1$ for finite points with known 3D location, and $w = 0$ for infinite points that were only observed in panorama keyframes. Infinite points are observed in one or more panorama keyframes of a single local panorama map.

4.2 Tracking

The tracking component processes the video stream of a single calibrated camera at frame-rate and tracks both general and rotation-only camera motion with respect to a global map that consists of finite and infinite features. The pose estimation combines measurements of both finite (known 3D location) and infinite features, and automatically computes either a 6DOF or 3DOF pose update. In case of incremental pose estimation failure, we provide a re-localization method (see Section 4.2.2) based on small blurry images [73].

4.2.1 Incremental pose tracking

Starting with a known pose from the previous input frame, the current camera pose is predicted by a simple decaying constant-velocity motion model.

We select a feature set for matching from all map features by filtering features for (1) visibility from the predicted camera pose, (2) only infinite features of the currently enabled panorama map, (3) overlapping feature re-projections, where we prefer finite over infinite features. At any point in time, either none or exactly one panorama map is enabled for tracking. Thus, we are only considering infinite features, if the center of rotation of the corresponding panorama map is located close to the camera. Note that panorama maps and keyframes are always used for mapping.

Then, the following steps are executed for each image pyramid level of the current frame, starting at the lowest level. We actively search for each selected feature in the current frame using NCC as score function. Matches with a sufficiently high NCC score are added to the correspondence set that is processed by our unified relative pose refiner. This yields a set of 2D observations O_i for map features X_i.

Given a pose prior and the set of observations, we iteratively estimate incremental pose updates. We optimize the re-projection error

$$E(C) = \sum_i W_i \| \operatorname{Proj}(C \cdot X_i) - O_i \|^2 \tag{4.1}$$

using standard Gauss-Newton iteration both for finite and infinite map points, where $\operatorname{Proj}(\cdot)$ is the camera projection including radial distortion.

For a given camera pose $C = \begin{pmatrix} R & T \end{pmatrix}$, the transformation of a map point $X = (x, y, z, w)$ is

$$C \cdot X = R \begin{pmatrix} x \\ y \\ z \end{pmatrix} + T \cdot w. \tag{4.2}$$

Thus, finite points add a constraint on the camera translation ($w = 1$), while infinite points do not ($w = 0$). To ensure that the system is stable, even if no finite points are observed (as in a pure panoramic tracking mode), we add a small regularization term to the linear system.

Additionally, we also apply a weight W_i to each measurement to balance the influence of infinite and finite features. Typically, we want to rather follow finite features (weighted with $W_i = 1$), therefore we weight infinite features with a factor $W_i = 0.01$ that was empirically determined.

4.2.2 Relocalization

We perform relocalization based on small blurry images (SBIs) that work with both 6DOF and panorama frames. A small blurry image is a small downscaled version of a video frame (40x30) with Gaussian blur applied. A history of SBIs is recorded with frames added at regular time intervals together with their 6DOF tracking poses. We query the history with an SBI computed from the current frame, resulting in a sorted set of similar SBI candidates. Each candidate is verified: First, the stored candidate pose is updated by estimating a relative 3DOF rotation between candidate and current frame with ESM [15]. Next, we execute the active search map tracker using the updated 6DOF pose as prior. If tracking succeeds for any of the candidates, the resulting pose is used to re-initialize relative pose tracking.

4.3 Keyframe selection and insertion

Selecting and inserting new keyframes into the map is an essential step in an efficient optimization-based SLAM system. To keep processing requirements low, only important keyframes should be chosen from the video stream. Important keyframes have two properties: (1) They image new parts of the scene, or known parts from different directions; (2) they have enough observations of known structure to ensure good connectivity in the map. Enabling the system to take more keyframes is essential to have a more detailed

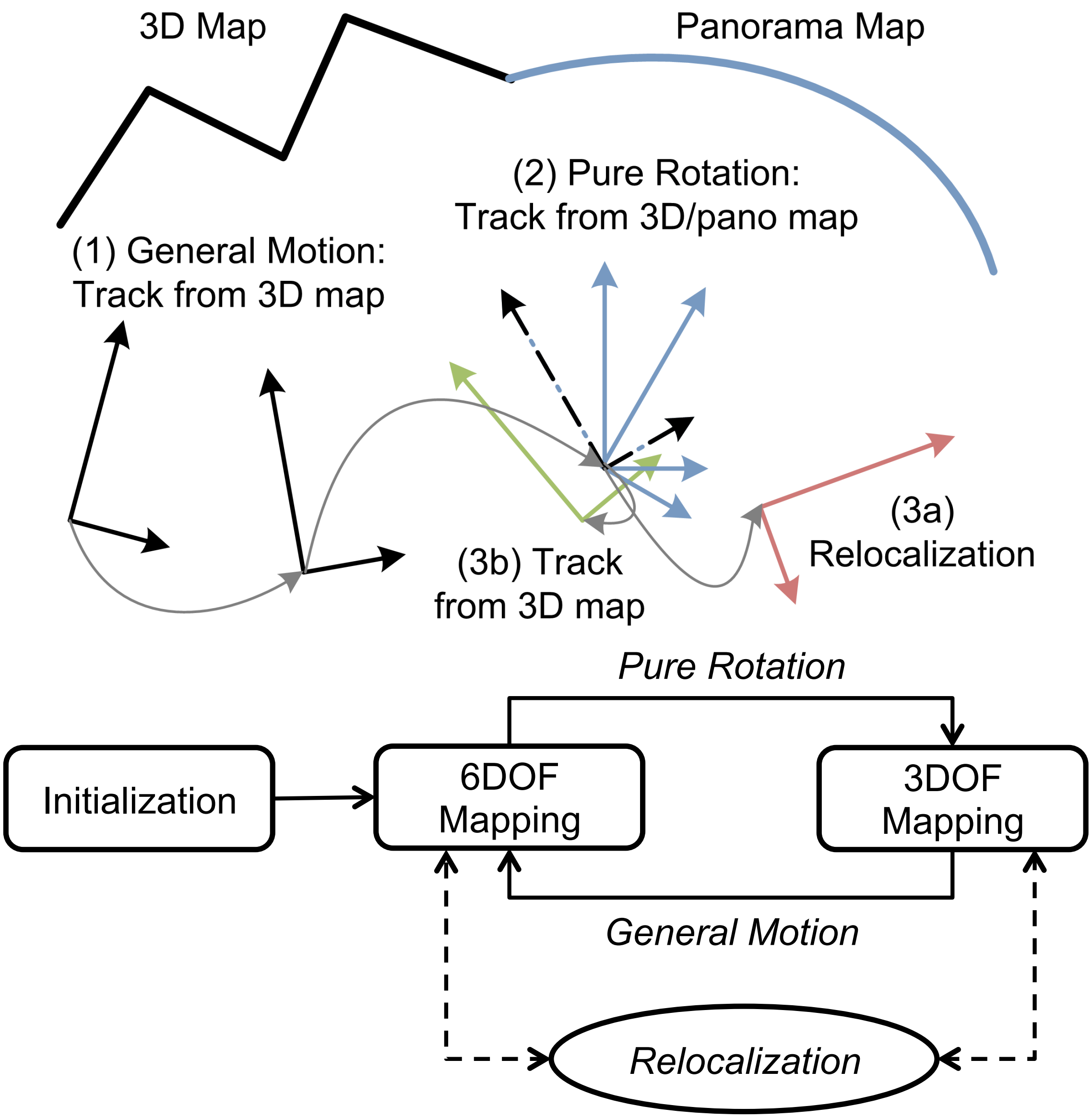


Figure 4.3: State diagram for the different states of keyframe selection during mapping. After initialization, the system starts to operate in full 6DOF mapping mode (1). If pure rotation motion is detected, the system switches to 3DOF mapping mode and creates a new panorama map (2). 6DOF measurements move the system back to full 6DOF operation (3b). In case of tracking failure, relocalization always recovers a full 6DOF pose (3a).

or expansive map. Our system is able to record more keyframes by relaxing the second requirement. Keyframes that are not well constrained in 6DOF through known 3D map features, but only through 2D-2D observations, are recorded as well, if they image substantially new parts of the scene.

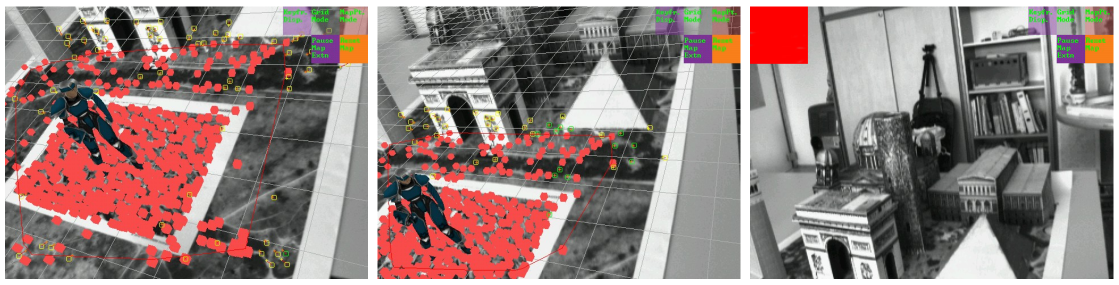
The top priority of keyframe insertion is to avoid map corruption by carefully selecting 6DOF keyframes and robustly localizing panorama keyframes. For 6DOF keyframe selection, we require a large number of correspondences for pose estimation and a conservative threshold for parallax. For panorama keyframe selection, we relax the rules: the pure rotation tracking may drift due to small camera translations. However, the relaxation does not harm the 3D map, as panorama keyframes are used for mapping only after they have been robustly localized later on. For localization, we again enforce strict thresholds on the pose estimation.

Similar to PTAM [72], our method uses heuristics for keyframe selection. We have adopted parallax and tracking quality criteria, and combined it with coverage and camera view angle criteria. In practice, distinguishing between true rotation-only motion and small translations is impossible due to measurement noise. Therefore, we use the parallax criterion to detect camera motion that allows for robust triangulation and 3D reconstruction. In the following, we describe the selection of keyframes in more detail. Figure 4.3 shows an overview of the different keyframe selection modes during operation.

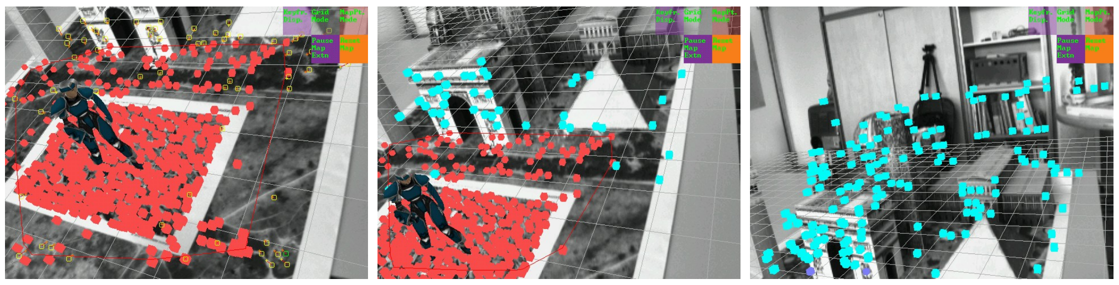
4.3.1 6DOF mapping

6DOF keyframes can be selected whenever we have a camera pose that is fully constrained in 6DOF. This is the case, if enough finite feature points are part of the pose estimation, as described in Section 4.2.1. Furthermore, we select regular 6DOF keyframes when they generate enough parallax to existing keyframes, while imaging a new part of the scene. Parallax is required for robust feature triangulation. New parts of the scene are characterized by areas in the image where few or no known features project into.

Parallax is the angle α between the viewing directions of two camera frames (*e.g.*, between the current frame and an existing keyframe) onto the commonly viewed scene geometry. It can be approximated as $\alpha = 2\arctan(d/(2f))$, where f is the mean depth of the finite map features observed in both frames, and d is the relative distance between the 3D locations of the frame pair. Note that, in contrast to distances, the parallax angle is scale-independent. We empirically determined a parallax angle of $\alpha > 5°$ as sufficient for 6DOF keyframe selection.



(a)



(b)

Figure 4.4: Handling pure-rotation camera motion at the map boundary. The same three non-consecutive frames as processed by (a) 6DOF SLAM and (b) hybrid SLAM (our approach). 6DOF SLAM discards low-parallax candidate keyframes, resulting in tracking failure due to a lack of new finite map features (rendered in red). Hybrid SLAM detects the pure-rotation camera motion, creates a local panorama map, and continues camera tracking from infinite map features (rendered in cyan).

To estimate low coverage, we compute the frame area ratio that is covered with finite feature projections. We divide frames into a regular grid with 4×3 cells and project finite map features. Grid cells with a minimum number of contained features are considered covered. The coverage is the ratio c of the number of covered to all grid cells. We empirically determined that a frame is not well covered if the coverage ratio $c < 0.75$.

When inserting the 6DOF keyframe into the map, we add new finite map feature observations and new finite map features. New observations are added to features which have been successfully tracked in the keyframe. New finite 3D map features arise from frame-to-frame 2D feature tracking between the previous and the current 6DOF keyframe. With the insertion of the current keyframe, the new correspondences are added as new finite features having triangulated 3D positions $(x, y, z, 1)$ and two observations.

4.3.2 Localized panorama keyframe insertion

When the system detects low coverage (*e.g.*, $c < 0.75$) but not enough parallax (*e.g.*, $\alpha < 5°$) between the current frame and existing keyframes, then tracking may fail, if all known features become invisible. Low coverage indicates that the camera points towards unmapped scene regions. However, a regular 6DOF keyframe cannot be taken, due to low parallax of pure-rotation camera motion. Thus, we select a panorama keyframe that is localized with respect to the 3D map. Figure 4.4 illustrates the different behavior of standard 6DOF SLAM and our approach.

We detect pure rotation camera motion based on the history of tracked 6DOF poses. Tracked 6DOF poses are stored chronologically in a history. We compute the parallax angle between the current pose and the history and discard all poses with a sufficiently high parallax (*e.g.*, $\alpha > 5°$). The remaining history poses have similar 3D locations as the current frame. Finally, we compute the angles between viewing directions and detect pure rotation, if we find a pose in the history that has low parallax and a sufficient angle with respect to the current frame. The view difference angles β are normalized with the field-of-view angle γ of the calibrated camera, resulting in a angle ratio $r = \beta/\gamma$. We empirically determined a ratio $r > 0.2$ as sufficient for pure-rotation detection.

The selection of a localized panorama keyframe marks the beginning of a local panorama map. The system creates a new local panorama map P_j and assigns its reference keyframe C_j^r that defines the center of rotation.

When inserting the localized panorama keyframe into the map, we add new observations and new infinite features. New observations are added to finite features which have been successfully tracked in the keyframe. New infinite features are initialized from 2D image features. We apply a corner detector on the keyframe, resulting in a 2D image feature set and subtract the projections of existing map features. The remaining 2D image features (u, v) are converted into rays $(x, y, z, 0)$ in world space and added as new infinite map features with a single observation.

4.3.3 Transition to 3DOF mapping

As soon as we do not observe sufficient finite features in the current frame, the hybrid pose estimation only updates the 3D orientation from infinite map features around a fixed 3D position using the local panorama map P_j. Slight camera translation may be estimated as additional rotation into the 3DOF poses.

4.3.4 Panorama keyframe insertion

The system continues to select panorama keyframes based on low coverage (*e.g.*, $c < 0.8$) and sufficient rotation. Low coverage indicates the camera continues to explore unmapped scene regions. Rotation is computed as the difference angle between the viewing directions of the current frame and keyframe poses of the current panorama map. Again, we normalize the view difference angle with the camera field-of-view angle, and determined sufficient rotation if the ratio $r > 0.2$.

When inserting the panorama keyframe into the map, we add new infinite feature observations and new infinite features. New observations are added to infinite features which have been successfully tracked in the keyframe. New infinite features are computed the same way as with localized panorama keyframes (see above).

4.3.5 Transition back to 6DOF mapping

The system implicitly moves back to the full 6DOF operation, if it observes parts of the 3D map again. Then the same criteria as before apply, and a new 6DOF keyframe can be created.

With the transition, the panoramic tracking session ends, and the current panorama map is disabled. Observations of map features within panorama keyframes of this session are disabled so that they are ignored by tracking future frames. We disable all feature observations of non-localized panorama keyframes. Localized keyframes keep their finite feature observations.

4.4 Mapping

The mapping component refines the map through establishing new data associations and bundle adjustment optimization. In particular, it also estimates full 6DOF poses for panorama keyframes and triangulates infinite features to extend the 3D map. As part of data association refinement, we seek new keyframe-feature observations to further constrain existing feature locations and keyframe poses. We apply active search and descriptor matching techniques to establish 2D-2D correspondences.

We robustly localize panorama keyframes with respect to finite map features. Panorama keyframes are initialized with poses from panoramic tracking that are considered unreliable since we cannot estimate the poses in full 6DOF from infinite features and, thus, cannot measure camera translation. However, by establishing correspondences to existing finite map features, we can estimate full 6DOF poses. Thus, we effectively convert panorama keyframes into regular 6DOF keyframes.

We exploit the information stored in local panorama maps for 3D mapping by triangulating infinite feature observations. We employ descriptor matching to find 2D-2D correspondences between robustly localized keyframes, *e.g.*, in separate local panorama maps that view the same scene regions. Correspondences which pass the verification tests constitute additional finite map features. Thus, we effectively convert infinite to finite features.

Finally, we also optimize the map with bundle adjustment [43]. Bundle adjustment updates the 6DOF poses of localized keyframes and the 3D positions of finite map features by minimizing again the reprojection error between feature locations and observations in keyframes. Non-localized panorama keyframes and infinite features are not optimized. However, we maintain map consistency by adjusting the registration of panorama maps within the optimized 3D map.

4.4.1 Panorama keyframe localization

Robust localization of panorama keyframes is an iterative process that finds new correspondences between infinite features in the panorama keyframes and finite features observed in normal keyframes. Once enough such correspondences are established, a dependent panorama frame contained in a local panorama map can be localized with a full 6DOF pose and converted to a normal keyframe. This, in turn, can lead to further triangulation of infinite feature points, which again may allow for localizing other panorama keyframes.

To find correspondences to finite features, we employ both active search and wide-baseline matching using visual descriptors. The active search technique is borrowed from relative pose tracking. We iterate over all finite map features. If a particular feature does not have an observation in the panorama keyframe, we project the feature and perform NCC matching in the neighborhood of the projected 2D image location. If we get a sufficiently high NCC score, we have found a 3D-2D correspondence and add a new feature observation.

Since the active search method relies on a reasonably accurate panorama keyframe pose, we additionally perform wide-baseline descriptor matching. We maintain a database (see Section 4.4.4) that contains descriptors of all finite feature observation patches in its leaves. The database is queried with a set of input descriptors from a panorama keyframe. These input descriptors are created from 2D image features. The query delivers a set of correspondences between 3D finite map features and 2D image features. We iterate over the correspondences and check whether the panorama keyframe already has an inlier observation of the map feature. If not, we add a new observation.

Finally, we attempt to localize the panorama keyframe with its current 3D-2D correspondences. We retrieve all of the keyframes' finite features observations, resulting in a set of 3D-2D correspondences. The correspondences are passed to a RANSAC algorithm which employs a three-point-pose estimator [58]. The output pose is additionally refined with our robust relative pose estimator using the RANSAC inlier correspondences only. If the pose result is valid, we consider the panorama keyframe as robustly localized. Localized panorama keyframes are removed from their native local panorama map and are declared as reference of a new local panorama map.

4.4.2 Infinite feature triangulation

We triangulate infinite features which have observations in localized keyframe pairs with sufficient baseline. Since infinite features cannot be projected into keyframes outside of their native local panorama map, we apply descriptor matching to find relevant 2D-2D correspondences.

We maintain a second database (see Section 4.4.4) that keeps descriptors of infinite feature observations contained in localized panorama keyframes. After major modifications, *e.g.*, after descriptors from a new keyframe have been added, the tree is queried with all localized keyframes. For each keyframe, we create a input descriptor set that contains 2D image features that do not coincide with existing finite feature observations. We receive a set of 2D-2D correspondences between two localized keyframes. Using the difference pose, we run an epipolar point-line check and discard outliers. We triangulate the 3D points and check if they are in front of both cameras. Finally, we enforce a minimum triangulation angle to avoid spurious depth estimates.

The remaining correspondences are converted into finite map features. We add the matched observation from the second key-frame and assign the triangulated 3D position.

4.4.3 3D and panorama map consistency

We do not include non-localized panorama keyframes and infinite map features in bundle adjustment. Since the poses of localized keyframes may be updated, the registration of local panorama maps within the 3D map becomes incorrect. We correct the registration of local panorama maps and its infinite features.

The pose of local panorama maps is defined by their reference keyframes, which are localized within the 3D map and updated in bundle adjustment. For each local panorama map we compute the difference pose of its reference keyframe before and after bundle adjustment. We apply this difference pose to the dependent non-localized keyframes. The infinite feature rays are re-evaluated by transforming the observations into world space with the updated poses and computing the centroid.

4.4.4 Feature descriptor database

We use offline-trained hierarchical k-means trees for nearest neighborhood matching that contain PhonySIFT descriptors [175] in its leaves. The PhonySIFT descriptor has 36 elements computed from 3x3 subregions with four bins each. For each map feature observation, we compute descriptors on multiple image pyramid levels to increase scale invariance. The trees have a fixed structure (branching factor of 8, 4 levels, resulting in 4096 leaves) and are trained offline from a large set of images. The trees allow adding and removing descriptors efficiently and are synchronized with the map as part of the mapping process.

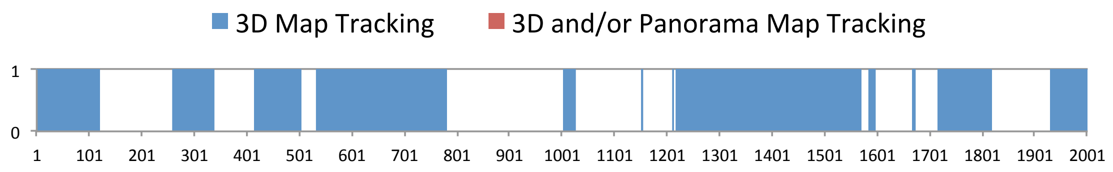
4.4.5 Map initialization

At system start-up, we employ a model-based detector and tracker [175] to create the initial map. Upon detection of a known planar image target, a first keyframe is created. The system continues to track the camera in 6DOF from the image target and, additionally, performs frame-to-frame tracking of 2D features. The second keyframe is selected as soon as sufficient 2D-2D correspondences can be robustly triangulated. Thus, two regular keyframes and the resulting finite map features constitute the initial map.

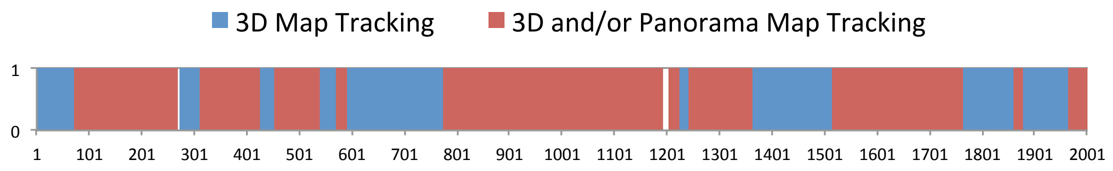
4.5 Results

The implementation of our method builds upon a state-of-the-art optimization-based 6DOF SLAM system that runs in real-time on modern mobile phones (see Section 4.5.1).

We compared our hybrid SLAM method with standard 6DOF SLAM on several image sequences. The two methods perform equal on image sequences that show general camera motion, such that the scene can be mapped with 6DOF keyframes only. This is no surprise, since tracking and mapping procedures are identical when operating with 6DOF keyframes and finite features only. Thus, to demonstrate the additional capabilities of our method,



(a)



(b)

Figure 4.5: Tracking status timelines of (a) 6DOF and (b) hybrid SLAM. Each timeline depicts the tracking state for each frame. Filled bars indicate successful tracking or relocalization, empty bars indicate tracking failure. For hybrid SLAM, we distinguish between camera tracking from only finite 3D map features (rendered in blue) and camera tracking from infinite panorama map features (rendered in red), including the hybrid case of camera tracking from both finite and infinite map features. For standard SLAM, the camera is tracked from finite features only.

we recorded a 2000-frame image sequence that shows several rotation-only camera pans that, in order to maintain tracking, require the selection of panorama keyframes.

The image sequence used for the comparison was recorded with a handheld camera and captures a well-textured room-sized indoor scene. The scene comprises a table-sized AR workspace in the foreground and the walls of the room in the background. We processed the image sequence with both hybrid and standard 6DOF SLAM methods and logged tracking and mapping statistics. The evaluation was performed on a laptop PC equipped with a quad-core 2.5GHz CPU and 8GB of RAM. We used a PointGrey Firefly MV handheld camera and recorded the images with a resolution of 640×480 pixels.

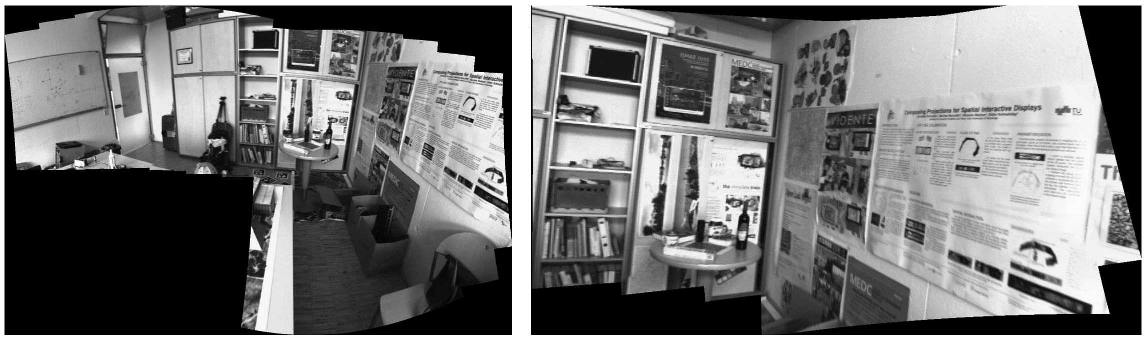
We present several timelines that depict tracking and mapping statistics for each frame of the image sequence. The timeline graphs do not consider the common map initialization phase and start with the first frame tracked from the initial 3D feature map having two 6DOF keyframes.

Tracking timeline. In Figure 4.5, we observe that the hybrid SLAM method tracked 98% of the frames, while the standard SLAM method only tracked 53% of the frames. Hybrid SLAM detected nine pure-rotation camera pans, resulting in an equal number of local panorama maps. While our method continues tracking from infinite features, standard SLAM tracking fails. For example, consider the period between frames 100 and 300: earlier, between frames 50 and 100, a pure-rotation camera movement started that was detected by hybrid SLAM around frame 80, resulting in the creation of a local panorama map. In contrast, standard SLAM discards keyframe candidates and runs out of map features around frame 110. The camera returns to the 3D mapped region around frame 270. Hybrid SLAM smoothly transitions from the local panorama map back onto the 3D map, while standard SLAM resumes finite map feature tracking after successful relocalization.

We conclude, that the detection of temporary pure-rotation camera movements and their mapping as local panorama maps improves the tracking performance of the hybrid SLAM method compared to the standard 6DOF SLAM method.

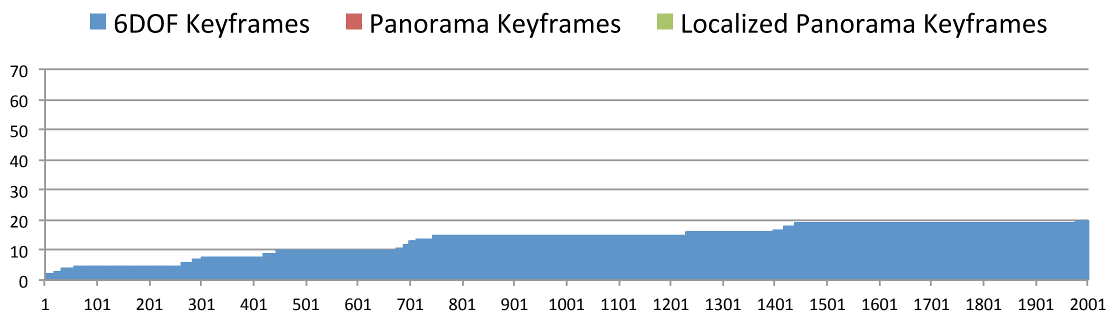
Keyframe timeline. In Figure 4.7 we observe that hybrid SLAM selects about three times as many keyframes as standard SLAM over the sequence. Even if we consider that standard SLAM only tracked half of the sequence and, thus, could have potentially selected twice as many keyframes, that is an increase of about one third (0.03 vs. 0.02 keyframes per successfully tracked frame). The increase mostly comes from additional panorama keyframes, while the number of 6DOF keyframes is even slightly lower.

We also see that hybrid SLAM quickly localizes panorama keyframes within the 3D map. The number of non-localized keyframes stays very low over the entire sequence.

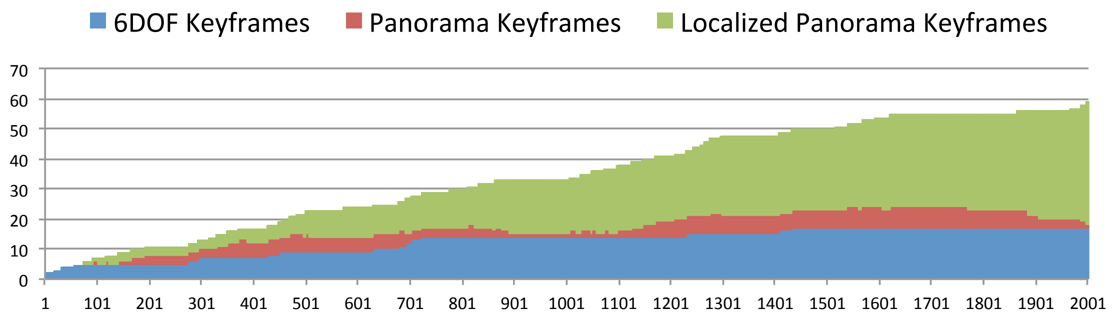


(a) **(b)**

Figure 4.6: Panoramas generated from keyframes of two local panorama maps. The local panorama map (a) was created in the period during frame 800 and 1200. The local panorama map (b) was created in the period during frame 1500 and 1800. Both panoramas depict the scene from two different view points with sufficient parallax for triangulation (*e.g.*, the wall on the right-hand side).



(a)



(b)

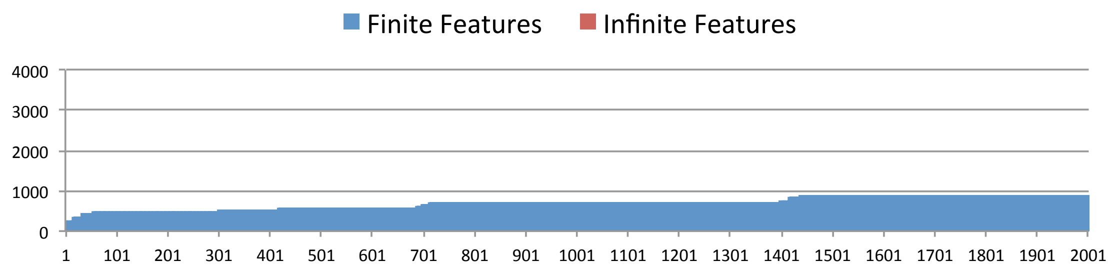
Figure 4.7: Keyframe timelines of (a) 6DOF and (b) hybrid SLAM. Each timeline shows the number of mapped keyframes over the sequence. For hybrid SLAM, we distinguish between 6DOF (blue), panorama (red) and localized panorama keyframes (green). Standard SLAM selects only full 6DOF keyframes in blue.

Furthermore, the localization of panorama keyframes enables the triangulation of infinite map features.

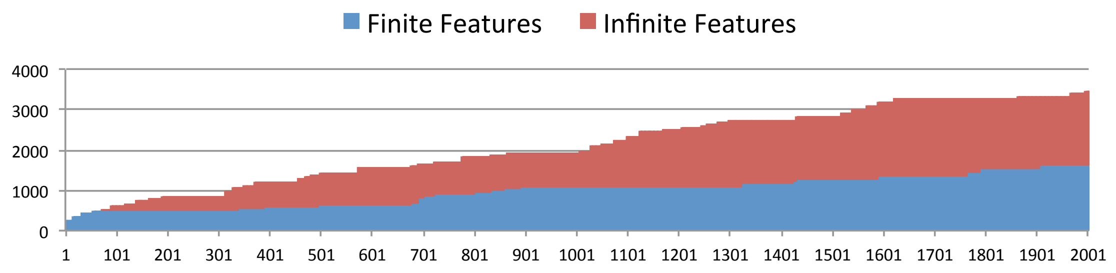
Map feature timeline. As we can see in Figure 4.8, hybrid SLAM maps contain about three to four times as much features as standard SLAM maps. Considering the finite map features only, hybrid SLAM maps are still two times larger than standard SLAM maps.

The ratio between finite and infinite features in hybrid SLAM maps is about 1:1. From Figures 4.8(b) and 4.8(c), we observe that about every third infinite feature was triangulated and, thus, contributed to 3D mapping. We assume some redundant infinite map features, since in our current implementation, we are not merging finite and infinite feature correspondences.

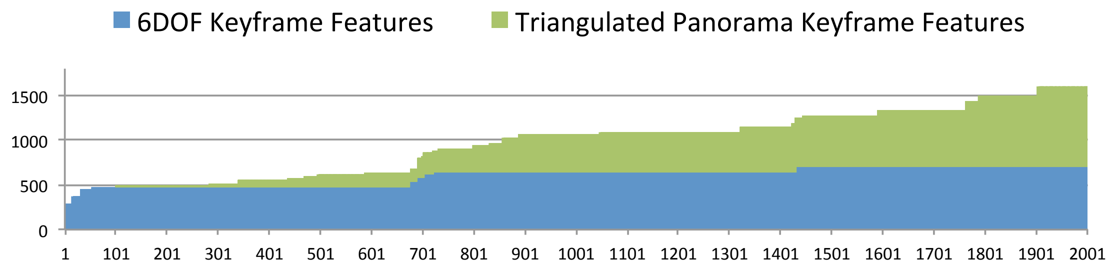
Figures 4.9 and 4.10 depict the reconstructed 3D maps of hybrid and standard SLAM,



(a)



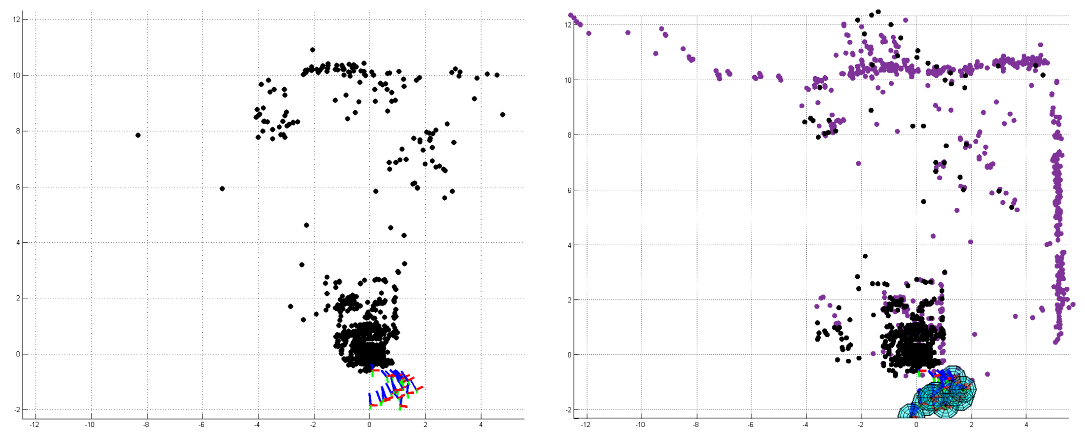
(b)



(c)

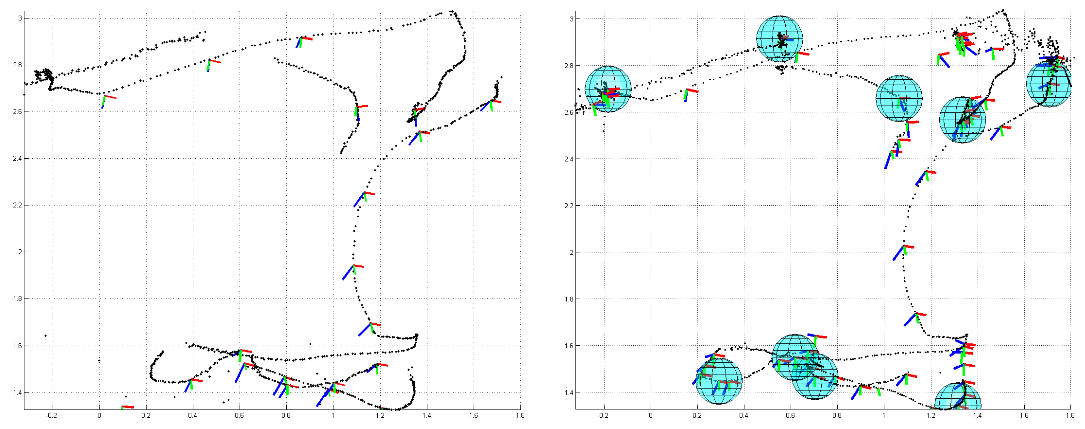
Figure 4.8: Map feature timelines of (a) 6DOF and (b) hybrid SLAM. Each graph shows the number of map features over the sequence. Standard SLAM maps consist of finite features only (blue). For hybrid SLAM, we distinguish between finite (blue) and infinite (red) map features. Additionally, in (c), finite map features are split into finite features triangulated from 6DOF keyframes (blue) and converted infinite features triangulated from panorama keyframes (green).

respectively. We see that the hybrid SLAM map reconstruction is considerably larger. More importantly, the hybrid mapping approach was able to reconstruct far more background detail, *e.g.*, the wall on the right is almost entirely reconstructed from infinite features, but not reconstructed at all by standard SLAM. Many of these infinite features have been triangulated with correspondences between keyframes two local panorama maps



(a) (b)

Figure 4.9: Final reconstructed 3D point feature maps of (a) 6DOF and (b) hybrid SLAM projected onto the XY-plane. Finite features triangulated from 6DOF keyframes are rendered in black, converted infinite features triangulated from panorama keyframes are rendered in purple. The keyframes are located south of the table that hosts the AR workspace.



(a) (b)

Figure 4.10: Reconstructed camera trajectories and keyframe locations of (a) 6DOF and (b) hybrid SLAM projected onto the XZ-plane. Local panorama maps of hybrid SLAM are rendered as spheres.

that have sufficient parallax, as can be seen in Figure 4.6.

Mapping time requirements. The increased map data volume and the additional operations for panorama keyframe localization and infinite feature triangulation require additional computational resources for mapping. Figure 4.11 shows the time requirements

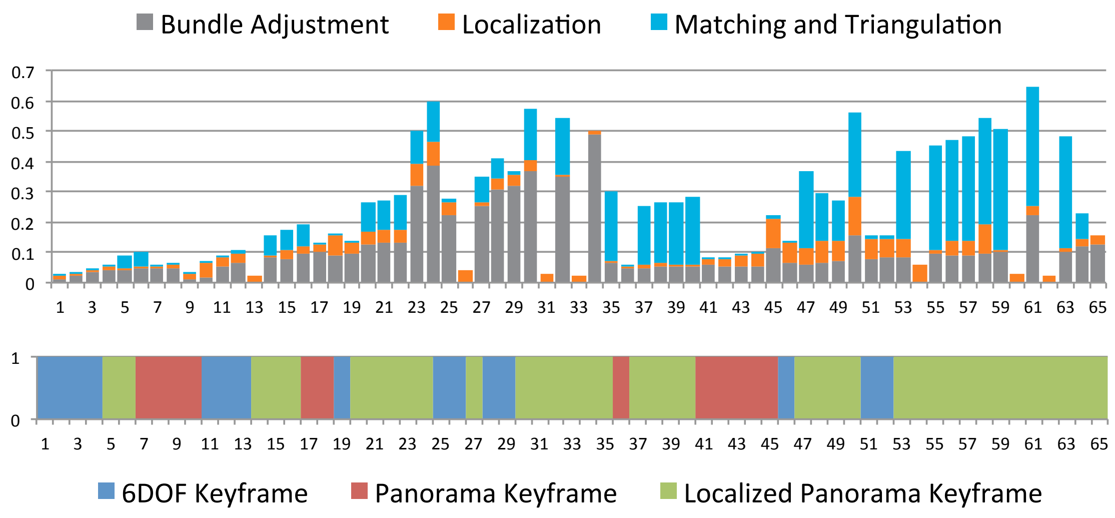


Figure 4.11: Major mapping tasks and its time requirements in seconds along the sequence. The tasks are triggered by diverse keyframe events. See text for further explanations.

of (a) bundle adjustment, (b) panorama keyframe localization and (c) infinite feature triangulation mapping tasks along the image sequence. Each timing slot refers to either a (1) 6DOF or (2) panorama or (3) localized panorama keyframe event that triggers the execution of the mapping thread. Depending on the keyframe event type, the mapping tasks behave differently. Currently running tasks may be interrupted by an upcoming high-priority keyframe event.

The effort for bundle adjustment increases quadratically with the number of localized keyframes. Non-localized panorama keyframes are not included in the optimization. Note that bundle adjustment is executed with a varying number of iterations. The effort for panorama keyframe localization depends on the number of non-localized panorama keyframes. The effort for infinite feature matching and triangulation increases linearly with the number of localized keyframes which are used to query the descriptor database. The number of query keyframes depends on the type of keyframe event. For a new 6DOF keyframe, the database is queried with this single keyframe only, while, for a new localized panorama keyframe, the database is updated with this keyframe and queried with all available localized keyframes. For new panorama keyframes, the database is not queried at all.

In our current implementation, the effort for infinite feature matching and triangulation sometimes exceeds the effort for bundle adjustment optimization. However, this can be improved with better mapping task scheduling and selection of input features for descriptor matching. We also consider a dedicated mapping task that removes redundant 6DOF and localized panorama keyframes with a similar method as described by Klein and Murray [74].

We conclude that the information contained in local panorama maps can be fruitfully used for 3D reconstruction resulting in SLAM maps with an increased number of 3D features. Furthermore, we see that the map is built quicker and covers a larger extent, both due to the possibility for delayed matching of infinite features. Larger and denser finite feature maps allow for continued and more robust camera tracking.

4.5.1 Mobile phone application

Our hybrid SLAM system runs in real-time on modern mobile phones. We used a Samsung Galaxy S2 equipped with a dual-core 1.2GHz ARM CPU and 2GB RAM running Android OS 4 for our tests. As expected, we found the overall robustness and performance restricted in comparison to the PC version. However, we applied the system in indoor and outdoor environments and created maps with about 100 keyframes and 4000 map features, including finite map features of 25 local panorama maps. Irrespective of the map size, tracking is mostly running with 20 to 30Hz. However, with increasing map sizes, we noticed congestion symptoms in the mapping thread, resulting in delayed map updates and, consequently, incremental pose tracking problems. Our implementation is not fully optimized yet, and we did not adjust all system parameters to the mobile phone platform. Figure 4.12 shows the mobile application handling a pure-rotation camera motion.

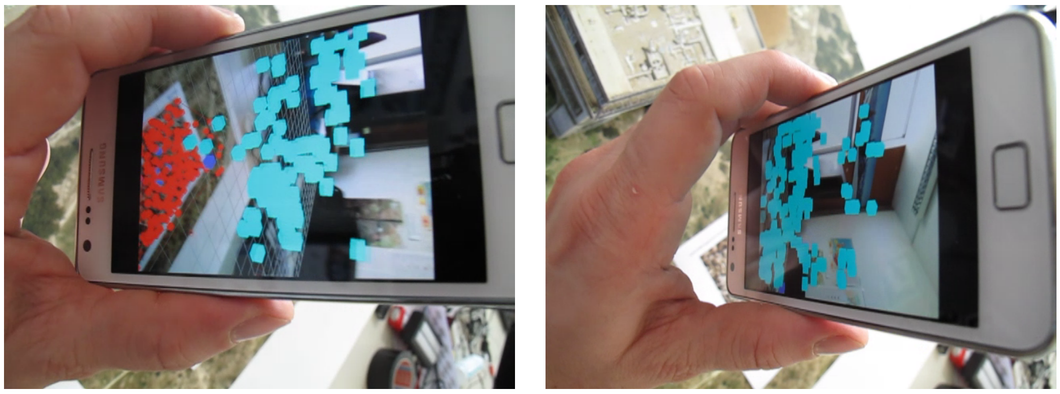
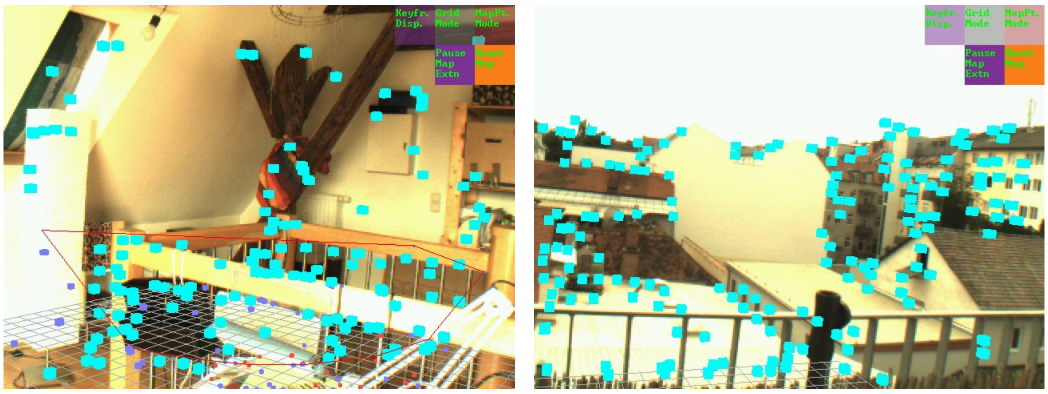


Figure 4.12: Hybrid SLAM system handling a pure-rotation camera movement in real-time on the mobile phone.



(a) **(b)**

Figure 4.13: Sample images of (a) indoor and (b) outdoor image sequences.

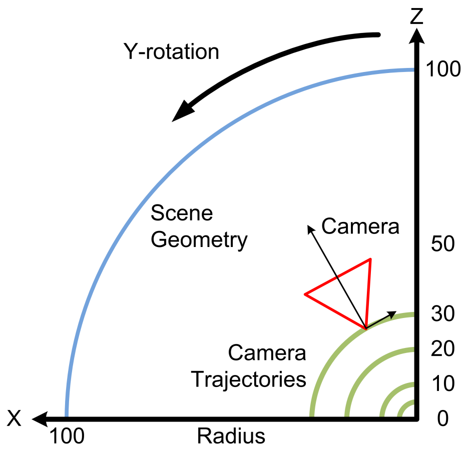
4.5.2 Panoramic SLAM accuracy and robustness

With the following experiment, we want to access the effects of camera translation onto the accuracy and robustness of 3DOF panoramic SLAM.

We created a virtual model that is used to render synthesized image sequences. The virtual model is depicted in Figure 4.14(a,b) and consists of a textured cylinder that is observed with a 64° FOV camera that moves along a circular trajectory with varying radius. With this model, we simulate the real world situation where users perform imperfect camera rotation movements. Starting with the ideal case of a camera rotating around its nodal point, we introduce more and more camera translation to challenge the panoramic SLAM method. The cylinder has a fixed diameter of 100cm and has its center in the coordinate system origin. The circular camera trajectories are located in the XZ-plane and have a varying radius $r \in \{0, 5, 10, 20, 30cm\}$, resulting in the ground truth camera poses $P_i = [R_i|(0, 0, -r)^t]$ with $R_1 = I$. We animated the camera to rotate 90° around the Y-axis. For each camera trajectory radius, we rendered a 100-frame image sequence.

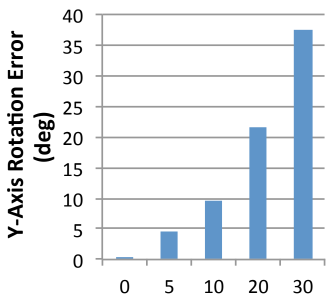
We processed the synthesized image sequences with our hybrid SLAM system. The system selects the first frame as panorama keyframe, assigns the initial pose $P_1 = [I|0]$, and initializes a map consisting of infinite features. The remaining images are processed in panoramic SLAM mode: Further keyframes are mapped, while the camera is tracked from infinite features, resulting in poses $P_i = [R_i|0]$ with 3DOF rotation matrices R_i, while the camera location stays fixed at the origin.

We recorded tracking and mapping statistics depicted in Figure 4.14(c, d, e). The hybrid SLAM system tracked and mapped all image sequences successfully. In Figure

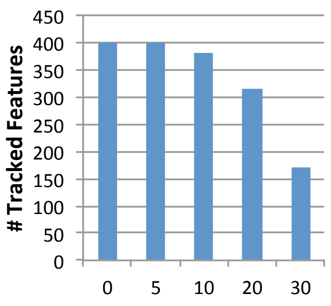


(a)

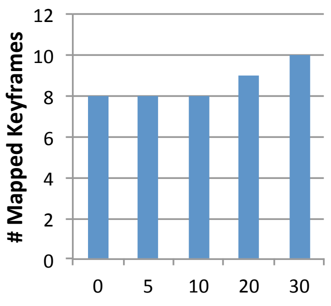
(b)



(c)



(d)



(e)

Figure 4.14: Effects of camera translation on panoramic SLAM accuracy and robustness: (a,b) virtual model used to generate synthesized image sequences. For each camera trajectory radius: (a) Y-axis rotation error in degrees, (b) number of successfully tracked map features, and (c) number of mapped keyframes.

4.14(c), we depict the Y-axis rotation error of the estimated camera pose of the final sequence image for all radii, *e.g.*, the pose estimated for the final 100$^{\text{th}}$ image frame with radius $r = 5$ has a relative error of about 5°. The camera translation encoded as additional rotation by the pose estimator results in a quadratically increasing error.

We observe in Figure 4.14(d) that the number of successfully tracked map features drops with increasing radius. That means that local active search fails for an increasing number of map features since their projected location in the current frame is too far away from their actual location.

Figure 4.14(e) shows the number of mapped keyframes, which increases with the radius. This behavior is a consequence of (c) in combination with the "view angle difference" keyframe selection criterion: With increasing translation, more virtual rotation is estimated, the threshold is reached quicker, and keyframes are selected earlier in the sequence.

We conclude that panoramic SLAM can handle a considerable amount of translation. In our system, we select panorama keyframes in a greedy manner, as our main priority is to maintain tracking. We accept erroneous poses resulting from camera translation, since panorama keyframes are localized in 6DOF anyway before being used for mapping.

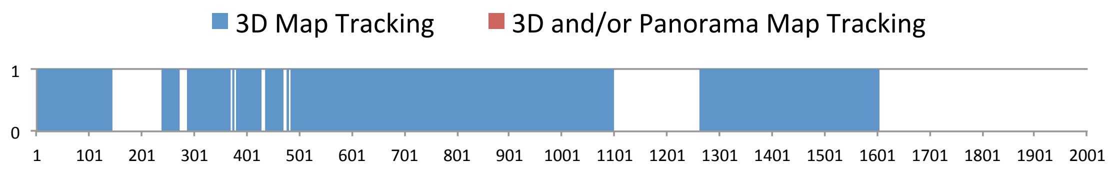
4.6 Discussion

In the following, we discuss certain issues (aspects, limitations) of our system and provide comparisons with the original PTAM system and our closest related work.

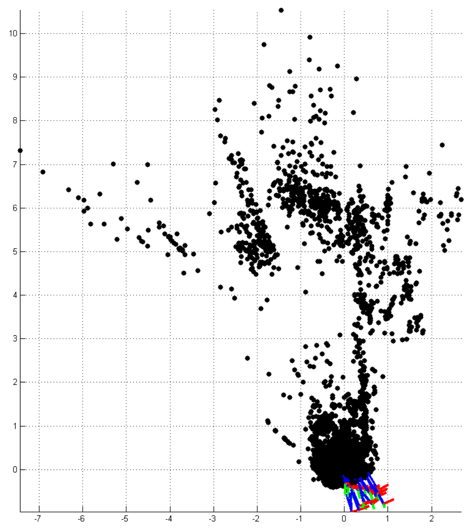
Our system combines 6DOF and panoramic SLAM methods and dynamically switches between these operation modes depending on the camera motion. While in panorama mode, we do not support transitions from pure rotation to general camera motion. This is due to the fact that panoramic SLAM only allows for tracking the camera pose as 3DOF rotation. Any camera translation is encoded as additional rotation, resulting in inaccurate poses. While panoramic SLAM shows quite some robustness against translation (see Section 4.5.2), there are cases where these inaccuracies result in tracking failure, *e.g.*, due to undetected camera loops. In rare cases, panorama maps may even become corrupted beyond recovery. While this does not harm the global 3D map, tracking requires to be reinitialized by relocalization with respect to the 3D map.

As described above, camera translation cannot be measured during 3DOF panoramic tracking. When returning from the panorama map onto the 3D map, the unrecognised camera translation results in 6DOF pose offsets. If these offsets are sufficiently large, local active search fails to match projected finite map features, resulting in tracking failure due to a lack of correspondences. In these cases, relocalization is required. Otherwise, we iteratively track mixed sets of finite and infinite map features and transition seamlessly.

Infinite feature matching and triangulation generates a moderate number of outliers. While we have not found these outliers severely disturbing our system, the system could be improved with a mapping task that removes outlier observations and features, *e.g.*, by verifying reprojection errors. Additionally, the runtime behaviour of our mapping method can be improved by revisiting descriptor matching and introducing redundant keyframe removal. The performance of feature matching can be improved with better selection of input features. Removing redundant keyframes and its map feature observations should reduce the computational complexity of bundle adjustment optimization as well as track-



(a)



(b)

Figure 4.15: Evaluating PTAM on the 2000-frame image sequence of Section 4.5: (a) tracking status timeline indicating tracking success/failure comparable to Figure 4.5, (b) final reconstructed 3D point feature map. See text for discussion.

ing.

4.6.1 Comparison with Klein *et al.*

We processed the 2000-frame image sequence from Section 4.5 with the publicly available PTAM software[1] described in the seminal paper of Klein *et al.* [72]. We modified the software to automatically select first and second keyframes and to log mapping and tracking statistics. The results are depicted in Figure 4.15.

From the tracking timeline in Figure 4.15(a), we see that PTAM tracks 63% of the frames successfully. The image sequence contains five major rotation movements as apparent in the tracking timeline of hybrid SLAM in Figure 4.5(b). PTAM fails to track two of the rotation movements completely, but manages to track three movements at least

[1]http://www.robots.ox.ac.uk/ gk/PTAM/

partially. Note that the comparison between PTAM and our SLAM system may not be entirely fair, since the tracking success criteria of PTAM is less strict than ours (*e.g.*, PTAM expects to track less map features to report tracking success).

The PTAM system selects 19 keyframes to reconstruct a map containing about 9000 point features. The initial map created from two keyframes already consists of about 5000 features. The PTAM method is well-known to gain its tracking robustness in large parts from assembling large amounts of map features. In particular, PTAM does not verify if landmarks have been mapped already, resulting in many redundant map features. The 3D map view in Figure 4.15(b) shows that PTAM triangulates many features despite little parallax, resulting in spurious depth estimates. For example, consider the scattered features along the camera view rays from bottom right to top left. Instead of unreliably triangulating features from keyframes with insufficient parallax, our hybrid SLAM system inserts these features as rays with infinite depth to local panorama maps.

Klein *et al.* [74] presented a heavily modified PTAM method that managed maps with dramatically less features and was thus running in real-time on the iPhone 3G. We argue that, even years later, the original PTAM method cannot be applied to modern mobile phones, regardless of their increased computational power. The goal must be to create high-quality maps with an optimized data volume that achieve similar robustness as PTAM. In this respect, we consider our system to be better suited for mobile phones. We provide two arguments: Firstly, comparing the reconstructed 3D maps of hybrid SLAM (Figure 4.9(b)) and PTAM (Figure 4.15(b)), we see that our system reconstructed a considerably larger portion of the scene with far less finite map features (1500 vs. 9000). Secondly, we evaluated the total time used for mapping tasks over the entire image sequence: PTAM uses about 36 seconds and thus about two times more than hybrid SLAM, which uses 18 seconds for mapping.

4.6.2 Comparison with Gauglitz *et al.*

In the following, we compare our method with Gauglitz *et al.* [53] - our closest related work. Both methods share the idea of supporting general and pure-rotation camera motion by combining 6DOF and panoramic SLAM methods. However, when looking at the details, the methods differ in many respects and appear as rather complementary approaches.

Mapping. The system of Gauglitz *et al.* handles arbitrary switches between general and pure-rotation camera motion, resulting in the creation of a new 3D/panorama submap with each context switch. Separate 3D submaps have distinct scales. Within a feature track, successive 3D and panorama maps are linked via a common keyframe. Upon tracking failure, a new feature track is started. Our system handles general camera motion alternated by temporary pure-rotation camera motions, resulting in local panorama maps that are registered within a single consistent 3D map. Given keyframe pairs with sufficient overlap exist, their system merges pairs of 3D-3D or panorama-panorama submaps. However, their system does not merge pairs of 3D-panorama submaps, while our system exploits the information contained in registered local panorama maps for 3D reconstruction.

Tracking. The system of Gauglitz *et al.* performs frame-to-frame tracking between the current frame and the latest keyframe of the current 3D or panorama map. From the resulting 2D-2D correspondences, multiple motion models are estimated. To get a global 6DOF camera pose, the existing keyframe poses are combined with a relative 6DOF pose converted from the motion models. The conversion of the motion models into a relative 6DOF pose is potentially ambiguous. In contrast, our system performs global 6DOF pose tracking by establishing 3D-2D correspondences between the current frame and the projected features of the optimized map, and robustly estimating a 6DOF pose. Thus, we additionally gain notion on whether the camera observes already mapped scene geometry, which allows for more fine-grained keyframe selection.

Keyframe selection. The system of Gauglitz *et al.* employs the generalized GRIC algorithm to distinguish homography/rotation and essential motion models, which may result in ambiguities, if (1) the scene is planar (2) the camera motion is not a true rotation. Our system employs the robustly tracked camera pose trajectory and scale-independent thresholds for parallax and camera view angles for the detection of pure-rotation camera motion.

Relocalization. The system of Gauglitz *et al.* performs delayed *loop-closing=recovery* relocalization, which actually refers to the merging of 3D-3D or panorama-panorama submap pairs that is done in the mapping backend. Our system performs immediate relocalization in the tracking frontend with respect to the optimized map.

Summary. The system of Gauglitz *et al.* aims at rapidly reconstructing the scene with potentially multiple submaps by continuously collecting image data in a visual-odometry-style tracking frontend. Our "classic" SLAM system aims at supporting interactive augmented reality applications by providing a persistent map coordinate system and 6DOF camera poses that allow to register and render virtual content on top of the reconstructed scene geometry.

4.7 Summary

The presented hybrid SLAM system demonstrates that the extension of a standard 6DOF optimization-based SLAM system with a dedicated panorama mode yields several improvements.

Most importantly, by employing a hybrid 3D map representation that contains finite as well as infinite features, the system is able to track camera motions and scene environments which cannot be handled by typical monocular keyframe-based SLAM systems. In particular, camera motions characterized as pure rotations and environments with large camera-scene distances can be tracked. Thus, we showed that the total tracking time can be substantially extended.

Employing a unified incremental map tracker, we obtain a robust and well-performing system, because it always has a known map to track from. Therefore, it can use active search combined with motion models, which has been demonstrated to work well under fast motions and difficult lighting situations. Similarly, Gauglitz *et al.* have extended their original work [53] with incremental map tracking capabilities, resulting in a considerably more accurate and stable system [54].

Furthermore, our local panorama maps created during pure-rotation motion phases can contribute substantially to a richer map. They allow estimating more points in less time in the near field of the camera as well as mapping background structure through wide baseline matching between these local panorama maps. This is enabled by making panorama keyframes available for matching against later incoming keyframes. This concept has been continued by Herrera *et al.* [61], who employ deferred triangulation of infinite features in background mapping as well as during pose tracking and keyframe selection.

Additionally, the local panorama maps represent high-level scene features that enable additional applications. For example, the local panorama maps could be rendered into wide field-of-view images employed as environment maps for the reconstruction of the local lighting conditions within the observed scene [34, 35] and, thus, enable photometric registration and visually coherent rendering.

Overall, we believe that the present hybrid approach is a valuable combination that naturally extends the design of optimization-based monocular SLAM systems. The approach contains ideas that have been taken on in follow-up works [54, 61]. In addition, these works tackle the major limitation of our system, that is measuring parallax-inducing camera motion during panoramic tracking of infinite features. Consequently, these algorithms build up and manage multiple, however, only topologically connected sub-maps having distinct scales.

In order to detect these transitions, Gauglitz *et al.* [54] employ finite *or* infinite features to perform model estimation and selection with generalized Geometric Robust Information Criterion (GRIC) [162], while Herrera *et al.* [61] developed a cost function for iterative pose estimation (as well as bundle adjustment) that takes both finite *and* infinite features into account, and additionally allows for observing infinite features moving along the epipolar line until they exhibit sufficient parallax for triangulation.

However, both of these algorithms are computationally expensive, such that our work remains the only system having "unconstrained" camera motion and scene structure capabilities, that has also been demonstrated to run in real-time on hand-held devices.

5 Urban Outdoor Localization and SLAM Initialization

In this chapter, we introduce a novel method for instant geo-localization of the video stream captured by a mobile device (see Figure 5.1). The first stage of our method registers the image to an untextured 2.5D map (2D building footprints + approximate building height), providing an accurately and globally aligned pose. This is done by first estimating the absolute camera orientation from straight-line segments, then, estimating the camera translation by segmenting the façades in the input image and matching them with those of the map. The resulting pose is suitable to initialize a SLAM system. The SLAM map is initialized by back-projecting the feature points onto synthetic depth images rendered from the augmented 2.5D map.

Our system has several major advantages over the state-of-the-art: First, the global localization component requires only OpenStreetMap-style data, which is widely available. Second, the global localization does not require searching through a large database and is, therefore, suitable for mobile devices with limited memory and computational capacity[1]. Third, the initialization of the SLAM system from the first frame avoids the need to tediously cover a sufficient outdoor baseline for stereo triangulation. Fourth, there is no restriction on the camera motion: Tracking and mapping are possible even in the case of purely rotational motion. This combination of features brings outdoor urban tracking a significant step closer to use in practice.

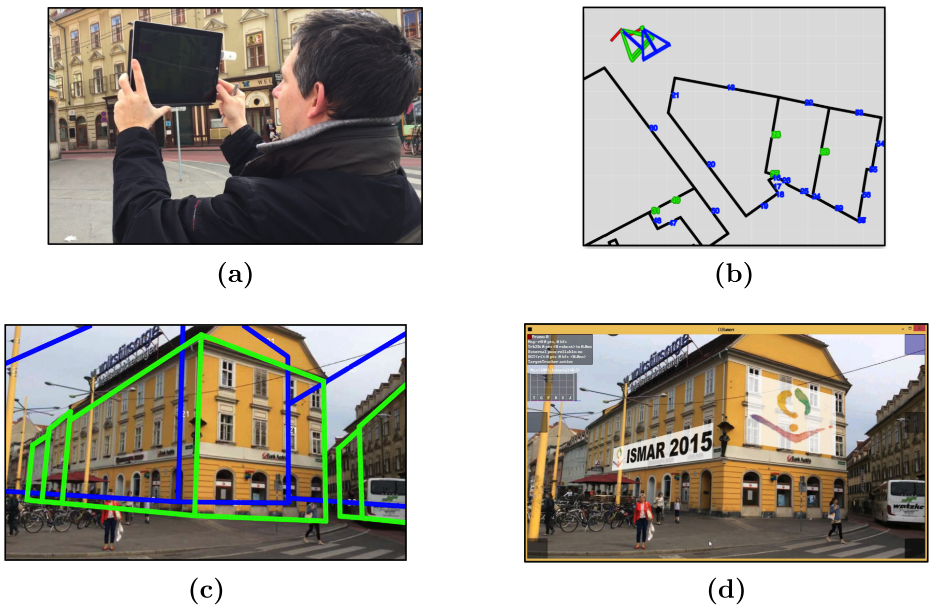


Figure 5.1: Outdoor urban usage of real-time SLAM with our novel localization technique. **(a)** User with a mobile device running our software. **(b)** Map view with sensor pose (**blue**) and the pose estimate from our method (**green**). **(c)** Reprojection of a globally aligned building model into the image using the sensor pose (**blue**) and the same reprojection after correction with our method (**green**). **(d)** Live camera view on the mobile device with globally aligned augmentations. We use the corrected pose to synthetically render a depth map from the building model and initialize a SLAM system, which starts tracking the camera motion. We can then instantly augment the scene with virtual elements. In contrast to previous systems, we do not need large translations for initializing the SLAM system, and we can handle both purely rotational and general 3D motions.

5.1 Method overview

As depicted in Fig. 5.2, our method first obtains a single image (camera) and a pose estimate from mobile sensors (GPS, compass, accelerometer), *i.e.*, the first keyframe acquired with a SLAM system running on a mobile device, plus a record of the built-in sensor values. From the sensor data, a first 6DOF pose estimate is compiled, using the fused compass and accelerometer input to provide a full 3×3 rotation matrix w.r.t. north/east and the earth center, and augmenting it with the WGS84 GPS information in metric Universal

[1]Note that our current unoptimized, single-threaded Matlab implementation of the global localization component is one order of magnitude away from real-time operation. However, the SLAM component runs at 30Hz, and we discuss later how the localization can be accelerated to meet real-time requirements.

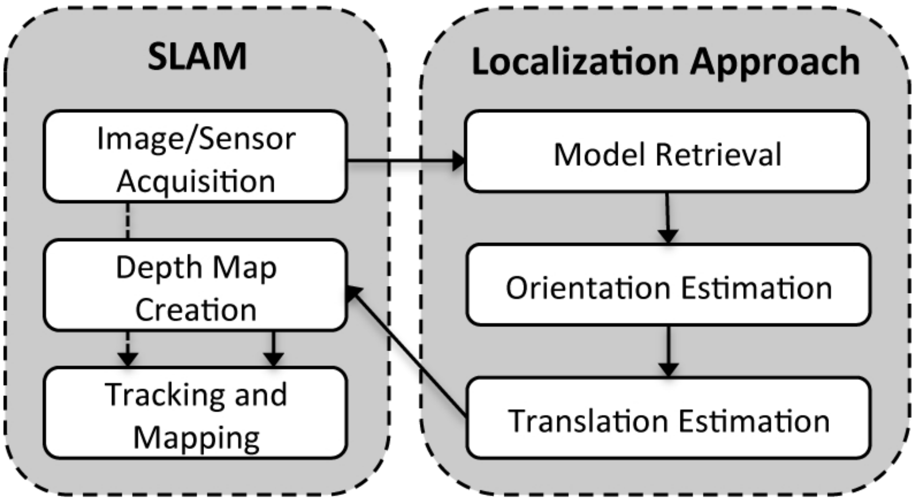


Figure 5.2: An overview of our method. The SLAM component provides image and sensor data to the localization module, returning an accurate pose estimate for depth map creation and SLAM initialization.

Transverse Mercator (UTM) coordinates to create a 3×4 pose matrix. This estimate is used to retrieve a 2.5D map containing the surrounding buildings, *i.e.*, using 2D building and height data from OpenStreetMap. Note that, however, that this pose can be off by $30°$ and 15 m, which is not accurate enough for SLAM initialization.

For our localization method, we assume that most image line segments extracted from the visible building façades are either horizontal or vertical. This is a common assumption used in vanishing point and relative orientation estimation, valid for many urban environments. Based on this assumption, we essentially solve a 2D-3D line correspondence problem. Three correct image-model correspondences allow for computing the 6DOF pose. However, our correspondence problem is profoundly non-trivial, since we find very little matching information in our input data. Additionally, the pose prior provided by the mobile sensors can be very noisy.

For the estimation of the global camera orientation (Section 5.2), we require a single correspondence between a horizontal image line and a model façade plane. This problem is robustly solved using minimal solvers in RANSAC frameworks.

For the subsequent estimation of the global 3D camera location (Section 5.3), we require two correspondences between vertical image lines and model façade outlines. This problem is tackled by first extracting potential vertical façade outlines in the image and matching them with corresponding model façade outlines, resulting in a sparse set of 3D location hypotheses. To improve the detection of potential vertical façade outlines in the image, we first apply a multi-scale window detector, before extracting the dominant vertical lines. We verify the set of pose hypotheses with an objective function that scores the match between a semantic segmentation of the input image and the reprojection of the 2.5D façade model. The semantic segmentation is computed with a fast light-weight multi-class support vector machine.

The resulting global 6DOF keyframe pose, together with the retrieved 2.5D model, is used by the SLAM system to initialize its 3D map (Section 5.4). We render a depth map and assign depth values to 2D keyframe features and thus initialize a 3D feature map. This procedure is repeated for subsequent keyframes to extend the 3D map, allowing for absolute 6DOF tracking of arbitrary camera motion in a global coordinate system.

5.2 Orientation estimation

We describe the estimation of the absolute orientation of the given camera image with respect to the 2D map. We start by computing the pitch and roll (Section 5.2.1), *i.e.*, the orientation of the vertical camera axis with respect to gravity, followed by computing the yaw (Section 5.2.2), *i.e.*, the remaining degree-of-freedom of the rotation in the absolute coordinate system of the 2D map.

5.2.1 Estimating the vertical axis

We want to estimate a rotation matrix $\mathbf{R}_v$ that aligns the camera's vertical axis with the gravity vector. We do so by determining the dominant vertical vanishing point in the image, using line segments extracted from the image. We rely on the Line Segment Detector (LSD) algorithm [56], followed by three filtering steps: (1) We only retain line segments exceeding a certain length. (2) Lines below the horizon line computed from the sensor rotation estimate are removed, since these segments are likely located on the ground plane or foreground object clutter. (3) Line segments are removed, if the angle between their projection and the gravity vector given by the accelerometer sensor is larger than a threshold [79].

The intersection point $\mathbf{p}$ of the projections $\mathbf{l}_1$ and $\mathbf{l}_2$ of two vertical lines is the vertical vanishing point. It can be computed with as a cross product using homogeneous coordinates:

$$\mathbf{p} = \mathbf{l}_1 \times \mathbf{l}_2 \ . \tag{5.1}$$

As suggested by Rother *et al.* [130], we search pairs of lines in order to find the dominant vanishing point. For each pair of vertical line segments, we compute the intersection point and test it against all line segments, using an angular error measure:

$$\mathrm{err}(\mathbf{p},\mathbf{l}) = \mathrm{acos}\left(\frac{\mathbf{p}\cdot\mathbf{l}}{||\mathbf{p}||\cdot||\mathbf{l}||}\right) \quad . \tag{5.2}$$

The dominant vertical vanishing point $\mathbf{p}_v$ is chosen as the one with the highest number of inliers using an error threshold of σ degrees, evaluated in a RANSAC framework.

Given the dominant vertical vanishing point $\mathbf{p}_v$, we compute the rotation which aligns the camera's vertical axis with the vertical vanishing point of the 2D map. The vertical direction of the 2D map is assumed as $\mathbf{z} = [0\ 0\ 1]^\top$. Besides that, no other information from the 2D map is needed.

Using angle-axis representation, the axis of the rotation is $\mathbf{u} = \mathbf{p}_v \times \mathbf{z}$, and the angle is $\theta = \mathrm{acos}(\mathbf{p}_v \cdot \mathbf{z})$, assuming that the vertical vanishing point is normalized. The rotation $\mathbf{R}_v$ can be constructed using $SO(3)$ exponentiation:

$$\mathbf{R}_v = \exp_{SO(3)}\left(\mathbf{u}\cdot\frac{\theta}{||\mathbf{u}||}\right) \quad . \tag{5.3}$$

5.2.2 Orientation in absolute coordinates

At this point, the camera orientation is determined up to a rotation around its vertical axis (yaw). In this section, we explain how to estimate this last degree of freedom for the orientation in the absolute coordinate system of the 2D map. The procedure is illustrated in Fig. 5.3: The image line segments are rotated and and back-projected onto an extruded 2D model. The optimal rotation $\mathbf{R}$ makes the back-projections as vertical and horizontal as possible. Note that for estimating $\mathbf{R}$, we do not require the building façade heights, but only rely on a 2D map that provides oriented line strips which allow for retrieving the building façade normals.

Given a façade f from the 2D map, its horizontal vanishing point is found as the cross product of its normal $\mathbf{n}_f$ and the vertical axis $\mathbf{z}$:

$$\mathbf{p}_h = \mathbf{n}_f \times \mathbf{z} \quad . \tag{5.4}$$

After orientation correction through $\mathbf{R}_v$, the horizontal image lines $\mathbf{l}$ lying on f should intersect $\mathbf{p}_h$. Thus, given a horizontal vanishing point $\mathbf{p}_h$ and a rotated horizontal line segment $\mathbf{l}_3 = \mathbf{R}_v^\top \mathbf{l}$, we can compute the rotation $\mathbf{R}_h$ about the vertical axis to align the

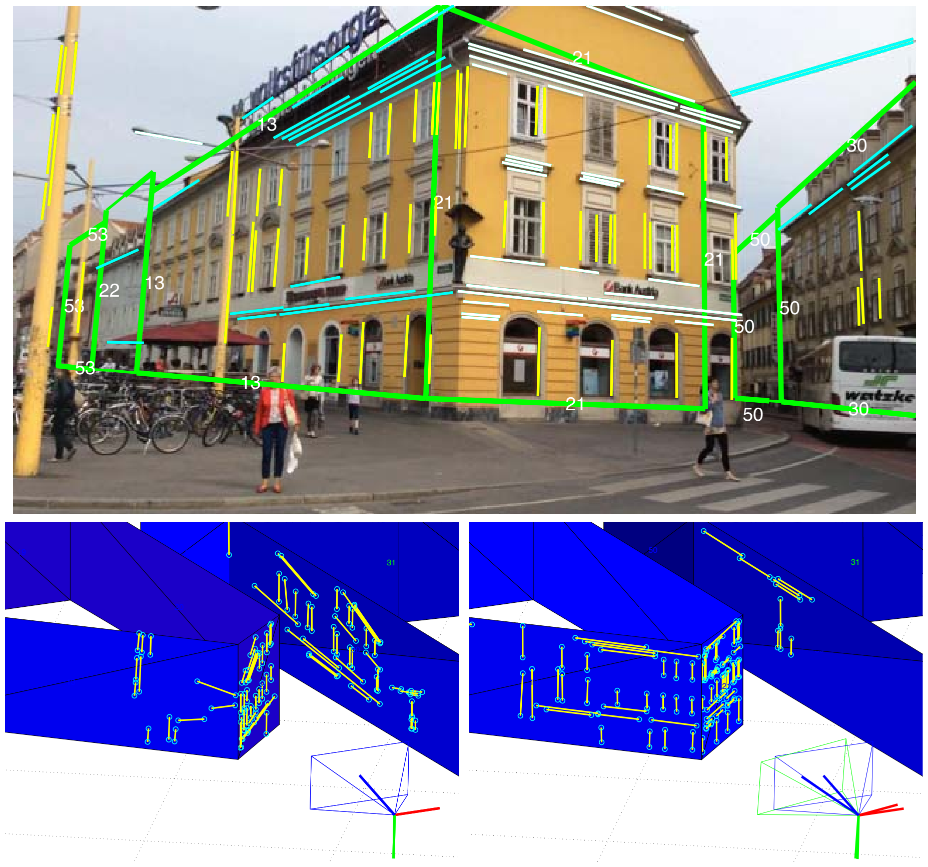


Figure 5.3: Estimating rotation $\mathbf{R}$. **Top:** line segments identified as vertical (yellow) and horizontal (cyan, white) in 3D compared to the reprojection (green) of the model after rotation correction, but before translation correction. **Bottom left:** Back-projection of the line segments using the sensor pose. **Bottom right:** Back-projection after rotation correction, which aligns the vertical and horizontal line segments with the façade model.

camera's horizontal axis with the horizontal vanishing point of f. This rotation has one degree of freedom, ϕ_z, the amount of rotation about the vertical axis:

$$\mathbf{R}_h = \begin{bmatrix} \cos\phi_z & -\sin\phi_z & 0 \\ \sin\phi_z & \cos\phi_z & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{5.5}$$

Using the substitution $q = \tan\frac{\phi_z}{2}$, we get $\cos\phi_z = \frac{1-q^2}{1+q^2}$ and $\sin\phi_z = \frac{2q}{1+q^2}$ [76]. We can parameterize our rotation matrix in terms of q:

$$\mathbf{R}_h = \frac{1}{1+q^2}\begin{bmatrix} 1-q^2 & -2q & 0 \\ 2q & 1-q^2 & 0 \\ 0 & 0 & 1+q^2 \end{bmatrix} . \tag{5.6}$$

The intersection constraint between $\mathbf{l}_3$ and the horizontal vanishing point $\mathbf{p}_h$ is expressed as

$$\mathbf{p}_h \cdot (\mathbf{R}_h \mathbf{l}_3) = 0 . \tag{5.7}$$

The roots of this quadratic polynomial in q determine two possible rotations. This ambiguity is resolved by choosing the rotation which best aligns the camera's view vector to the inverse normal $-\mathbf{n}_f$.

Finally, the absolute rotation $\mathbf{R}$ of the camera is computed by chaining the two previous rotations $\mathbf{R}_v$ and $\mathbf{R}_h$:

$$\mathbf{R} = \mathbf{R}_v \mathbf{R}_h . \tag{5.8}$$

In practice we create pairs $< \mathbf{l}, f >$ from line segments $\mathbf{l}$ assigned to visible façades f, identified from the 2D map using the initial pose estimate from the sensors. We use a Binary Space Partition (BSP) tree for efficient search the 2D map for visible façades – a BSP tree is a data structure from Computer Graphics to efficiently solve visibility problems [52]. We evaluate the angular error measure from Eq. (5.2) for a rotation estimate from the pair $< \mathbf{l}, f >$ in a RANSAC framework, choosing the hypothesis with the highest number of inliers.

We have to consider the following case of $< \mathbf{l}, f >$ pairs, where $\mathbf{l}$ is actually located on a *perpendicular* façade plane $f_\perp$, resulting in rotation hypotheses $\mathbf{R}$ which are 90° off the ground truth. Given a visible façade set where all façades are pairwise perpendicular, such a rotation hypothesis may receive the highest number of inliers. We discard such $< \mathbf{l}, f_\perp >$ pairs by computing the angular difference between the sensor pose and the rotation hypothesis $\mathbf{R}$ and discard the hypothesis, if it exceeds a threshold of 45°.

The case of $< \mathbf{l}, f >$ pairs where $\mathbf{l}$ is actually located on a *parallel* façade $f_\|$ does not cause any problems, because in this case $f_\|$ and f have the same horizontal vanishing point $\mathbf{p}_h$.

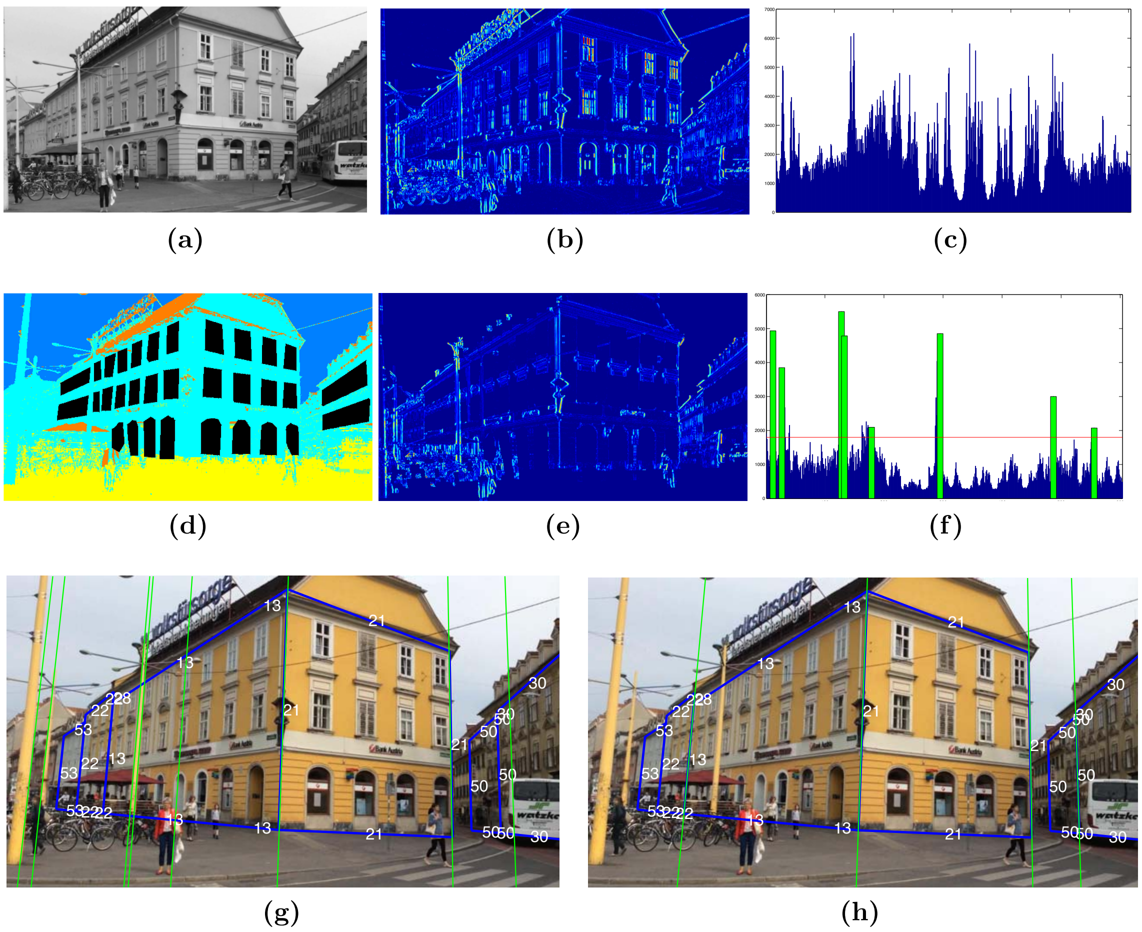


Figure 5.4: Generating translation hypotheses. **(a)**: Vertically rectified image. **(b)**: Image gradients. **(c)**: Histogram of the sums of the gradient magnitude over the columns. **(d)**: Segmentation of the façades in cyan and window detections in black. **(e)**: Image gradients only for the pixels lying on a façade, but not on a window. **(f)**: Histogram of gradient sums and selected vertical image lines. **(g)**: Selected image lines overlaid on the original image. **(h)**: 3D model lines from building corners overlaid on the original image using the ground truth pose. Most of the visible building outlines were successfully detected with our method.

5.3 Translation estimation

The vertical and horizontal segments on the façades allow estimation of the camera's orientation in a global coordinate system. Unfortunately, the segments do not provide any useful constraint to estimate the translation, since we do not know their exact 3D location. In theory, the pose could be computed from correspondences between the edges

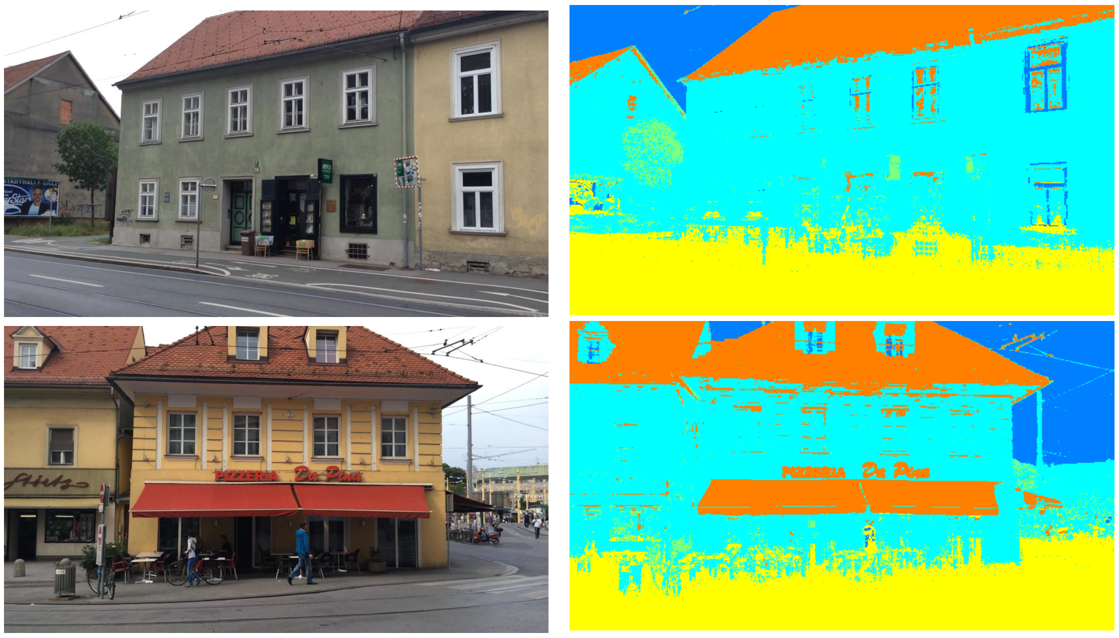


Figure 5.5: Pixel-wise segmentations obtained with a multi-class SVM for two different images. Cyan corresponds to façades, blue to sky, orange to roofs, green to vegetation, and yellow to ground plane.

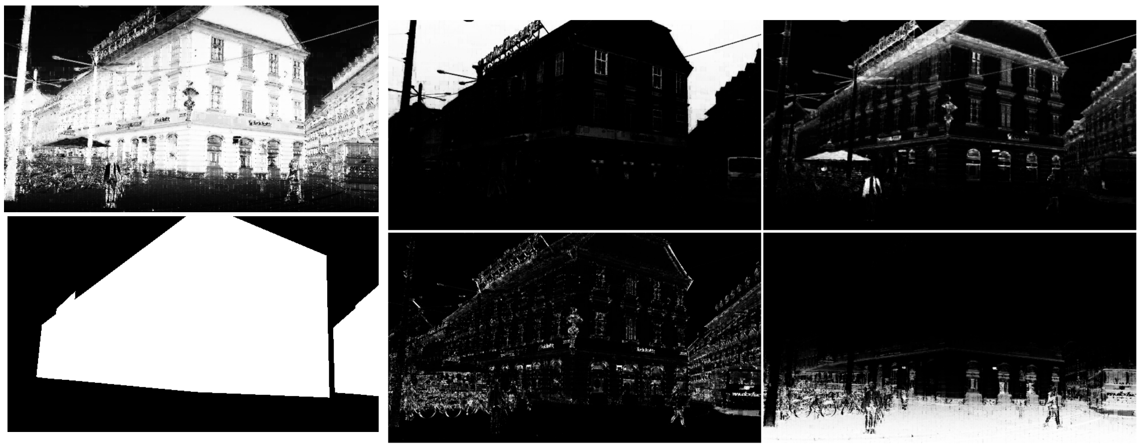


Figure 5.6: Computing the log-likelihood. **Left:** Probability map for c_f, the façade class. **Middle:** Probability maps for c_s, c_r, c_v and c_g. **Right:** Reprojection $\text{Proj}(M, \mathbf{p})$ for a pose close to the ground truth.

of the buildings in the 2D map and their reprojections in the images. In practice, it is virtually impossible to directly obtain such matches reliably in absence of additional information.

We approach this problem by aligning the 2D map with a semantic segmentation of the image. We can estimate the translation of the camera as the one that aligns the façades of the 2D map with the façades extracted from the image.

To speed up this alignment and to make it more reliable, we first generate a small set of possible translations, given the line segments in the image that potentially correspond to the edges of the buildings in the 2D map. We keep the hypotheses that best aligns the 2D map with the segmentation. We give details on these two steps below.

5.3.1 Generating translation hypotheses

In practice, the translation along the vertical axis is the most problematic one to estimate from the image, because the bottoms of the buildings are typically occluded by foreground clutter (cars, pedestrians). We, therefore, simply set the height of the camera at 1.6 m, which is reasonable for a handheld device.

For the remaining two degrees of translational freedom, we generate possible horizontal translations (parallel to the ground plane) for the camera by matching the edges of the buildings with the image. However, this is a very challenging task, as the images are very cluttered in practice.

As shown in Fig. 5.4, we generate a set of possible image locations for the edges of the buildings with a heuristic. We first rectify the input image using the orientation, so that vertical 3D lines also appear vertical in the image, and we sum the image gradients along each column.

The columns with a large sum are likely corresponding to the border of a building. However, since windows also have strong vertical edges, they tend to generate many wrong hypotheses. To reduce their influence, we trained a multi-scale window detector based on the work of Viola and Jones [172] on the ZuBud building database [140]. Only almost frontally viewed windows were manually extracted and used for training, resulting in a set of 1170 positive images. As negatives, a set of private images from travels and parties was used. The resulting detector has 28 stages with a total of 651 features. Despite this simple procedure, the detector works reasonably well, still leaving a lot of room for optimization, both in terms of overall cascade depth as well as the selection of negative imagery given the expected urban streetview domain (see *e.g.*, [67]).

Pixels lying on the windows found by the detector are ignored, when computing the gradient sums over the columns. We also use the façade segmentation result described in Section 5.3.2 to consider only the pixels that lie on façades, but not on windows. Since the sums may take very different values for different scenes, we use a threshold estimated automatically for each image. We fit a Gamma distribution to the histogram of the sums and evaluate the quantile function with a fixed inlier probability.

Finally, as shown in Fig. 5.4(g) and Fig. 5.4(h), we generate translation hypotheses for each possible pair of correspondences between the vertical lines extracted from the image and the building outlines. The building outlines come from the corners in the 2D maps that are likely to be visible, given the location provided by the GPS and the orientation estimated during the first step, again using the BSP tree for efficient retrieval. Given two vertical lines in the image, $\mathbf{l}_1$ and $\mathbf{l}_2$, and two 3D points which are the corresponding building corners, $\mathbf{x}_1$ and $\mathbf{x}_2$, the camera translation $\mathbf{t}$ in the ground plane can be easily computed by solving the following linear system:

$$\left\{ \begin{array}{rcl} \mathbf{l}_1 \cdot (\mathbf{x}_1 + \mathbf{t}) & = & 0 \\ \mathbf{l}_2 \cdot (\mathbf{x}_2 + \mathbf{t}) & = & 0 \end{array} \right. . \tag{5.9}$$

We filter the hypotheses set based on their estimated 3D location. First, hypotheses which have a location outside of a sphere whose radius is determined by the assumed GPS error of 12.5 m [184] are discarded. Second, we remove hypotheses which are located within buildings.

5.3.2 Aligning the 2.5D map with the image

To select the best translation among the ones generated using the method described above, we evaluate the alignment of the image and the 2.5D map after projection, using each generated translation.

We use a simple pixel-wise segmentation of the input image, by applying a classifier to each image patch of a given size to assign a class label to the center location of the patch.

The segmentation uses a multi-class Support Vector Machine (SVM) [138, 22], trained on a dataset of images from a different source than the one used in our evaluations, manually segmenting the images using the *LabelMe* service [132]. We use the integral features introduced by Dollar *et al.* [36], and consider five different classes $C = \{c_f, c_s, c_r, c_v, c_g\}$ for *façade, sky, roof, vegetation* and *ground*, respectively. By applying the classifier exhaustively, we obtain a probability estimate p for each image pixel over these classes. Fig. 5.5 shows an example of a segmentation for a typical input image.

As illustrated in Fig. 5.6, given the 2D projection Proj(M, $\mathbf{p}$) of our 2D map+height

M into the image using pose hypothesis $\mathbf{p}$, we compute the log-likelihood of the pose:

$$s_{\mathbf{p}} = \sum_{i}^{\text{Proj}(M,\,\mathbf{p})} \log p_i(c_f) + \sum_{i}^{\neg\text{Proj}(M,\,\mathbf{p})} \log\left(1 - p_i(c_f)\right) \ , \tag{5.10}$$

where $\neg\text{Proj}(M,\ \mathbf{p})$ denotes the set of pixels lying outside the reprojection $\text{Proj}(M,\ \mathbf{p})$. The pixels lying on the projection $\text{Proj}(M,\ \mathbf{p})$ of the façades should have a high probability to be on a façade in the image, and the pixels lying outside should have a high probability to not be on a façade. We keep the pose $\hat{\mathbf{p}}$ that maximizes the log-likelihood:

$$\hat{\mathbf{p}} = \arg\max_{\mathbf{p}} s_{\mathbf{p}} \ . \tag{5.11}$$

In practice, the 3D location estimated from the sensors is often not accurate enough to directly initialize our method. We therefore sample six additional initial locations around the sensor pose in a hexagonal layout, and combine the locations with the previously estimated orientation. We execute our method initialized from each of these seven poses, searching within a sphere having 12.5 m radius [184] for each initial pose. Thus, our method can find the correct image location within a region of up to $40x40$ m. Finally, we keep the computed pose with the largest likelihood.

Note that this approach naturally extends to more complex building models, for example, if the roofs of the buildings are also present in the model. The log-likelihood then becomes:

$$s_{\mathbf{p}} = \sum_{c\in C_M} \sum_{i}^{\text{Proj}(M_c,\,\mathbf{p})} \log p_i(c) + \sum_{i}^{\neg\text{Proj}(M,\,\mathbf{p})} \log\left(1 - \sum_{c\in C_M} p_i(c)\right) \ , \tag{5.12}$$

where C_M is a subset of C and made of the different classes that can appear in the buildings model, and $\text{Proj}(M_c,\ \mathbf{p})$ is the projection of the components of the buildings model for class c.

Much more sophisticated methods could be used [142, 44], but we have empirically verified that the camera translation is reliably computed, despite the relatively limited quality of our segmentation.

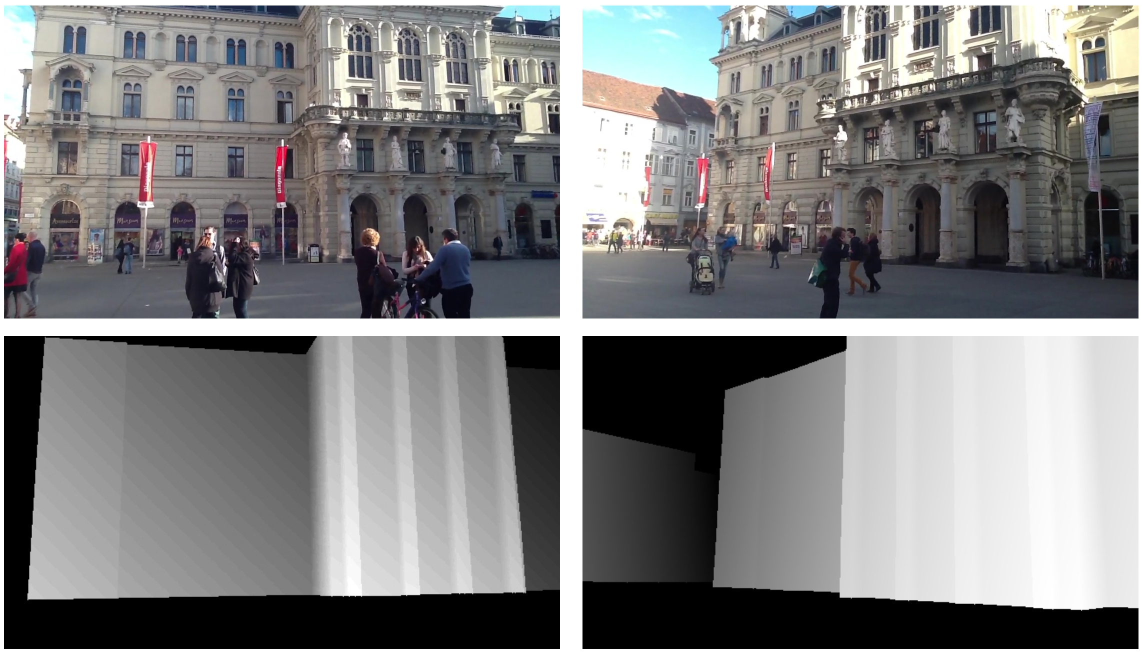


Figure 5.7: Top row: Two keyframes from the globally aligned SLAM system on a test sequence. **Bottom row:** Corresponding depth images rendered from the actual camera pose.

5.4 SLAM initialization using depth from 2.5D models

Recent SLAM systems for indoor use often rely on depth sensors for superior robustness and instant initialization. These sensors are not available outdoors, but we can use the 2.5D map to generate and use *synthetic* depth images as a cue for mapping and tracking.

We use a keyframe-based SLAM system, similar to PTAM [72]. The tracking and the mapping thread run asynchronously and periodically exchange keyframe and map information. Our localization approach registers the first keyframe to the 2.5D map. Using this pose estimate, we render a polygonal model using the graphics hardware and retrieve the depth buffer to assign depth to those map points which correspond to observed façades. We arrive at a full 3D map already for the first keyframe. Previous approaches required establishing a baseline of several meters between the first two keyframes for initial triangulation [169].

As the SLAM system acquires more keyframes, the procedure is repeated, and tracked map points collect multiple observations for real triangulation, once the baseline between keyframes is sufficient. Fig. 5.7 shows two keyframes of a test sequence and the corresponding depth images.

5.5 Experimental results

In this section, we first describe the dataset we built to evaluate our localization approach, and then report and discuss the results of the evaluation. Finally, we demonstrate its application to globally-aligned instant urban outdoor SLAM.

5.5.1 Dataset

To demonstrate the applicability of our approach, we captured a dataset of 32 images with an *Apple iPad Air* in urban and suburban environments of Graz, Austria.

The images were captured without any special consideration for satellite shadowing or surrounding metallic structures. As a consequence, the accuracy of the pose estimated with the sensors only ranges from very accurate, about 0.4 m position and $2°$ rotation error, to very poor, up to 16.5 m position and about $30°$ rotation error. Since altitude estimates from sensors tend to be very poor, we overrode these estimates of the poses predicted by the sensors with a default value of 1.6 m. For each test image, we calculated a ground truth pose by manually matching 2D image locations with 3D points from the maps.

We retrieved 2D maps of the surroundings from OpenStreetMap and extruded them with a coarse estimate of the height of the building façades. OpenStreetMap data consists of oriented line strips, which we converted into a triangle mesh including face normals. Each building façade plane is modeled as 2D quad with four vertices, two ground plane vertices and two roof vertices. The heights of the vertices were taken from aerial laser scan data. All vertical building outlines were aligned with the global vertical up-vector.

5.5.2 Orientation and Translation accuracy

Fig. 5.8, top, plots the angular error of the camera pose predicted by the sensors and after correction with our method. The error is calculated as the angular difference between the estimated rotation and the ground-truth rotation in angle-axis representation. We ranked the images from the one with the largest error after correction to the one with the smallest error. The sensor error can become as large as $30°$. With our method, all our orientation estimates have an angle error below $5°$, with the exception of a single outlier image which contains very few horizontal lines. 90.6% of the estimates are below $3°$, 84.4% below $2°$ and 50% below $1°$ of angular error w.r.t. the ground truth rotation.

Fig. 5.8, bottom, gives the results of our translation estimation method. As for the

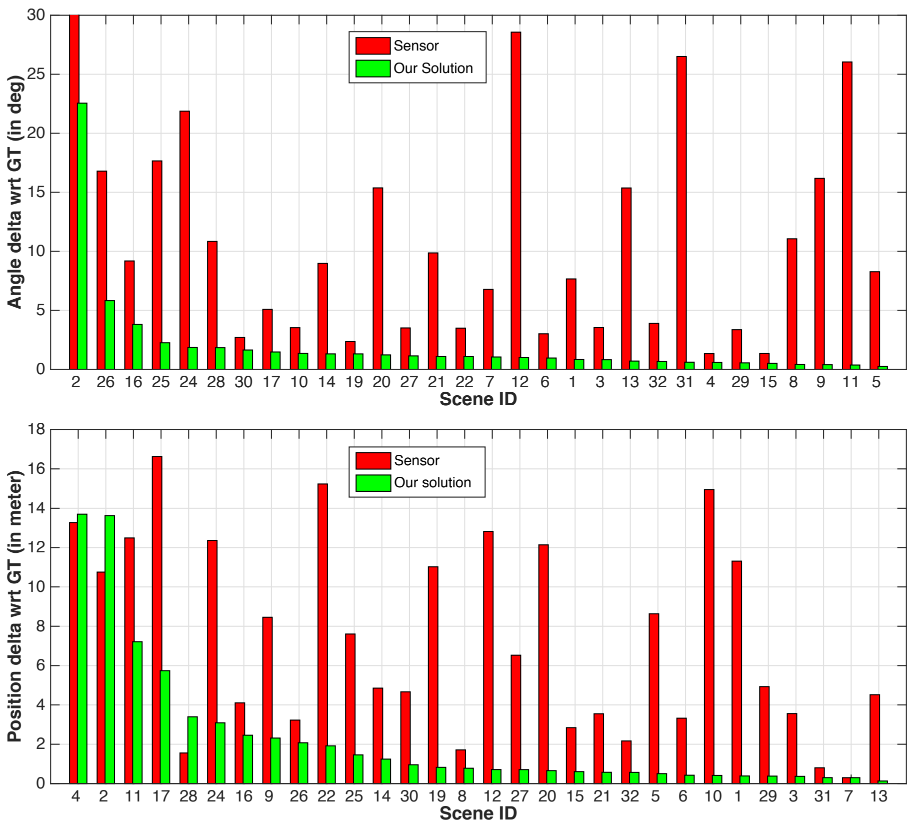


Figure 5.8: Pose estimates accuracy. **Top:** Orientation, and **Bottom:** Translation. We ranked the images from the one with the largest error after correction to the one with the smallest error. Our method significantly improves the accuracy of the orientation and translation estimates.

rotation, we ranked the images from the one with the largest error after correction to the one with the smallest error. The sensor errors range from about 0.4 m to about 16.5 m, with an average error of about 8 m. Our method significantly decreases the translation error in most of the cases. The worst results are due to adjacent buildings, with edges that cannot be extracted correctly. Overall, our method is able to considerably improve the position estimates from the sensors, with the pose estimates for 87.5% of the images being below 4 m, 68.8% below 2 m and 59.4% below 1 m of error w.r.t. the ground truth position.

5.5.3 Visual inspection

Fig. 5.9 presents the final results of our algorithm for a wide variety of test images, showing the reprojection of the model using both the sensor pose and the pose retrieved from our

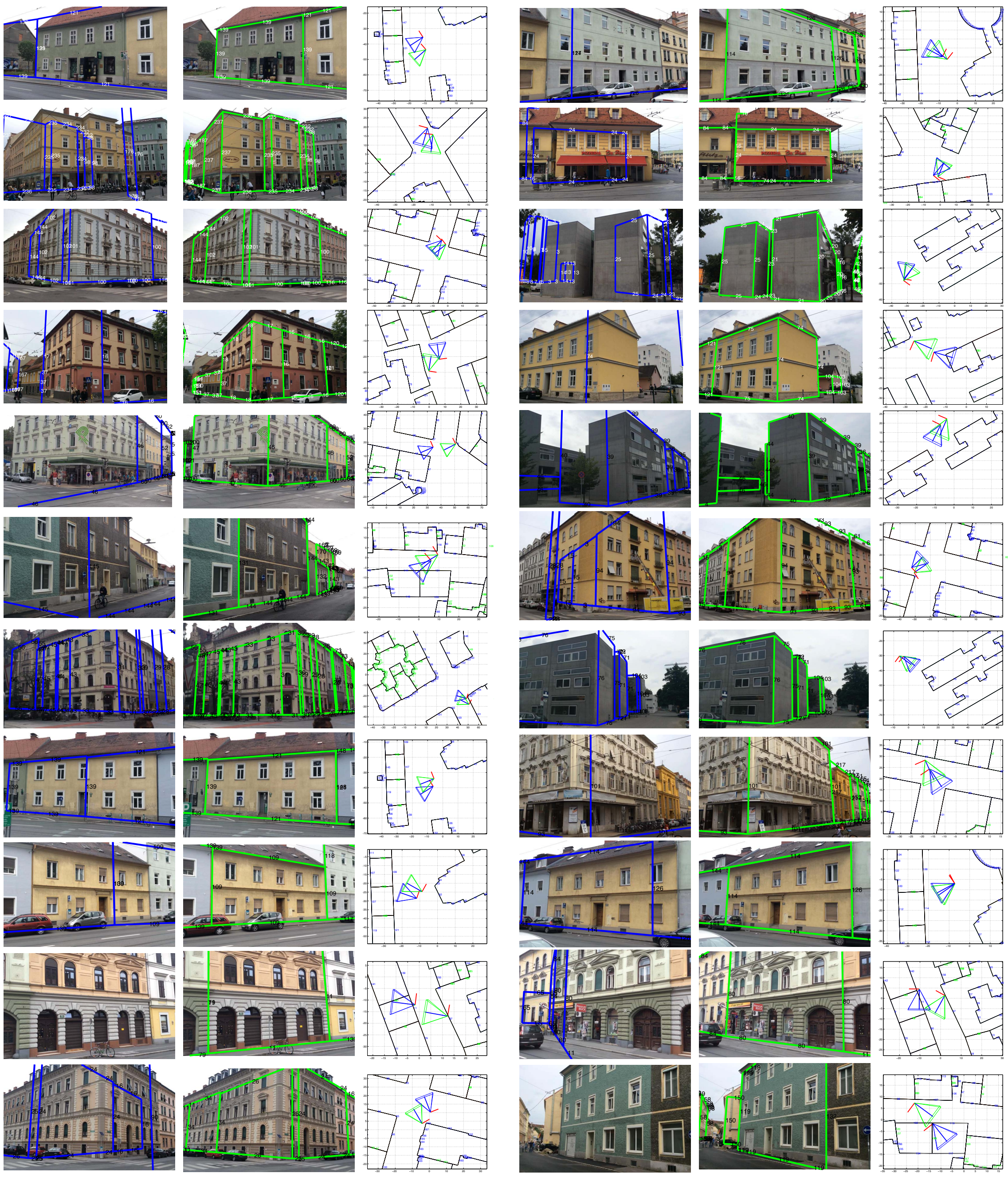


Figure 5.9: Results of our approach on test images. For each triplet of images: **Left:** Model reprojection into the image using the initial sensor pose. **Middle:** Model reprojection into the image using the final estimated pose. **Right:** Map view, sensor pose (in blue) and the corrected pose (in green).

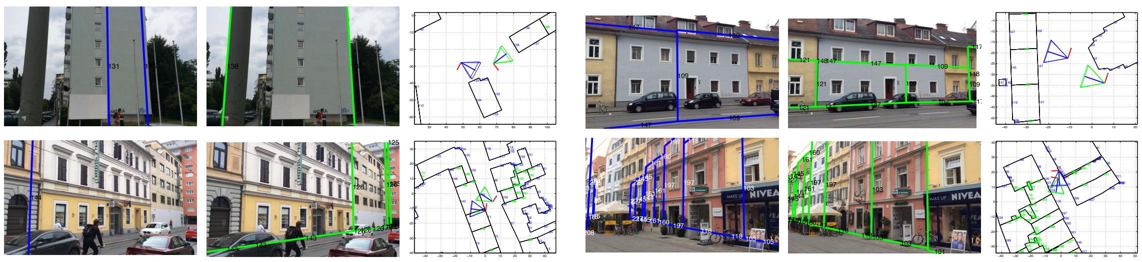


Figure 5.10: Images with the largest pose errors. For each triplet of images: **Left:** Model reprojection into the image using the initial sensor pose. **Middle:** Model reprojection into the image using the final estimated pose. **Right:** Map view, sensor pose (in blue) and the corrected pose (in green). Even for these images, the model reprojection tends to be close to the expected position.

approach. After pose estimation, the outlines of the models nicely fit the building outlines, even for very challenging scenes with many façades visible and a considerable rotation and position error in the sensor estimate. The amount of correction can be assessed from the map view, as both the rotation and translation undergo a significant correction during the application of our method.

Fig. 5.10 shows the images with the largest pose errors. In the upper left scene, the algorithm is fooled by a street lamp, in the upper right scenario the classification result is bad around the window areas, finally resulting in a wrong estimate of the scene distance. In the lower left scenario, a model of the background building is missing. In the lower right case, the classification result and the model line selection is bad, causing the translation estimation to fail.

5.5.4 Unconstrained SLAM using depth images

We evaluated the integration of the single image geo-localization for initializing our SLAM system on multiple sequences, featuring both rotation-only and general camera motion.

Our new approach has noticeable benefits over conventional SLAM in urban outdoor environments, as it provides accurate 6DOF localization right from the start. Usability is considerably improved, in particular, for panoramic camera motion, which users can hardly avoid [53, 119]. A direct comparison to Ventura *et al.* [169] with frames of a SLAM sequence is given in Fig. 5.11. In the previous approach, it takes more than 12 seconds for the system to successfully initialize, while our approach provides a globally accurate 6DOF pose for SLAM starting from the first captured image. Note that the estimated camera trajectory is considerably smoother, because the 3D locations of feature points are

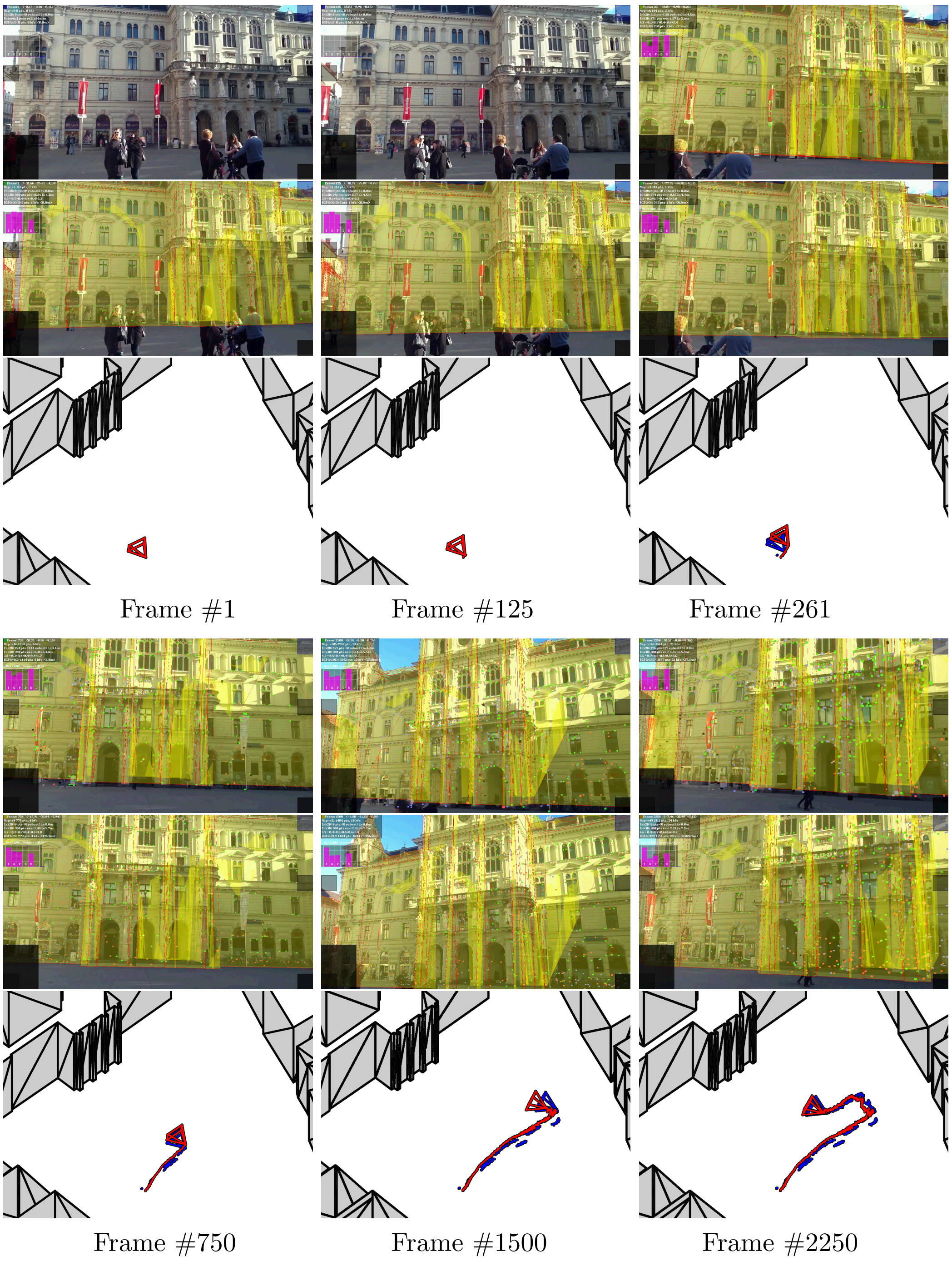


Figure 5.11: Comparison of previous work and our approach, together with the estimated camera trajectories. **Top row:** Results from previous work [169]. Due to the required baseline between keyframes, the system initializes after about 12 seconds. **Middle row:** Results from our approach. **Bottom row:** Trajectories estimated by previous work (blue) and our approach (red). Note that the trajectory estimated by our new approach is considerably smoother.

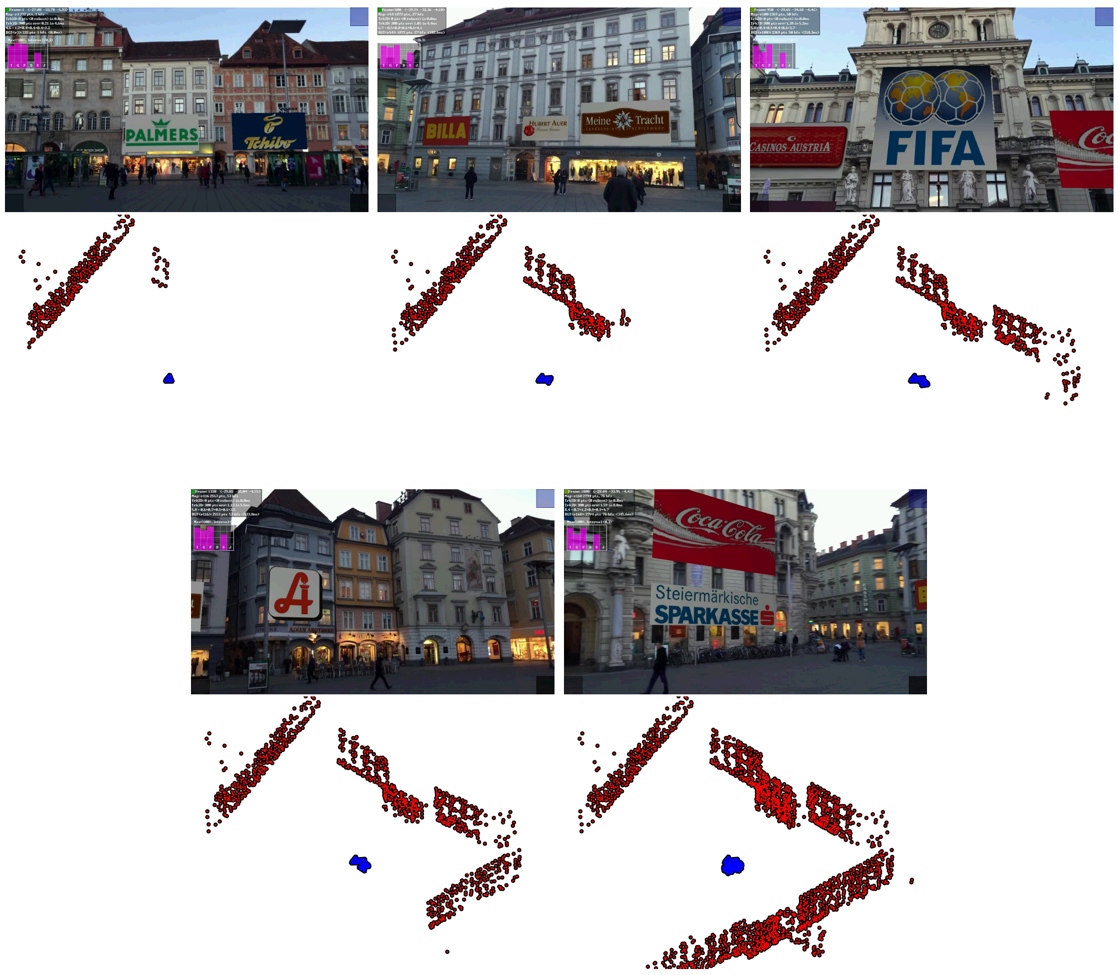


Figure 5.12: Top row: Annotation results from a rotation-only sequence. **Bottom row:** Expanding SLAM map, while the user rotates a handheld device. The map nicely resembles the structure of the surrounding buildings.

constrained by projecting them onto the façades.

Fig. 5.12 shows results of augmentations for a rotation-only sequence together with the evolving SLAM map over time as the user rotates the device. Note that the planar labels nicely align with the real building façades, and the recovered 3D SLAM maps resemble the surrounding building structure quite accurately.

5.6 Discussion

In the following, we critically discuss our results with respect to the current status of our framework and potential improvements.

5.6.1 Localization method

Our localization method gives good results on a large number of diverse scenes. The method is very versatile, since it only sets one strict requirement onto an input image: the visibility of two vertical building façade outlines – meaning that a façade may also be truncated towards the sky and/or towards the ground plane. Furthermore, we can cope with considerable sensor pose errors, potentially as much as 45° rotation offset and up to 40 m of position offset.

We identified the main reasons for a complete failure of our method to be cases where the pose prior from sensors is unusable – either, because it is inside a building, or, because it is misplaced in an a incorrect street segment, where the correct building façades cannot be observed. A simple improvement for the former case is to move the pose prior to the nearest street location, but this heuristic is not guaranteed to solve the problem in all cases.

The main reason for errors in our orientation estimation is an insufficient number of suitable vertical or horizontal lines found in an image to allow for robust vanishing point estimation. Tweaking the line detection parameters might partially overcome this problem for certain scenes.

Errors in our translation estimation are mainly due to the inability of our algorithm to detect the correct façade outlines for various reasons, *e.g.*, because the orientation estimate is bad, adjacent façades look too similar, the façade texture contains too much clutter or the window detector does not properly filter out windows. Similarly, the scoring function might select the wrong hypothesis, because of a bad semantic segmentation result or a bad reprojection result owed to insufficient model detail. All individual steps might undergo additional tuning to improve the overall performance of the approach.

Currently, the pose computation is implemented as unoptimized, single-threaded Matlab code and requires about 30 seconds per frame on a single CPU core on a 2011 Intel 2.5 GHz i5 Macbook Pro for a 640×360 image. The execution time for the individual major parts of the algorithm are given in Table 5.1. However, each of these steps have been demonstrated at much higher speed with optimized code and a GPU. For example, most of the time for the window detection step is used for image warping. Using the GPU would provide a significant speedup.

Part	Algorithm	Approx. Time [s]
(i)	Window detection	10
(ii)	Segmentation	14
(iii)	Translation estimation	6

Table 5.1: Timings of major parts of our unoptimized localization procedure running mainly in Matlab on a single CPU core on a 2011 Intel 2.5 GHz i5 MacBook Pro for a 640×360 image

5.6.2 SLAM

The SLAM system is very robust, runs in real-time on current mobile devices and works sufficiently well even under difficult lighting conditions outdoors. Currently, the system uses a single localization result to initialize the 3D map and to maintain it over time through rendering new keyframes and extending the map with new 3D points. However, even a small rotational error (< 1 degree) or a position error below $25cm$ accumulated through drift in the system over time might create a noticeable offset in the visualization of annotations in the distance. Therefore, the use of multiple localization results over time and the inclusion of more advanced methods in the iterative optimization of the system (*i.e.*, bundle adjustment) would improve the overall system robustness and accuracy considerably. We consider this a future work topic.

5.6.3 AR content and models

Our entire approach is designed to only take into account the minimum information one might hope for to be available anywhere globally: a 2D map and some building height information. Naturally, with more detailed and accurate models, even better results could be achieved. Also note Eq. (5.12), where we mentioned the natural extension of our approach in terms of semantic information in the segmentation stage.

In AR, this raises an important point where synergies can be exploited: As there needs to be annotated content to be visualized, this information can, in turn, feed back into the localization approach to improve localization performance. For example, using the AR annotations of windows or doors can be used in connection to our window detector to add another semantic class to the scoring function. We therefore argue that certain AR content might itself be used to improve localization performance within our framework, although this content is largely missing at this point in time.

5.7 Summary

We have presented a novel approach for accurate and efficient 6D pose geo-localization and mapping which relies only on components readily available for urban outdoor AR: open-source 2.5D maps as well as motion and camera sensors, which are built into modern hand-held devices. Given a single narrow field-of-view image, our method considerably improves the coarse 6D pose priors retrieved from motion sensors (GPS, compass, accelerometer) using computer vision techniques. In particular, we are exploiting vertical and horizontal line segments located on planar building façades. Essentially, our method only requires the input image to depict two – possibly truncated – vertical building façade outlines, and can correct as much as $45°$ orientation and up to $40\ m$ position offsets. Our approach, therefore, offers new opportunities for making AR practical in outdoor urban environments.

One major advantage of our method is the independence from reference image databases and related problems. The collection and maintenance of image databases is expensive and cumbersome. Community images are usually only available for certain popular regions within a city, such as touristic hotspots. Additionally, image-based geo-localization approaches suffer from appearance discrepancies between query and reference images. In contrast, the untextured 2.5D maps required by our method are available for almost all countries and city regions and can be retrieved from geographic information services such as the community-driven OpenStreetMap project. While OpenStreetMap at the moment provides building height annotations for certain regions only, we believe that the availability will steadily increase. Alternatively, the 2D cadastral maps can be combined with aerial laser scan data, provided by public local authorities and private vendors. Usually, these geographic information services adapt their maps very quickly to changes in the urban environment, *e.g.*, due to construction activity.

We provide a 6D pose exploitable by SLAM applications for 3D map initialization in the global coordinate system of the 2.5D model. Furthermore, we support the SLAM system in mapping and tracking arbitrary camera motion and structure by providing depth maps rendered from the 2.5D model, effectively synthesizing a wide-range depth camera. Thus, the SLAM system can track and map pure-rotation camera motion, and reconstruct structure which could otherwise not be triangulated due to insufficient parallax. Since the 3D maps are registered within a global coordinate system, such as UTM, AR applications can easily query location bases services, and display geo-referenced information as well as enable end-user content authoring. Additionally, we may integrate the building façade planes into the 3D SLAM map and provide a high-level scene representation.

In contrast to other localization methods using untextured models, we believe that our method has the potential to provide accurate geo-localization close to real-time, even on computationally restricted mobile platforms. Besides offering evidence in this respect, the SLAM geo-localization can be improved the following way: the geo-localization method may run in a dedicated thread and periodically provide global 6D pose updates for certain frames. These updates can be used to adopt and refine the 3D map. Additionally, the depth-map based mapping method should be more closely integrated with the native SLAM mapping. Native SLAM mapping may reconstruct foreground scene regions which exhibit sufficient parallax, while depth maps are employed to reconstruct the distant background. The resulting foreground/background segmented 3D reconstructions may also allow for generating interesting virtual visualizations.

6 Conclusions

6.1 Summary

In this thesis, we presented a set of techniques which make visual real-time localization and mapping more user-friendly and practical. We investigated several severe problems and limitations of general monocular SLAM algorithms. In particular, we investigated the mapping and tracking of arbitrary camera motion and scene structure, the initialization of 3D SLAM maps with respect to metric global coordinate systems, and the mapping of high-level scene features for global map optimization and scene understanding. In developing SLAM algorithms that tackle these problems, we exploited specific combinations of constraints on the camera motion performed by the user, as well as constraints on geometry and semantics of the observed scene structure.

We approached the mapping and tracking of arbitrary camera motion and scene structure with two techniques depending on the environment knowledge. First, in unknown environments, we modeled hybrid maps that contained features with finite or infinite depth. Second, in partially modeled urban outdoor environments, we simulated a wide-range depth camera by rendering synthesized depth images from a given untextured 2.5D city model. Both techniques tackle the parallax problem that stems from pure-rotation camera motion or scenes with large camera-object distances. Neither provide sufficient parallax for robust triangulation of 3D map features. Using synthesized depth images, the required 3D feature locations can be directly computed. Using hybrid feature maps, we combined general SLAM and constrained panoramic SLAM methods into a hybrid method: features with exhibit insufficient parallax were mapped with infinite depth into local panorama maps on the plane at infinity, while features which exhibit sufficient parallax were mapped with finite depth in 3D. Additionally, we were constantly seeking to repeatedly observe infinite features, until they exhibit sufficient parallax for deferred triangulation. Our hybrid maps allowed for tracking arbitrary camera motion in full 6DOF. Moreover, our tracking algorithm allowed for smooth transitions between 3D and local panorama maps by estimating the 6D pose from both features with finite and infinite depth. In practice, we found that, by handling parallax-free camera movements and applying deferred triangulation, we could considerably extend the successful camera tracking time and thus the usability of AR applications. Deferred triangulation, *e.g.*, using wide-baseline matching between local panorama maps, led to the reconstruction of additional scene regions in 3D.

We approached the initialization of 3D SLAM maps in urban outdoor environments with a geo-localization technique suited for hand-held devices. Given a single image and a coarse initial 6D pose as input, our geo-localization method employed computer vision techniques to estimate a refined 6D pose with respect the global coordinate system provided by an untextured 2.5D model of the urban neighborhood. The untextured 2.5D model, composed of the 2D city map and corresponding building height information, was retrieved from the community-driven OpenStreetMap geographic information service. Our geo-localization algorithm exploited constraints on the geometric and semantic scene structure. In particular, we employed horizontal and vertical line features exhibited by planar building façades for 3DOF orientation estimation by matching vanishing points on the plane at infinity. Furthermore, we segmented the input image into semantic classes such as “façade”, “sky” and “ground”, and employed this additional layer of information to formulate an objective function for 3D position estimation. Our geo-localization algorithm was found very versatilely applicable and efficiently performing, having the potential to run near real-time even on hand-held devices, as the individual components could be implemented using CPU and GPU parallelization. Finally, the resulting globally aligned 6DOF pose was employed to initialize the 3D map of a SLAM system, using the synthesized depth map rendering technique described above.

We approached the mapping of high-level scene features by exploiting homography constraints. Homographies provide strong constraints on both camera motion and scene structure, which we employed to implement a very efficient mapping algorithm, running at least a magnitude faster than bundle adjustment. Especially suited for SLAM on lo-fi hand-held devices, this algorithm allowed for mapping and tracking unmodeled planar scenes, which we found indoors and outdoors on numerous human-made structures. In addition, the reconstructed planes could be rendered as ortho-images and reused by other model-based detection and tracking systems. Similarly, our hybrid SLAM approach allowed for rendering panoramic mosaics, which could be used for photometric reconstruction, *i.e.*, estimation of the local lighting situation. Both, planar and panoramic maps provided high-level features, which increased the scene understanding of AR applications.

Extent of World Knowledge With respect to Milgram and Koshino's Extent of World Knowledge (EWK) continuum [102], our contributions increased both the geometric and semantic world knowledge in partially modeled as well as in unmodeled environments. Starting in unmodeled environments, our SLAM algorithms reconstructed both parallax-inducing and parallax-free scene structure and camera motion into hybrid, panoramic and planar models. The models provided scene understanding to AR applications and could be exported and reused ("What"). In turn, the models could be used for online and offline camera localization ("Where"). Starting in partially modeled environments, we employed untextured 2.5D models for geo-localization in urban outdoor environments ("Where"), and initialized 3D SLAM maps with the resulting globally aligned 6D pose ("What"). The 3D SLAM maps allowed for continuous tracking of arbitrary camera motion ("Where"). Overall, we believe that our contributions shifted the "What" and "Where" markers alongside the EWK continuum and increased the knowledge about the world that, in turn, could be fruitfully utilized by mobile AR applications.

Mobile augmented reality We want reconsider the mobile AR application scenario we introduced earlier to discuss our achievements and improvements:

> A user is sitting in a hotel lobby of a foreign city, browsing the news on his smartphone, and discovers an exhibition she wants to visit. Leaving the hotel towards the museum, she makes use of 2D map navigation to orient herself. At an ambiguous junction, however, she chooses to directly superimpose the correct walking directions over streets and buildings. Arriving at the museum, ticket and digital exhibition catalog can be purchased online with a few clicks. Moments later, she stands in front of an interesting piece and augments the object with information about its making and history in a perspectively correct manner via her device. After the visit, she can similarly visualize and explore a virtual version of the piece in an arbitrary environment, *e.g.*, at the airport gate waiting for her plane, or back home in her living room. In the public space at the airport gate, the visualization may be unregistered with the environment,

providing a purely virtual view. In the private space of her living room, the visualization may be spatially registered, providing an augmented view.

The user investigates a museum exhibition piece in a-priori unknown environments. We presented hybrid SLAM algorithms, which allow for real-time tracking and mapping of arbitrary camera motion and scene structure, extending the agility of the user. Furthermore, our reconstructed models contained high-level geometric features such as planes, which allow for registering the exhibition piece in 3D space, and enable perspectively correct visualization and exploration.

The user navigates in the city by superimposing walking directions over streets and buildings on her held-held device. This requires self-localization with respect to a global coordinate system. We presented a geo-localization method that allows for near real-time initialization of a SLAM system.

6.2 Taxonomy

We propose a unified registration system for mobile Augmented Reality (AR), which can be created by combining our contributions with the distinguished methods of others. The combination of these methods yields a multifunctional localization and mapping system, that can be applied in indoor, and, in particular, in urban outdoor environments. Running an AR application on a mobile device in an arbitrary environment, the proposed system aims at providing robust camera pose tracking as early as possible after startup, and ideally in a global coordinate system such as Universal Transverse Mercator (UTM). Since we cannot generally assume the immediate availability of a globally registered model at system startup, we differentiate between the following operation phases:

1. Localization with respect to a local coordinate system, *e.g.*, by initializing and mapping a SLAM map.

2. Registration with an existing offline-generated environment model, *e.g.*, by globally aligning the SLAM map.

3. Localization with respect to the updated global coordinate system of the SLAM map, possibly including further mapping of the scene.

The proposed system aims at minimizing the time span until local and global localization by making the best possible use of the available resources, including camera

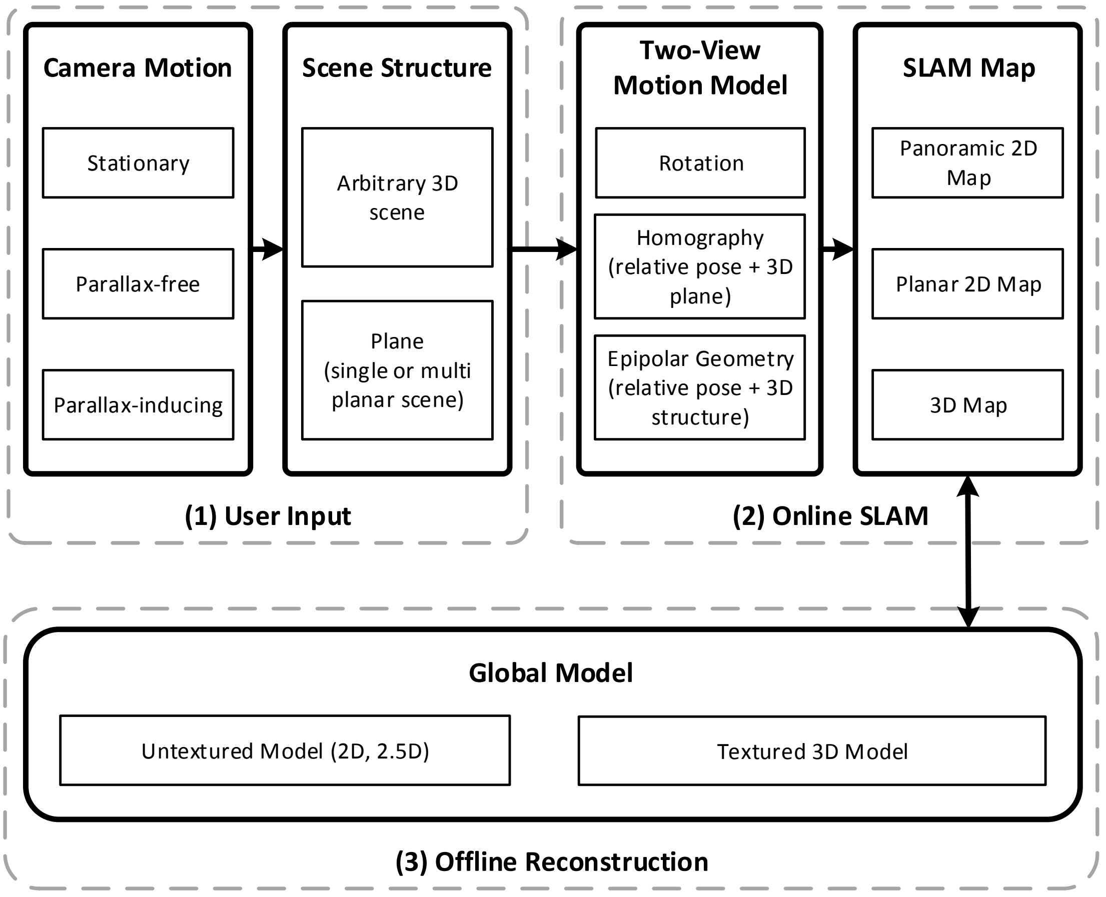


Figure 6.1: Summary of configuration options for a unified registration system. (1) Given the input of a mobile user, *i.e.*, the specifically performed camera motion and observed scene structure, (2) an online SLAM system can use diverse motion models to reconstruct a map, which can be registered with (3) available offline-generated globally aligned models.

and motion sensors on the mobile device, networking, and cloud services. In particular, constraints which facilitate the localization problem are exploited, including constraints resulting from the observed scene structure and device sensor measurements (*e.g.*, GPS location, compass orientation). Concurrently, the system aims to support a wide range of user inputs, including parallax-free and parallax-inducing camera motion. Figure 6.1 depicts all possible options which can be combined to specific system configurations.

During the first operation phase, the system supports the following configurations for initializing and building up a SLAM map, that can be used for camera localization in a local coordinate system:

- **parallax-free $\rightarrow$ rotation $\rightarrow$ panoramic 2D map**
  Parallax-free camera motion can be estimated with a rotation motion model, resulting in a panoramic 2D map [35, 173, 90], which can be represented as a panorama image.

- **parallax-inducing + plane $\rightarrow$ homography $\rightarrow$ planar 2D map**
  Parallax-inducing camera motion, when observing a single or multi-planar scene, can be estimated with a homography motion model, resulting in planar 2D maps [118, 133], which can be represented as an orthoimage, as described in Chapter 3.

- **parallax-inducing $\rightarrow$ epipolar $\rightarrow$ 3D map**
  Parallax-inducing camera motion, when observing arbitrary scene geometry, can be estimated with an epipolar motion model, resulting in a 3D map [72].

- **parallax-free/-inducing $\rightarrow$ hybrid 2D/3D map**
  Alternating parallax-free and parallax-inducing camera motion can be estimated with the appropriate motion models, resulting in hybrid 2D/3D maps [53, 119, 61], as described in Chapter 4.

- **stationary $\rightarrow$ images**
  A stationary camera simply delivers RGB images. Without further single-view scene assumptions, no geometric reconstruction is possible.

- **stationary + plane + lines $\rightarrow$ homography $\rightarrow$ planar 2D map**
  A single stationary camera image, when observing a planar scene providing vertical and horizontal lines, can be used to estimate vertical and horizontal vanishing points, resulting in a planar 2D map [14].

- **{ panoramic 2D map } $\rightarrow$ 3D map**
  A set of two or more overlapping panoramic 2D maps can be used to reconstruct a full 3D map [117].

The resulting SLAM maps are composed of one or more registered small field-of-view camera images from the mobile device, which can be stitched to wide field-of-view images in certain configurations (*e.g.*, from panoramic or planar maps). Besides camera tracking, these images can be employed for registration with offline-generated environment models providing a global coordinate frame. These models can be stored directly on the mobile device, or retrieved in-situ from Geographic Information System (GIS) cloud services using a sensor pose prior. Depending on what type of SLAM map and what type of offline model is available, the following registration options are available:

- **[images, panoramic map, planar map] $\leftrightarrow$ untextured model**
  Single images, and potentially also panoramic and orthographic images, can be localized with respect to untextured 2D [31, 24] or 2.5D models [157, 4], as described in Chapter 5.

- **[images, panoramic map, planar map] $\leftrightarrow$ textured 3D model**
  Single images can be localized with respect to textured 3D models, *i.e.*, 3D models composed of a set of registered reference images, using image-based localization methods [66, 6, 134]. Similarly, panoramic images, and potentially also ortho-images can be used as query images [2, 3].

- **3D map $\leftrightarrow$ textured 3D model**
  The registration of a 3D map with a textured 3D model is possible, *e.g.*, with the anchor point method of Ventura *et al.* [169].

Additionally, the registration with a global model allows for upgrading 2D SLAM maps into full 3D maps, *e.g.*, by rendering synthesized depth images from a dense global model such as a 2.5D digital elevation model, using the estimated global pose. The latter technique has been described in Chapter 5. Alternatively, the SLAM map can also be replaced or extended with information from the registered global model, followed by model-based detection and tracking, as proposed by Ventura *et al.* [171].

Subsequently, the SLAM system allows for continuous localization and mapping of the environment, providing globally aligned 6D poses. The 6D poses allow for displaying geo-referenced in-situ information, such as navigation hints, retrieved from location-based services.

6.3 Outlook

In conclusion, while we believe that the proposed unified registration system, including the contributions presented in this thesis, represent important steps towards the realization of the envisioned ubiquitous mobile AR experience, we have to ask ourselves: What could be the next steps to make mobile AR applications practically available everywhere, anytime, and for everybody?

Feature-based visual Simultaneous Localization and Mapping (SLAM) in the spirit of Parallel Tracking and Mapping (PTAM) has meanwhile reached a considerable degree of maturity on mobile computing platforms such as phones and tablets. Having invested many hours of research and careful engineering, commercial vendors including Qualcomm and Apple Metaio meanwhile integrate SLAM components are into their Software Development Kits (SDKs). Yet, visual feature-based SLAM must fail in conditions when camera images simply do not provide features, for example due to untextured scene surfaces, bad lighting conditions, motion blur, rapid camera motion, or combinations of these effects.

Concerning mobile devices, two important ongoing hardware developments may open up opportunities to overcome the limitations of visual feature-based SLAM. First, phones and tablets are expected to be equipped with GPUs which will truly provide general purpose computing capabilities. Second, time-of-flight and structured light depth sensors will likely be installed on mobile devices in the future. For example, the experimental Google Tango is the first device that is equipped with a GPGPU-enabled chipset and an RGB+Depth (RGB-D) sensor.

With these hardware capabilities available, RGB-D SLAM [69] and fusion-based SLAM [109] become feasible on mobile devices. However, depth sensors also have their weaknesses: most severely, they do not provide depth measurements on reflective surfaces, *e.g.*, in sunny outdoor conditions. Additionally, the depth sensor range is limited, *e.g.*, the Google Tango depth sensor gives values between 0.5 and 4 meters. However, by exploiting complementary strengths and weaknesses, visual and depth-based SLAM techniques could be combined to make SLAM more versatile and robust.

Furthermore, dense [110] and semi-dense [41] monocular visual SLAM algorithms gradually become available with more powerful and capable GPUs and CPUs implemented in mobile devices. These SLAM methods perform direct photometric image alignment, and probabilistic depth map estimation and fusion, resulting in more detailed maps and more robust camera tracking.

The problems related to alternating parallax-free and parallax-inducing camera motion, however, remain the same for these SLAM methods and thus there will be a demand for even more robust and advanced hybrid unconstrained SLAM methods. Especially since, as we observed in practice, untrained users regularly have severe problems when applying SLAM, often resulting in frustration, particularly when a SLAM system imposes to understand its limitations and inner workings.

Since current feature-based SLAM algorithms largely lack the capabilities to provide the scene knowledge that is required to develop more sophisticated AR applications, the mentioned “next-generation” SLAM methods and their (semi-)dense maps could provide the means to perform more advanced geometric scene understanding, including automatic detection of geometric primitives and shapes (planes, cubes, spheres), as well as semantic scene understanding, including detection of complex shapes and objects.

Alternatively or complementary to performing scene understanding in the local SLAM map, in-situ scene information can also be retrieved from location based cloud services. Cloud-based map services such as Microsoft Bing, Apple Maps and Google Earth contain massive amounts of geo-referenced information, similar to community-driven services such as OpenStreetMap. Retrieving information from these services, however, requires registration within a common reference coordinate system.

In outdoor environments, the accuracy and availability of Global Navigation Satellite Systems (GNSS) will likely improve in the future, as several GNSS are pending, including China’s BeiDou and the European Union’s Galileo, which will both become globally available by 2020. For the time being, we expect, however, that the accuracy of pure mobile motion sensor-based registration will remain insufficient for rendering convincing virtual AR overlays over real-world objects. Consequently, registration with existing environment models will remain an important research topic.

Localization with respect to large image databases and textured 3D models suffers from problems related to appearance changes and scalability. While Structure from Motion (SfM) has reached the point where world-scale 3D reconstructions from several tens of millions of images can be computed on a desktop computer within a few days [60], appearance changes between imagery taken at different times of day, weather conditions, or seasons, remain a big challenge for place and pose recognition systems. One approach to address this issue could be databases that contain redundant information, *i.e.*, reference images that depict the same objects multiple times covering different appearance “states”.

In contrast, untextured cadastral maps are basically available everywhere, but unfortunately, as we found, often in insufficient quality for geo-localization purposes (yet). Many of our failure cases were related to completely missing or erroneous model details such as building façade heights. Furthermore, although our method only required the visibility of two vertical building façade outlines in the query images, we observed that users often stood too close to the buildings and could not picture the entire façade with the narrow field-of-view phone cameras. These and further problems need to be addressed in the future to meet the practical requirements of mobile AR.

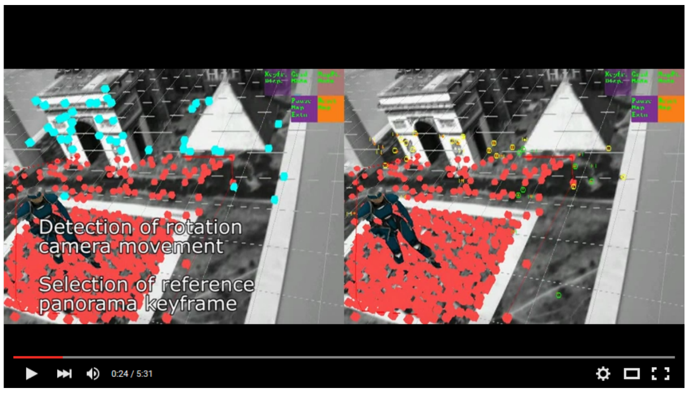
A Videos

A.1 Related to Chapter 3

Available at: `https://youtu.be/TxKCju5bIT0`

Related publication: C. Pirchheim and G. Reitmayr. Homography-based planar mapping and tracking for mobile phones. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'11)*, pages 27–36. IEEE, Oct. 2011

A.2 Related to Chapter 4


Detection of rotation camera movement
Selection of reference panorama keyframe
0:24 / 5:31

Available at: `https://youtu.be/oDsNPUhXuEU`
Related publication: C. Pirchheim, D. Schmalstieg, and G. Reitmayr. Handling pure camera rotation in keyframe-based SLAM. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'13)*, pages 229–238. IEEE, Oct. 2013

A.3 Related to Chapter 5

Available at: `https://youtu.be/PzV8VKC5buQ`
Related publication: C. Arth, C. Pirchheim, J. Ventura, D. Schmalstieg, and V. Lepetit. Instant Outdoor Localization and SLAM Initialization from 2.5D Maps. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'15)*, 2015

Acronyms

A-GPS	Assisted GPS.
AI	Artificial Intelligence.
API	Application Programming Interface.
AR	Augmented Reality.
BA	Bundle Adjustment.
CAD	Computer Aided Design.
CG	Computer Graphics.
CGI	Computer-Generated Imagery.
CPU	Central Processing Unit.
CUDA	Compute Unified Device Architecture.
CV	Computer Vision.
DoF	Degrees of Freedom.
DoG	Difference of Gaussian.
DR	Diminished Reality.
DTAM	Dense Tracking and Mapping.
EKF	Extended Kalman Filter.

ESM	Efficient Second-order Minimization.
EWK	Extent of World Knowledge.
FoV	Field Of View.
GIS	Geographic Information System.
GNSS	Global Navigation Satellite System.
GPGPU	General Purpose computing on Graphics Processing Units.
GPS	Global Positioning System.
GPU	Graphics Processing Unit.
GRIC	Geometric Robust Information Criterion.
HCI	Human-Computer Interaction.
HMD	Head-Mounted Display.
ICP	Iterative Closest Point.
IDE	Integrated Development Environment.
INS	Inertial Sensor.
KLT	Kanade-Lucas-Tomasi.
LBS	Location Based Service.
LiDAR	Light Detection and Ranging.
MAV	Micro Aerial Vehicle.
MR	Mediated Reality.
NCC	Normalized Cross Correlation.
NFC	Near Field Communication.
OpenCL	Open Computing Language.
P3P	Perspective Three Point.
PTAM	Parallel Tracking and Mapping.
RANSAC	Random Sample Consensus.
RGB-D	RGB+Depth.

ROI Region Of Interest.

RTK Real-Time Kinematic.

SBI Small Blurry Images.

SDK Software Development Kit.

SfM Structure from Motion.

SIFT Scale-Invariant Feature Transform.

SL Structured Light.

SLAM Simultaneous Localization and Mapping.

ToF Time of Flight.

UTM Universal Transverse Mercator.

VO Visual Odometry.

VR Virtual Reality.

WGS World Geodetic System.

Bibliography

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a day. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 5, pages 72–79, 2009. (page 21)

[2] C. Arth, M. Klopschitz, G. Reitmayr, and D. Schmalstieg. Real-time self-localization from panoramic images on mobile devices. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*, pages 37–46. Ieee, Oct. 2011. (page 33, 35, 141)

[3] C. Arth, A. Mulloni, and D. Schmalstieg. Exploiting Sensors on Mobile Phones to Improve Wide-Area Localization. In *International Conference on Pattern Recognition*, 2012. (page 35, 141)

[4] C. Arth, C. Pirchheim, J. Ventura, D. Schmalstieg, and V. Lepetit. Instant Outdoor Localization and SLAM Initialization from 2.5D Maps. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'15)*, 2015. (page 141)

[5] C. Arth, G. Reitmayr, and D. Schmalstieg. Full 6DOF Pose Estimation from Geo-Located Images. In *ACCV'12 Proceedings of the 11th Asian conference on Computer Vision*, volume 7726 LNCS, pages 705–717. 2012. (page 35)

[6] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide area localization on mobile phones. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 73–82. IEEE, Oct. 2009. (page 11, 27, 35, 141)

[7] R. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997. (page 17)

[8] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys. Large Scale Visual Geo-Localization of Images in Mountainous Terrain. In *ECCV'12 Proceedings of the 12th European conference on Computer Vision*, volume 7573 LNCS, pages 517–530. 2012. (page 27)

[9] G. Baatz, O. Saurer, K. Koser, and M. Pollefeys. Leveraging Topographic Maps for Image to Terrain Alignment. *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 487–492, Oct. 2012. (page 27)

[10] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001. (page 41)

[11] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, Feb. 2004. (page 21)

[12] M. Bansal and K. Daniilidis. Geometric Urban Geo-Localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3978–3985. Ieee, June 2014. (page 27)

[13] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *ECCV*, pages 404–417. 2006. (page 22)

[14] J. C. Bazin and M. Pollefeys. 3-line RANSAC for orthogonal vanishing point detection. In *IEEE International Conference on Intelligent Robots and Systems*, number M, pages 4282–4287. Ieee, Oct. 2012. (page 140)

[15] S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 943–948. IEEE, 2004. (page 41, 42, 86)

[16] P. Bunnun and W. W. Mayol-Cuevas. OutlinAR: An assisted interactive model building system with reduced computational effort. *Proceedings - 7th IEEE International Symposium on Mixed and Augmented Reality 2008, ISMAR 2008*, pages 61–64, 2008. (page 48)

[17] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *Proceedings of the European Conference on Computer Vision. ECCV'10*, volume 6314, pages 778–792. 2010. (page 22)

[18] J. Canny. A computational approach to edge detection. *IEEE transactions on pattern analysis and machine intelligence*, 8(6):679–698, 1986. (page 21, 32)

[19] R. Castle, G. Klein, and D. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *2008 12th IEEE International Symposium on Wearable Computers*, pages 15–22. Ieee, 2008. (page 41, 42)

[20] R. O. Castle and D. W. Murray. Keyframe-based recognition and localization during video-rate parallel tracking and mapping. *Image and Vision Computing*, 29(8):524–532, July 2011. (page 45, 46)

[21] T.-J. Cham, A. Ciptadi, W.-C. Tan, M.-T. Pham, and L.-T. Chia. Estimating camera pose from a single urban ground-view omnidirectional image and a 2D building outline map. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 366–373, June 2010. (page 27)

[22] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, Apr. 2011. (page 121)

[23] D. Chekhlov, A. P. Gee, A. Calway, and W. Mayol-Cuevas. Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual SLAM. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, pages 1–4. Ieee, Nov. 2007. (page 46)

[24] H. Chu, A. Gallagher, and T. Chen. GPS Refinement and Camera Orientation Estimation from a Single Image and a 2D Map. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 171–178, June 2014. (page 26, 27, 28, 141)

[25] J. Civera, A. Davison, and J. Montiel. Interacting multiple model monocular SLAM. In *2008 IEEE International Conference on Robotics and Automation*, pages 3704–3709. Ieee, May 2008. (page 12, 50)

[26] J. Civera, A. J. Davison, J. A. Magallón, and J. M. M. Montiel. Drift-free real-time sequential mosaicing. *International Journal of Computer Vision*, 2009. (page 49, 81)

[27] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, 2008. (page 50)

[28] L. Clemente, A. Davison, I. Reid, J. Neira, and J. Tardos. Mapping Large Loops with a Single Hand-Held Camera. In *Proceedings of Robotics: Science and Systems*, 2007. (page 42)

[29] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008. (page 42)

[30] M. Cummins and P. Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123, 2011. (page 39)

[31] P. David and S. Ho. Orientation descriptors for localization in urban environments. In *IEEE International Conference on Intelligent Robots and Systems*, Sept. 2011. (page 27, 141)

[32] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003. (page 8, 12, 36, 37)

[33] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: real-time single camera SLAM. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–67, June 2007. (page 37, 81)

[34] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, pages 11–20, New York, New York, USA, 1996. ACM Press. (page 108)

[35] S. DiVerdi, J. Wither, and T. Hollerer. Envisor: Online Environment Map Construction for Mixed Reality. In *2008 IEEE Virtual Reality Conference*, pages 19–26. IEEE, 2008. (page 12, 47, 49, 81, 108, 140)

[36] P. Dollar, Z. Tu, P. Perona, and S. Belongie. Integral Channel Features. In *Procedings of the British Machine Vision Conference 2009*, pages 91.1–91.11. British Machine Vision Association, 2009. (page 121)

[37] T. Drummond. Lie groups and Lie algebras for 3D Geometry, Engineering and Computer Vision. Technical report, Department of Electrical and Computer Systems Engineering at Monash University, 2012. (page 22)

[38] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, 2002. (page 32)

[39] E. Eade and T. Drummond. Scalable monocular SLAM. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 469–476. Ieee, 2006. (page 81)

[40] E. Eade and T. Drummond. Unified Loop Closing and Recovery for Real Time Monocular SLAM. In *Procedings of the British Machine Vision Conference 2008*, pages 6.1–6.10. British Machine Vision Association, 2008. (page 42)

[41] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *ECCV*, pages 834–849. 2014. (page 8, 38, 142)

[42] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1456. IEEE, 2013. (page 38)

[43] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. *Photogrammetric computer vision*, 2006. (page 44, 92)

[44] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning Hierarchical Features for Scene Labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, Aug. 2013. (page 122)

[45] O. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 02(03):485–508, Sept. 1988. (page 10, 41, 53, 58, 61)

[46] S. Feiner, B. MacIntyre, T. Höllerer, and A. Webster. A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. *Personal and Ubiquitous Computing*, 1(4):208–217, 1997. (page 11, 24)

[47] S. K. Feiner. Augmented reality: a new way of seeing. *Scientific American*, 286(4):48–55, 2002. (page 17)

[48] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, 1981. (page 22)

[49] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building Rome on a Cloudless Day. In *ECCV'10 Proceedings of the 11th European conference on Computer vision*, volume 6314 LNCS, pages 368–381. 2010. (page 21)

[50] F. Fraundorfer and D. Scaramuzza. Visual odometry: Part II: Matching, robustness, optimization, and applications. *IEEE Robotics and Automation Magazine*, 19(2):78–90, 2012. (page 8, 22)

[51] F. Fraundorfer, K. Schindler, and H. Bischof. Piecewise planar scene reconstruction from sparse correspondences. *Image and Vision Computing*, 24(4):395–406, 2006. (page 48)

[52] H. Fuchs, G. D. Abram, and E. D. Grant. Near real-time shaded display of rigid objects. *Proceedings of the 10th annual conference on Computer graphics and interactive techniques - SIGGRAPH '83*, 17(3):65–72, 1983. (page 117)

[53] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Hollerer. Live tracking and mapping from both general and rotation-only camera motion. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 13–22. IEEE, Nov. 2012. (page 50, 106, 108, 127, 140)

[54] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Höllerer. Model Estimation and Selection towards Unconstrained Real-Time Tracking and Mapping. *IEEE transactions on visualization and computer graphics*, pages 1–14, 2013. (page 12, 50, 51, 108, 109)

[55] A. Gee, D. Chekhlov, A. Calway, and W. Mayol-Cuevas. Discovering Higher Level Structure in Visual SLAM. *IEEE Transactions on Robotics*, 24(5):980–990, Oct. 2008. (page 45, 46, 48)

[56] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. LSD: a fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):722–32, Apr. 2010. (page 21, 114)

[57] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Procedings of the Alvey Vision Conference 1988*, pages 23.1–23.6. Alvey Vision Club, 1988. (page 21)

[58] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, second edition, 2004. (page 8, 22, 59, 93)

[59] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997. (page 22)

[60] J. Heinly, J. L. Schoenberger, E. Dunn, and J.-M. Frahm. Reconstructing the World in Six Days. In *CVPR*, 2015. (page 143)

[61] C. D. Herrera, K. Kim, J. Kannala, K. Pulli, and J. Heikkila. DT-SLAM: Deferred Triangulation for Robust SLAM. In *2014 2nd International Conference on 3D Vision*, 2014. (page 12, 43, 51, 52, 108, 109, 140)

[62] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab. Dominant orientation templates for real-time detection of texture-less objects. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2257–2264. Ieee, June 2010. (page 48)

[63] T. Höllerer, J. Wither, and S. DiVerdi. Anywhere Augmentation: Towards Mobile Augmented Reality in Unprepared Environments. *Location Based Services and TeleCartography*, pages 393 – 416, 2007. (page 1)

[64] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629, Apr. 1987. (page 42)

[65] P. J. Huber. *Robust Statistics*, volume 82. 1981. (page 22)

[66] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2599–2606. IEEE, June 2009. (page 11, 33, 34, 35, 141)

[67] V. Jain and E. Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. In *CVPR 2011*, pages 577–584. IEEE, June 2011. (page 120)

[68] Y. Jeong, D. Nistér, D. Steedly, R. Szeliski, and I.-S. Kweon. Pushing the envelope of modern methods for bundle adjustment. *IEEE transactions on pattern analysis and machine intelligence*, 34(8):1605–17, Aug. 2012. (page 44)

[69] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2100–2106. Ieee, Nov. 2013. (page 39, 142)

[70] G. Klein and T. Drummond. Robust visual tracking for non-instrumental augmented reality. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, 2003. (page 32)

[71] G. Klein and T. Drummond. Sensor fusion and occlusion refinement for tablet-based AR. In *ISMAR 2004: Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, number Ismar, pages 38–47. Ieee, 2004. (page 32)

[72] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE, Nov. 2007. (page 8, 9, 12, 36, 37, 38, 40, 43, 55, 72, 81, 83, 88, 105, 123, 140)

[73] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In *ECCV*, volume 5303 LNCS, pages 802–815, 2008. (page 9, 37, 40, 42, 85)

[74] G. Klein and D. Murray. Parallel Tracking and Mapping on a camera phone. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 83–86. IEEE, Oct. 2009. (page 9, 11, 37, 38, 40, 46, 101, 106)

[75] K. Kolev, P. Tanskanen, P. Speciale, and M. Pollefeys. Turning Mobile Phones into 3D Scanners. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3946–3953, 2014. (page 38)

[76] Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-Form Solutions to Minimal Absolute Pose Problems with Known Vertical Direction. In *ACCV*, pages 216–229, 2011. (page 116)

[77] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. (page 39)

[78] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard. Large scale graph-based SLAM using aerial images as prior information. *Autonomous Robots*, 30(1):25–39, Aug. 2010. (page 39, 45)

[79] D. Kurz and S. Benhimane. Gravity-aware handheld augmented reality. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*, pages 111–120. Ieee, Oct. 2011. (page 114)

[80] W. Lee, Y. Park, V. Lepetit, and W. Woo. Point-and-shoot for ubiquitous tagging on mobile phones. In *9th IEEE International Symposium on Mixed and Augmented Reality 2010: Science and Technology, ISMAR 2010 - Proceedings*, pages 57–64. Ieee, Oct. 2010. (page 48)

[81] V. Lepetit and P. Fua. Monocular Model-Based 3D Tracking of Rigid Objects. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005. (page 22)

[82] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006. (page 30, 42)

[83] V. Lepetit, L. Vacchetti, D. Thalmann, and P. Fua. Fully automated and stable registration for augmented reality applications. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 93–102. IEEE Comput. Soc, 2003. (page 29, 30, 32)

[84] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3d point clouds. In *ECCV*, 2012. (page 34, 35)

[85] J. Lim, J.-M. Frahm, and M. Pollefeys. Online environment mapping. In *CVPR 2011*, pages 3489–3496. IEEE, June 2011. (page 45)

[86] T. Lindeberg. Scale-space theory: a basic tool for analyzing structures at different scales. *Journal of Applied Statistics*, 21(1):225–270, 1994. (page 21)

[87] C. Loop and Zhengyou Zhang. Computing rectifying homographies for stereo vision. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 125–131. IEEE, 1999. (page 53, 58)

[88] M. I. Lourakis and A. A. Argyros. Efficient, causal camera tracking in unprepared environments. *Computer Vision and Image Understanding*, 99(2):259–290, Aug. 2005. (page 29)

[89] M. I. a. Lourakis and A. a. Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1):1–30, 2009. (page 72)

[90] S. Lovegrove and A. J. Davison. Real-time spherical mosaicing using whole image alignment. In *ECCV*, volume 6313, pages 73–86, 2010. (page 12, 47, 49, 53, 81, 140)

[91] S. Lovegrove, A. J. Davison, and J. Ibañez Guzmán. Accurate visual odometry from a rear parking camera. In *IEEE Intelligent Vehicles Symposium*, pages 788–793. IEEE, June 2011. (page 47, 48)

[92] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2. Ieee, 1999. (page 22)

[93] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004. (page 30)

[94] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint Conference on Artificial Intelligence*, number x, pages 674–679, 1981. (page 21)

[95] E. Malis. Improving vision-based control using efficient second-order minimization techniques. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 2(April):1843–1848, 2004. (page 48, 50)

[96] E. Malis and M. Vargas. Deeper understanding of the homography decomposition for vision-based control. Technical report, 2007. (page 13, 41, 53, 58)

[97] S. Mann. Mediated Reality with implementations for everyday life. *Presence: Teleoperators and Virtual Environments*, pages 1–12, 2002. (page 17)

[98] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, Sept. 2004. (page 21)

[99] C. Mei, S. Benhimane, E. Malis, and P. Rives. Efficient homography-based tracking and 3-D reconstruction for single-viewpoint sensors. *IEEE Transactions on Robotics*, 24(6):1352–1364, 2008. (page 48)

[100] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. A Constant-Time Efficient Stereo SLAM System. In *Procedings of the British Machine Vision Conference 2009*, pages 54.1–54.11. British Machine Vision Association, 2009. (page 39)

[101] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt. Scalable 6-DOF Localization on Mobile Devices. In *ECCV*, pages 268–283, 2014. (page 35)

[102] P. Milgram and F. Kishino. Taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems*, E77-D(12):1321–1329, 1994. (page 1, 5, 7, 21, 137)

[103] N. Molton, A. Davison, and I. Reid. Locally Planar Patch Features for Real-Time Structure from Motion. In *Procedings of the British Machine Vision Conference 2004*, pages 90.1–90.10. British Machine Vision Association, 2004. (page 45)

[104] J. M. M. Montiel and A. J. Davison. A visual compass based on SLAM. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2006, pages 1917–1922, 2006. (page 12, 49)

[105] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3D reconstruction. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 363–370, 2006. (page 45)

[106] A. Mulloni, M. Ramachandran, G. Reitmayr, D. Wagner, R. Grasset, and S. Diaz. User friendly SLAM initialization. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 153–162. Ieee, Oct. 2013. (page 10, 41)

[107] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Van Gool, and W. Purgathofer. A survey of urban reconstruction. *Computer Graphics Forum*, 32(6):146–177, 2013. (page 21)

[108] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1498–1505. IEEE, June 2010. (page 38)

[109] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, Oct. 2011. (page 8, 39, 142)

[110] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327. IEEE, Nov. 2011. (page 8, 12, 38, 81, 142)

[111] D. Nister. Preemptive RANSAC for live structure and motion estimation. In *Proceedings Ninth IEEE International Conference on Computer Vision*, volume 16, pages 199–206 vol.1. IEEE, 2003. (page 37)

[112] D. Nister. A minimal solution to the generalised 3-point pose problem. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, pages 560–567. IEEE, Jan. 2004. (page 37)

[113] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–77, June 2004. (page 10, 22, 37, 41)

[114] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 2004. (page 8, 37)

[115] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006. (page 34)

[116] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, (May):2262–2269, 2006. (page 45)

[117] Q. Pan, C. Arth, E. Rosten, G. Reitmayr, and T. Drummond. Rapid scene reconstruction on mobile phones from panoramic images. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*, pages 55–64. Ieee, Oct. 2011. (page 140)

[118] C. Pirchheim and G. Reitmayr. Homography-based planar mapping and tracking for mobile phones. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'11)*, pages 27–36. IEEE, Oct. 2011. (page 140)

[119] C. Pirchheim, D. Schmalstieg, and G. Reitmayr. Handling pure camera rotation in keyframe-based SLAM. In *Proceedings of the International Symposium on Mixed*

and Augmented Reality (ISMAR'13), pages 229–238. IEEE, Oct. 2013. (page 12, 127, 140)

[120] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed Real-Time Urban 3D Reconstruction from Video. *Int. J. Comput. Vision*, 78(2-3):143–167, 2008. (page 21, 39)

[121] M. Pollefeys, F. Verbiest, and L. Van Gool. Surviving Dominant Planes in Uncalibrated Structure and Motion Recovery. In *ECCV*, 2002. (page 43)

[122] S. J. D. Prince, K. Xu, and A. D. Cheok. Augmented reality camera tracking with homographies. *IEEE Computer Graphics and Applications*, 22(6):39–45, 2002. (page 29)

[123] S. Ramalingam, S. Bouaziz, and P. Sturm. Pose estimation using both points and lines for Geo-localization. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4716–4723. Ieee, May 2011. (page 26)

[124] G. Reitmayr and T. W. Drummond. Going out: Robust model-based tracking for outdoor augmented reality. In *Proceedings - ISMAR 2006: Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 109–118. Ieee, Oct. 2006. (page 32)

[125] G. Reitmayr and T. W. Drummond. Initialisation for Visual Tracking in Urban Environments. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–9. IEEE, Nov. 2007. (page 11, 32)

[126] G. Reitmayr, E. Eade, and T. W. Drummond. Semi-automatic Annotations in Unknown Environments. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, number 50, pages 1–4. IEEE, Nov. 2007. (page 45)

[127] J. Repko and M. Pollefeys. 3D models from extended uncalibrated video sequences: Addressing key-frame selection and projective drift. In *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM*, 2005. (page 43)

[128] D. Robertson and R. Cipolla. An Image-Based System for Urban Navigation. In *Procedings of the British Machine Vision Conference 2004*, volume 1, pages 84.1–84.10. British Machine Vision Association, 2004. (page 34)

[129] E. Rosten and T. Drummond. Machine Learning for High-Speed Corner Detection. In *ECCV'06 Proceedings of the 9th European conference on Computer Vision*, volume 3951 LNCS, pages 430–443. 2006. (page 21, 31, 41, 67)

[130] C. Rother and S. Carlsson. Linear multi view reconstruction and camera recovery using a reference plane. *International Journal of Computer Vision*, 49(2-3):117–141, 2002. (page 114)

[131] A. Ruiz, P. E. Lopez-de Teruel, and L. Fernandez-Maimo. Practical Planar Metric Rectification. In *Procedings of the British Machine Vision Conference 2006*, pages 60.1–60.10. British Machine Vision Association, 2006. (page 53, 58, 59)

[132] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, Oct. 2008. (page 121)

[133] R. F. Salas-Moreno, B. Glocken, P. H. J. Kelly, and A. J. Davison. Dense planar SLAM. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 157–164, 2014. (page 80, 140)

[134] T. Sattler, B. Leibe, and L. Kobbelt. Improving Image-Based Localization by Active Correspondence Search. In *ECCV'12 Proceedings of the 12th European conference on Computer Vision*, pages 752–765. 2012. (page 34, 141)

[135] D. Scaramuzza and F. Fraundorfer. Tutorial: Visual odometry. *IEEE Robotics and Automation Magazine*, 18(4):80–92, 2011. (page 8, 22)

[136] G. Schall, D. Wagner, G. Reitmayr, E. Taichmann, M. Wieser, D. Schmalstieg, and B. Hofmann-Wellenhof. Global pose estimation using multi-sensor fusion for outdoor Augmented Reality. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 153–162, 2009. (page 25)

[137] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–7. Ieee, June 2007. (page 11, 34)

[138] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001. (page 121)

[139] T. Schöps, J. Engel, and D. Cremers. Semi-dense visual odometry for AR on a smartphone. In *IEEE International Symposium on Mixed and Augmented Reality*, 2014. (page 38)

[140] H. Shao, T. Svoboda, and L. V. Gool. ZuBuD – Zurich Buildings Database for Image Based Recognition. Technical Report 260, 2003. (page 120)

[141] J. S. J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, number June, 1994. (page 21, 41)

[142] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2008. (page 122)

[143] D. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. *Robotics: science and systems*, 220(3):1–8, 2009. (page 45)

[144] G. Simon. In-Situ 3D Sketching Using a Video Camera as an Interaction and Tracking Device. In *The European Association for Computer Graphics - Eurographics*, 2010. (page 48)

[145] G. Simon. Tracking-by-synthesis using point features and pyramidal blurring. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*, volume 2011, pages 85–92. Ieee, Oct. 2011. (page 32)

[146] G. Simon, M.-o. Berger, L. Uhp, and N. Inria. Real time registration of known or recovered multi-planar structures : application to AR. In *BMVC*, pages 567–576, 2002. (page 29)

[147] G. Simon, A. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, pages 120–128. IEEE, June 2000. (page 29)

[148] I. Skrypnyk and D. G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *ISMAR 2004: Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 110–119, 2004. (page 29, 30)

[149] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, 1987. (page 8, 37)

[150] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics*, 2006. (page 21)

[151] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Scale Drift-Aware Large Scale Monocular SLAM. *Robotics: Science and Systems*, 2:5, 2010. (page 45)

[152] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Visual SLAM: Why filter? *Image and Vision Computing*, 30(2):65–77, Feb. 2012. (page 9, 38)

[153] R. Szeliski. Image Alignment and Stitching: A Tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2(1):1–104, 2006. (page 22, 49)

[154] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010. (page 11, 21)

[155] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. *Proceeding of the 1st ACM international conference on Multimedia information retrieval - MIR '08*, page 427, 2008. (page 11, 34)

[156] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao. Robust monocular SLAM in dynamic environments. In *2013 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2013*, pages 209–218, 2013. (page 42)

[157] A. Taneja, L. Ballan, and M. Pollefeys. Registration of Spherical Panoramic Images with Cadastral 3D Models. *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 479–486, Oct. 2012. (page 11, 26, 27, 141)

[158] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live Metric 3D Reconstruction on Mobile Phones. In *2013 IEEE International Conference on Computer Vision*, pages 65–72, 2013. (page 38)

[159] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *2008 IEEE/RSJ International*

Conference on Intelligent Robots and Systems, pages 2531–2538. Ieee, Sept. 2008. (page 39)

[160] T. Thormählen, H. Broszio, and A. Weissenfeld. Keyframe selection for camera motion and structure estimation from multiple views. In *ECCV*, pages 523–535. Springer, 2004. (page 10, 43)

[161] C. Tomasi. Detection and Tracking of Point Features. Technical Report April, School of Computer Science, Carnegie Mellon Univ., 1991. (page 21)

[162] P. H. S. Torr. Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision*, 50(1):35–61, 2002. (page 43, 50, 109)

[163] B. Triggs, P. F. Mclauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle Adjustment - A Modern Synthesis. *Vision algorithms: theory and practice. S*, 34099:298–372, 2000. (page 9, 22, 44)

[164] G. Vaca-Castano, A. R. Zamir, and M. Shah. City scale geo-spatial trajectory estimation of a moving camera. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1186–1193. Ieee, June 2012. (page 34)

[165] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3D camera tracking. In *ISMAR 2004: Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 48–57, 2004. (page 30)

[166] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3D tracking using online and offline information. *IEEE transactions on pattern analysis and machine intelligence*, 26(10):1385–91, Oct. 2004. (page 30)

[167] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. VideoTrace. In *ACM SIGGRAPH 2007 papers on - SIGGRAPH '07*, volume 26, page 86, New York, New York, USA, 2007. ACM Press. (page 21)

[168] A. van den Hengel, R. Hill, B. Ward, and A. Dick. In situ image-based modeling. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 107–110. IEEE, Oct. 2009. (page 48)

[169] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. Global localization from monocular SLAM on a mobile phone. *IEEE Transactions on Visualization and Computer Graphics*, 20(4):531–539, 2014. (page 10, 35, 123, 127, 128, 141)

[170] J. Ventura and T. Hollerer. Online environment model estimation for augmented reality. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 103–106. IEEE, Oct. 2009. (page 45, 46, 48)

[171] J. Ventura and T. Höllerer. Wide-area scene mapping for mobile visual tracking. In *ISMAR 2012 - 11th IEEE International Symposium on Mixed and Augmented Reality 2012, Science and Technology Papers*, pages 3–12. Ieee, Nov. 2012. (page 141)

[172] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004. (page 120)

[173] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg. Real-time panoramic mapping and tracking on mobile phones. In *IEEE Virtual Reality*, pages 211–218. Ieee, Mar. 2010. (page 12, 47, 50, 53, 81, 140)

[174] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings - 7th IEEE International Symposium on Mixed and Augmented Reality 2008, ISMAR 2008*, pages 125–134. Ieee, Sept. 2008. (page 30)

[175] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE transactions on visualization and computer graphics*, 16(3):355–68, 2010. (page 30, 31, 42, 69, 94)

[176] D. Wagner, D. Schmalstieg, and H. Bischof. Multiple target detection and tracking with guaranteed framerates on mobile phones. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 57–64. Ieee, Oct. 2009. (page 30)

[177] A. Wendel, M. Maurer, G. Graber, T. Pock, and H. Bischof. Dense reconstruction on-the-fly. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1450–1457, 2012. (page 38)

[178] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós. A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 57(12):1188–1197, Dec. 2009. (page 42)

[179] B. Williams, G. Klein, and I. Reid. Real-time SLAM relocalisation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8. Ieee, 2007. (page 42)

[180] C. Wu. Towards linear-time incremental structure from motion. In *Proceedings - 2013 International Conference on 3D Vision, 3DV 2013*, pages 127–134, 2013. (page 21)

[181] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, number 1, pages 3057–3064. Ieee, June 2011. (page 44)

[182] X. Yin, P. Wonka, and A. Razdan. Generating 3D building models from architectural drawings: a survey. *IEEE computer graphics and applications*, 29(1):20–30, 2009. (page 21)

[183] A. R. Zamir and M. Shah. Accurate Image Localization Based on Google Maps Street View. In *ECCV'10 Proceedings of the 11th European conference on Computer vision*, volume 6314 LNCS, pages 255–268. 2010. (page 34)

[184] P. A. Zandbergen and S. J. Barbeau. Positional Accuracy of Assisted GPS Data from High-Sensitivity GPS-enabled Mobile Phones. *Journal of Navigation*, 64(03):381–399, July 2011. (page 25, 121, 122)

[185] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, 1999. (page 22)

[186] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. In *Artificial Intelligence*, 1995. (page 21, 68)

[187] B. Zitová and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, Oct. 2003. (page 22)