RENAUD GERVAIS

Towards High-Accuracy Augmented Reality GIS for Architecture and Geo-Engineering

Mémoire présenté
à la Faculté des études supérieures et postdoctorales de l'Université Laval
dans le cadre du programme de maîtrise en sciences géomatiques
pour l'obtention du grade de Maître *ès* sciences (M.Sc.)

Faculté de Foresterie, de Géographie et de Géomatique
UNIVERSITÉ LAVAL
QUÉBEC

2012

Résumé

L'architecture et la géo-ingénierie sont des domaines où les professionnels doivent prendre des décisions critiques. Ceux-ci requièrent des outils de haute précision pour les assister dans leurs tâches quotidiennes. La Réalité Augmentée (RA) présente un excellent potentiel pour ces professionnels en leur permettant de faciliter l'association des plans 2D/3D représentatifs des ouvrages sur lesquels ils doivent intervenir, avec leur perception de ces ouvrages dans la réalité. Les outils de visualisation s'appuyant sur la RA permettent d'effectuer ce recalage entre modélisation spatiale et réalité dans le champ de vue de l'usager. Cependant, ces systèmes de RA nécessitent des solutions de positionnement en temps réel de très haute précision. Ce n'est pas chose facile, spécialement dans les environnements urbains ou sur les sites de construction. Ce projet propose donc d'investiguer les principaux défis que présente un système de RA haute précision basé sur les panoramas omnidirectionels.

Abstract

Architecture and geo-engineering are application domains where professionals need to take critical decisions. These professionals require high-precision tools to assist them in their daily decision taking process. Augmented Reality (AR) shows great potential to allow easier association between the abstract 2D drawings and 3D models representing infrastructure under reviewing and the actual perception of these objects in the reality. The different visualization tools based on AR allow to overlay the virtual models and the reality in the field of view of the user. However, the architecture and geo-engineering context requires high-accuracy and real-time positioning from these AR systems. This is not a trivial task, especially in urban environments or on construction sites where the surroundings may be crowded and highly dynamic. This project investigates the accuracy requirements of mobile AR GIS as well as the main challenges to address when tackling high-accuracy AR based on omnidirectional panoramas.

To the three loves of my life, Odette, Joline and Dominique and to my father, Paul.

A month in the laboratory can often save an hour in the library.
– Frank Westheimer

Acknowledgments

I would like to thank the following persons and organizations for their support:

- Stéphane Côté, my co-director at Bentley Systems for his infinite supply of fresh ideas and paradigm shifts and his availability to discuss those ideas. Stéphane was also the one who introduced me to the wonderful world of research and he greatly contributed to shape the very motivating path that I now follow. I thank him deeply.
- Sylvie Daniel, my director at Laval University for her experience and rigor which were key elements helping this project stay on track, even when entering deep forests of thoughts was unavoidable.
- Bentley Systems for offering a stimulating applied research environment and for giving me experience both in the corporate and academic world at the same time.
- Stéphane Poirier, my dear friend and colleague for all the discussions and help he provided. A one hour problem solving session with Stéphane could easily save me day(s) of work each time.

This project was financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Fonds québécois de la recherche sur la nature et les technologies (FQRNT) and Bentley Systems Inc. in the context of the Industrial Innovation Scolarships program BMP-Innovation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Context

Architecture and civil engineering are fields that interact with abstract representations of real objects. These abstractions are required for different tasks: planning, design, construction, maintenance, inspection, etc. These tasks, however, often involve interpretation ambiguities. For instance, in the case of the *construction* task, builders have to correctly interpret the CAD (Computer-Aided Design) files that the architects and engineers provided in order to build the given infrastructure according to specifications. As abstract representations (2D blueprints and 3D models) are used to convey the information, and because it is impossible to model every square or cubic millimeter of the environment and of the objects to be built, ambiguities are inherent to these representations. Thus, they could be interpreted differently by the various actors exploiting them on site and at the office. Therefore, *inspection and monitoring* tasks are generally requested in architecture or civil engineering projects in order to ensure that minimal differences exist between the as-built and the as-planned[1] infrastructure. Ambiguities in the interpretation stem from the need to compare 2D or 3D abstractions with real, 3D objects that present very different physical features (*e.g.* texture, color) and that are located in a different visual setting.

Nowadays, the main tools that support the use of 2D and 3D models in the architecture and geo-engineering fields consist of pieces of software that can be classified into two categories: GIS (Geographic Information System) that are mainly used to capture,

[1]The term "as-planned" or "as-designed" is often used to describe the way a structure was planned to be built (e.g. the original blueprints) while the term "as-built" describes the way the same construction was actually built in the real world (i.e. its actual realization).

store, manipulate, analyze, manage and display all types of geographically referenced data and CAD (Computer-Aided Design) that are used to streamline the design process: drafting, documentation and manufacturing. These tools can be used when the user has access to a computer; data can be requested, visualized, updated, etc. However, if the architect or engineer needs to go on-site, his alternatives for getting access to the data begin to fall short; paper plans are often the only viable support. While paper is a great tool, being light, foldable and resistant, it presents one main limitation: the model it represents is *static* both in time and space.

That limitation can be illustrated in the following example: a civil engineer is on a construction site where a given structure is being built (Figure 1.1). His visit of the site aims at conducting the usual verifications (validation of the project progress according to the schedule, survey of the differences between the as-planned and as-built structures, etc). Before leaving the office, he takes the paper plan representing the building as it should look like at today's date if the project is on time. He prints the paper plans from multiple points of view from key locations. During the on-site evaluation, he has to constantly compare the objects in the real world with their abstract representation on blueprints in order to know where he is located in the plan's reference system. This is not a trivial task, especially if the assessment task requires a high level of accuracy: the engineer has to compare the actual reality with a mental, intangible, 3D reconstruction of a 2D blueprint. If he notices a particular structure that highly differs from the actual model due to delay in the construction schedule, he needs to access the plans and notes of the previous evaluation (checkpoint) to be able to quantify the effective delay in the project. If the structure also happens to be complex, the civil engineer may feel the need to use 3D information to better conduct his assessment task (the usefulness of 3D data increases with the complexity of the structure being surveyed as 2D projections of complex structures tend to get spatially confusing – see Figure 1.2 –, mostly because of clutter and the fact that the human brain can only keep so much information in its working memory (3-5 chunks of information [9])). If he did not anticipate such a need (it is indeed difficult to bring on site all the data related to a project) the only way he can access 3D models, most of the time, is to go to the site's portacabin (if there is one on the site) or go back to the office to either take a decision or to print a new set of paper plans with the required information.

Mobile technology, however, continues to evolve and alternatives to static desktops or bulky laptops begin to appear for actors of construction sites. Mobile GIS are tools that are specifically well adapted for manipulating and accessing geospatial data on the field (ESRI[2], Bentley[3], etc.). Such systems retain the main features of standard, desktop

[2]http://goo.gl/SBrrr
[3]http://goo.gl/BjbhG

Figure 1.1 – Examples of construction site monitoring in action. (*source: (left): http://goo.gl/kmUvE*)

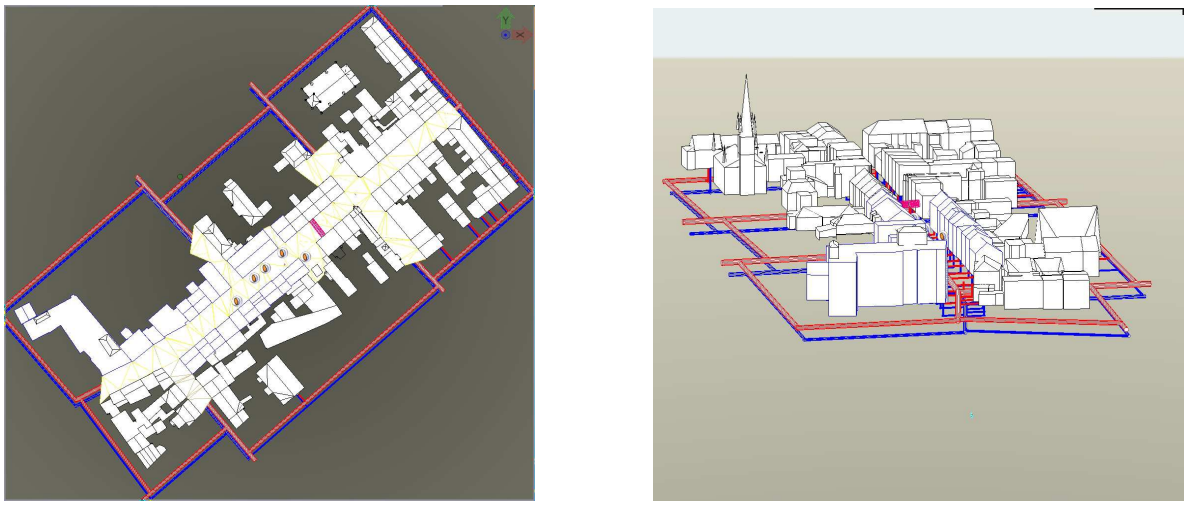

Figure 1.2 – (left) Top view of a city and the corresponding (right) 3D isometric view. The top view, which emulates what can often be seen on traditional blueprints, makes some of the elements difficult to interpret and their depth in relation to the camera is impossible to estimate.

GIS solutions, but is especially designed for mobile platforms (UMPC[4], ThoughBook, Tablet PC, smartphone). Thus, a user can access the data *dynamically* (queries, manipulations, updates, modifications, etc.) on the field or out of the office. However the interaction paradigm adopted for such devices has not really changed from those of desktop solutions involving a mouse and keys combinations. Mobile device screens are generally small. Therefore, the space for interaction is limited and prevents an accurate (and effective) reproduction of the desktop navigation environment [10]. CAD software also have followed this trend and it is now possible to navigate 3D models on mobile devices as well. It must be noted that most of these mobiles applications are more geared towards visualization and navigation than editing which still requires complex interfaces only provided by desktop solutions.

As manipulated data mainly consists of 2D and 3D models (often geolocalized[5]), tools allowing better manipulation, representation and thus, communication of this information, are necessary in order to provide more adapted and efficient tools to the different actors working on a construction site, especially those who are concerned about inspection tasks. More efficient tools should lead to improvements in the overall construction monitoring process (better error detection rate, better monitoring of the project's schedule, etc.). In order to provide such innovative solutions, new means of representation, visualization and interaction with 3D data for mobile solutions in architecture and civil engineering contexts must be proposed. Among the current efforts to meet such goal, augmented reality is a steadily growing research field that proposes promising approaches to visualize and interact with the real environment and its related data.

Augmented reality (AR) is a research area and an emerging technology related to the registration of computer generated elements to reality. The artificial elements displayed are directly related to the real objects in the physical world; these elements can be information about an object (*e.g.* the metadata related to objects in the real world – Figure 1.3 (left)), instructions about specific actions to be taken (*e.g.* highlight a specific path to follow to reach a target while driving a car – Figure 1.3 (right)) or virtual 2D or 3D models (*e.g.* 3D model of a building showing what the current scenery could look like if such a building was built at a specific location – Figure 1.4). The goal of AR is to achieve, in real time, a combination of virtuality and reality such that users of such technology could have access to a coherently unified world (*i.e.* as if the virtual elements were actually part of the real world). The final rendered image, merging both the virtual and real elements, is also called an *augmentation*.

[4] UMPC: Ultra-Mobile PC.

[5] *Geolocalized* means that the model contains meta-information about its geographic location in the real world.

Figure 1.3 – (left) Different types of metadata (restaurants, user's photos, stores, etc) displayed in Layar and (right) Wikitude Drive, an application that displays the route to follow directly overlayed on the road (based on GPS) (*source: (left): http://goo.gl/BbFEi (right): http://goo.gl/mhbQm*).

Figure 1.4 – AR as a mean to preview the scenery before the building process (*source: http://goo.gl/hQrf0*).

Figure 1.5 – (left) A streetlight selected in reality and its related meta-information; and (right) "X-Ray" vision used to see an underground pipe network (*source: (right): http://goo.gl/qoSYP*).

Mobile GIS would greatly benefit from AR. Indeed, using such a combination of technologies, a user located, for instance, on a construction site or on a street, would be able to use the AR component to select an object directly in reality[6] in order to access, for example, the object's metadata. This presents a much more intuitive way to interact with the data [11]: instead of having to look up the object on a map, the user could simply select it in the reality and the metadata related with that object would be overlaid to the display (Figure 1.5 (left) shows an example of such results being displayed over the objects – streetlamps in this specific case). This aim and click interface could be seen as a "real world GIS", where the map data would be directly overlaid to reality, upon user request. In addition, AR would allow the usual 2D data (*e.g.* blueprints) to be displayed in their full 3D CAD format and in context (Figure 1.5 (right) shows an example of this principle; a 3D model of underground pipes are overlaid over the reality to allow a user to see what is hidden beneath the surface).

However, developing an AR system is not easy to achieve: in order to perform a correct superposition of the virtual and real world elements, the system needs to know, at all time, the exact position (x,y,z) and orientation $(roll,pitch,yaw)$ of the user. The position and orientation data are used to correctly align the virtual world so that it "fits" seamlessly with the real world. That process is called *registration* and is fundamental to AR. This is why a tracking solution is required for any AR system to work. Tracking consists of retrieving the user's position and orientation by following its movements in the world. There are two distinct approaches to tackle this user tracking problem, namely *passive*, which consists in relying on available infrastructures that send information from which the user's position can be inferred (*e.g.* GPS), and *active*,

[6] In fact, the selection takes place on a live video capturing the reality. See Section 2.2 for more details on video see-through AR.

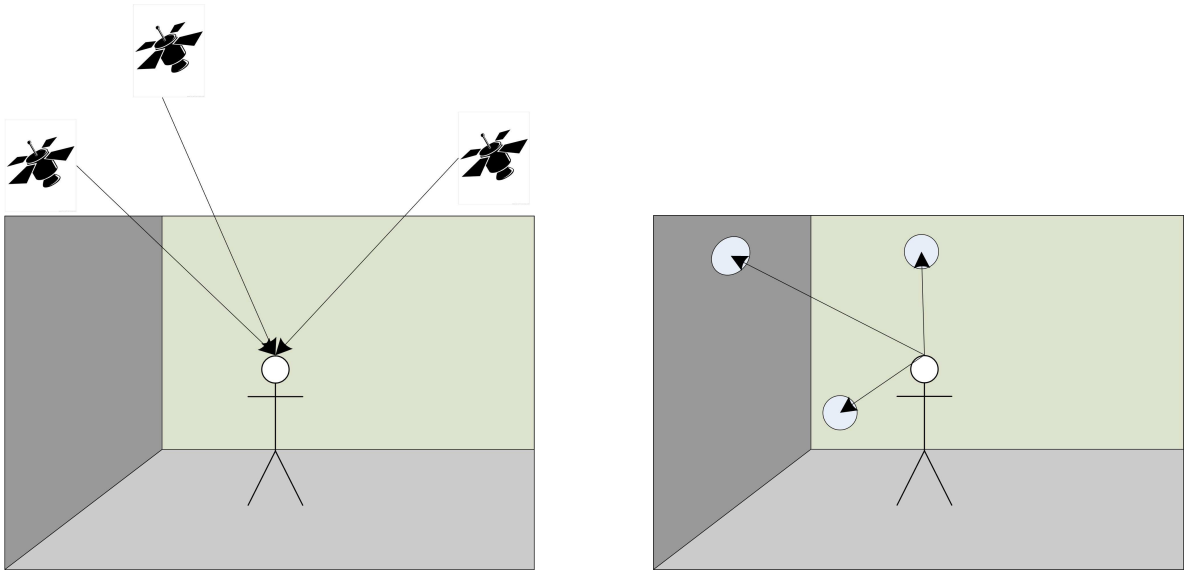

Figure 1.6 – (left) *Passive* tracking where external sources send position data to the user and (right) *active* tracking where the user's device identifies its position by analysing the environment.

which consists in relying on user's devices (camera, laser, IR, sonar, etc) that tries to identify their location by making sense of their environment (Figure 1.6). For each of these tracking paradigms, many solutions exist. A short review of the different tracking approaches will be proposed in Section 2.3.

The overall quality of the augmentation will directly depend on the chosen tracking solution to correctly overlay the virtual information over the real world image. Different tracking technologies produce different levels of accuracy and precision. Thus, not all tracking technology are suited or relevant for all AR applications. One way to categorize the different AR applications is by their accuracy and precision requirements; they span a whole spectrum and range from *mild* AR to *strong* AR[7]. *Mild AR* is generally associated with "casual" applications where the displayed data is an obvious abstraction of the elements to which they are related or are providing information about (*e.g.* a restaurant is represented by a "fork-and-a-plate" icon) and the positioning of the information is approximate (*e.g.* the restaurant icon can be positioned anywhere on the surface or the surrounding of the building's façade). Layar[8] and Wikitude[9] are both recent applications that fall into this category (Figure 1.3). While very useful and entertaining, the "mild" type of AR is not well suited for engineering-level work; the precision provided by the tracking technology tends to be insufficient. This low quality

[7]The terms *mild AR* and *strong AR* are defined by us to better illustrate the nature of a given AR application in relation to the level of accuracy of the data it manipulates.

[8]www.layar.com

[9]www.wikitude.org

of the tracking does not only affect the accuracy and precision of the augmentation, but, as the user is mobile, it heavily affects the stability of the rendering (jitter can often be observed when using mild-type AR tracking). It is worth mentioning mild AR is not regarded as an augmented reality solution in the AR community. On the other side of the spectrum lies *strong AR*. Strong AR applications often relate to information that needs to be precisely registered with the reality (for instance overlaying an as-designed 3D CAD model over the actual structure being built for an inspection or displaying meta-information about a specific infrastructure elements). In these applications, the AR system will be used to support decisions that bear major consequences. Any error induced by the system could therefore compromise projects, in part or entirely. In order to allow infrastructure professionals to take decisions based on AR systems, *strong* AR is required. Therefore, this research will only be interested in strong AR applications and its related challenges.

1.2 Problem Statement

During the last ten years, AR research work has tackled a variety of topics, among which tracking techniques, interactions techniques, calibration and registration, AR applications and display techniques have been the five main research areas [12]. Among these topics, tracking has been one of the most investigated problem for one very good reason: tracking is fundamental for AR systems. Indeed, as mentioned in the previous section, tracking is the favored option to retrieve, in real time, the position and orientation of a user. Therefore, the quality of the tracking directly influences the quality of the augmentation. Generally, there are three different, but related key issues that are considered when tackling the tracking problem in an AR context: *stability*, *coherence* and *accuracy*. A stable system ensures that the position of the virtual elements move in a consistent manner with the user's movements. This characteristic is important since virtual elements that moves around even when the user stands still greatly hinder both the experience and the usability. A coherent system ensures that the virtual elements does not look "out-of-place" (*e.g.* overlaying a 3D model of a house over the corresponding real house in such a way that no visible offset between the two exists – any visible offset would break the illusion of an "unified" world). Finally, an accurate system displays the elements at their *real* position and orientation in the world. For example, if the virtual element *should* be located 30 cm to the right, as specified by its geolocalization, it will be displayed at the correct location. It might mean that an offset between the real element and its virtual counterpart will be displayed, affecting the coherence aspect. Using an accurate AR system, offset between the real element and its virtual counterpart (i.e. incoherent visualization) will allow detecting differ-

ences between the as-built and as-planned item (see Figure 1.7 for an illustration of the difference between the *coherence* concept and the *accuracy* concept).

Engineers and architects are professionals that typically take decisions that bear important consequences; bad decisions could easily compromise part or the entire project (that can cost both important amount of time and money) or even compromise public safety. Therefore they first and foremost requires *accuracy* out of their systems to make sure the project is conducted as planned. The accuracy of the augmentations (which is directly related to the tracking accuracy) is therefore the main criteria a mobile AR GIS solution should fulfill to be able to support engineering-level decisions [1]. However, achieving such accuracy with geolocalized, 3 dimensional data (*e.g.* 3D CAD models) faces several challenges, the principal ones being listed below:

Mobile tracking limitation Very high accuracy augmentation on mobile devices evolving in unprepared outdoor environments[10] cannot currently be achieved easily (see Section 2.3). This is due to the current limitations of the majority of available tracking solutions and to the high accuracy need of architecture and geo-engineering fields. There are some tracking technology that allow a very high level of precision and accuracy, but these are either too constraining in terms of environment control (the environment needs to be calibrated and prepared beforehand) or are simply unavailable indoor.

Hardware limitation In order for an AR GIS platform to be accepted and adopted by architects and engineers operating on construction sites, it must be usable with already available technological means. That means that, ideally, the platform should avoid specialized (*i.e.* costly) components as much as possible. If positive results can be obtained from a low-cost platform as of now, it proves that a mobile AR GIS platform should be sustainable in the future as well. The platform should also be mobile; this add constraints on the hardware selection (*e.g.* bulkiness, heaviness, calibration requirements, availability, etc.). A good way to respect these limitations is to favor already integrated technologies, such as tablets and smartphones which come already equipped with cameras (even if they offer low resolutions).

Accuracy requirements Engineering and architecture work includes a very wide variety of tasks. Each of these tasks will have varying level of accuracy requirements. Moreover, AR may not be an appropriate support for all these tasks. Shin *et al.* [1, 13] identified a subset of engineering tasks that could benefit from AR. In

[10] Unprepared environments are environments that have not been modified in order to use a tracking solution (*i.e.* no markers have been installed, no additional infrastructure has been set up, etc.).

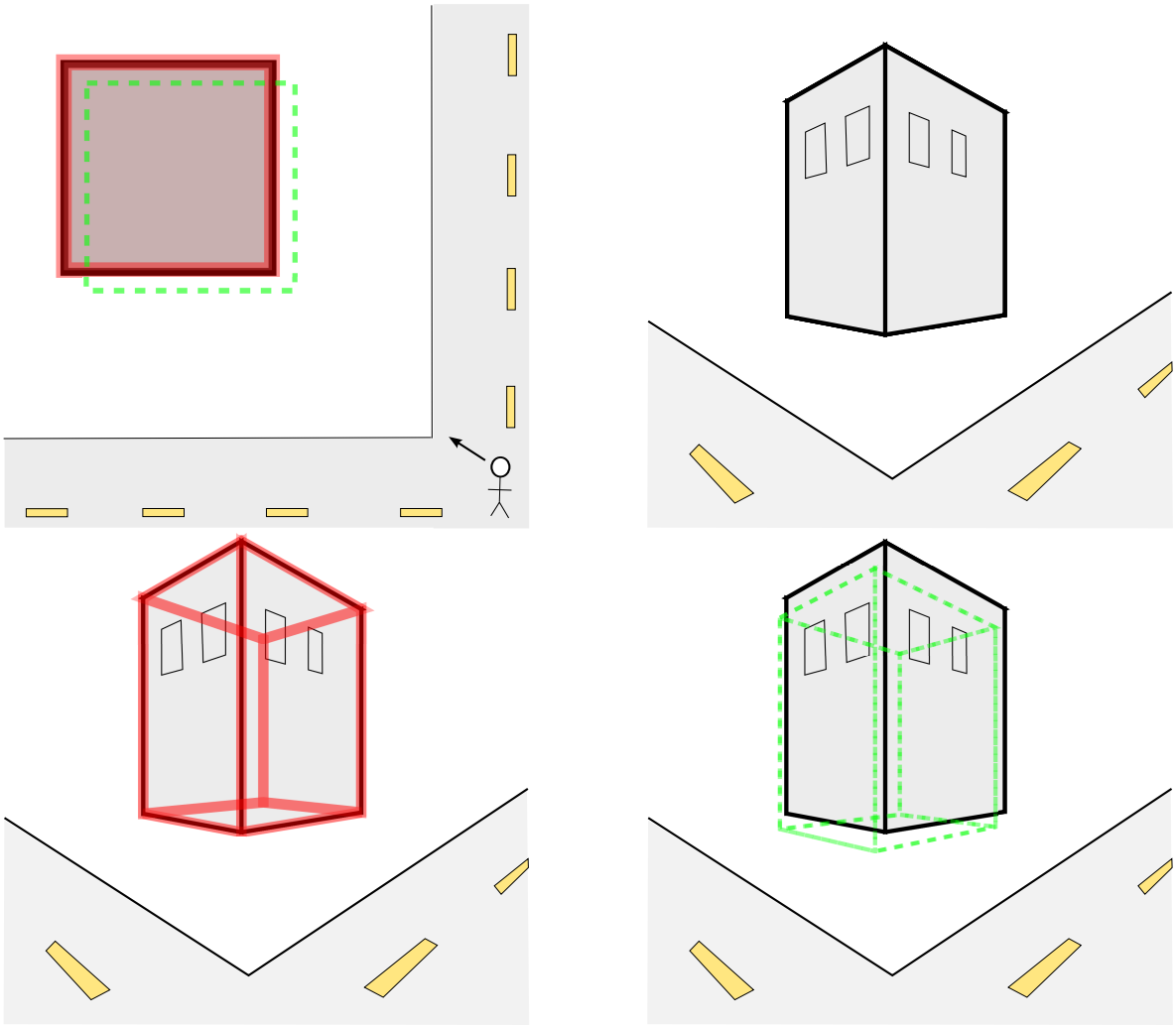

Figure 1.7 – Illustration of what is understood by *coherence* and *accuracy* in terms of AR. (top left) Bird's view of a street and building being augmented. The user is represented in the bottom right corner of the illustration and is currently looking at the building. The *black* square represents the actual, physical building (as-built). The *red* square is a coherent augmentation, that is, perfectly aligned with the as-built structure. The *green* dashed square represents the as-designed, geolocalized model that is the accurate representation. In this example, and to better illustrate the difference between "coherent" and "accurate", there are differences between the as-designed and the as-built. (top right) The scene from the perspective of the user when looking at the building with his naked eyes. (bottom left) The user perspective of a coherent augmentation. (bottom right) The user perspective of an accurate augmentation; in this illustration, the user is able to see errors (offsets) that took place during the construction process. (*note: (top left): The user should be represented from a bird's view to be coherent, but it was left in side view for the sake a clarity.*)

these studies, quantitative information about the level of accuracy that engineering work requires has also been documented. However, no accuracy level threshold for a general-use mobile AR GIS has been specified. This threshold is required in order to design AR platform aiming to support decisions in high-accuracy fields like engineering and architecture.

Factors influencing the accuracy level Accuracy is likely to be influenced by multiple factors. Indeed, the environment in which architects and engineers operate are not always ideal for tracking (*e.g.* the environment might be dynamic – passersby, cars, machinery, etc. – thus decreasing the accuracy of an AR system using computer vision). Moreover, these factors might be strongly linked to the approach that is used to perform the augmentation of the real world. As accuracy has not been a main area of focus in AR, the factor influencing this metric have not been clearly determined. To propose an accurate approach to mobile AR, these factors must be inventoried and their relative importance weighted.

As it has been highlighted in Section 1.1, there is a need for mobile GIS solutions that are specifically adapted to engineering and architecture tasks on the field. AR appears like a promising approach to answer this need. However, as of now, there is no available solution on the market mainly due to the high accuracy need of these fields of application. One of the main components of AR systems is tracking. This research aims to explore tracking solutions in order to propose a mobile AR GIS solution adapted to engineering and architecture monitoring tasks. Before being able to design such solutions, it is necessary to determine which tracking approaches are able to meet the requirements of these tasks. This is the problematic of the proposed research. In other words, the main question to be answered is the following:

which tracking approach(es) is/are the most promising towards the design of mobile AR GIS dedicated to architecture and geo-engineering fields given the high accuracy required, the lack of specification of the targeted accuracy, the outdoor mobile tracking context, and the numerous factors that can affect this accuracy?

1.3 Objectives

The main objective of this project is to provide an answer to the problem stated in the previous section, that is:

to define the accuracy requirements of an AR mobile GIS in a geo-engineering context and list the technical approach(es) that are most suited to achieve this accuracy level. This main objective can be broken down in three sub-objectives described in the next subsections.

1.3.1 Sub-objective 1: specifying the accuracy requirements

As mentioned previously in Section 1.2, accuracy requirements vary depending on the task at hand. Before being able to evaluate the adequacy of tracking approaches for mobile AR GIS in geo-engineering contexts, the accuracy requirements of these mobile systems need to be specified in such a way that these requirements can be used to evaluate the system independently of the task being conveyed. The determination of these accuracy requirements aims at ensuring the accuracy of a tracking approach is able to meet the specifications of the task or, if it does not, quantify by how much it differs.

1.3.2 Sub-objective 2: determining suitable tracking approaches for mobile AR GIS dedicated to geo-engineering

This sub-objective aims to determine, among the current tracking methods available, which ones present good potential for achieving mobile AR GIS solution compliant with the geo-engineering context (i.e. meeting the accuracy requirements specified through sub-objective 1 and allowing architects or civil engineers to conduct inspection tasks on the field).

1.3.3 Sub-objective 3: validating the selected potential tracking approaches

This last sub-objective aims at validating the potential of the proposed tracking approaches. This last step will ultimately yield recommendations on which approach(es) is/are suitable for a mobile AR GIS or what prevents it/them to be appropriate in the geo-engineering context of this research.

1.4 Methodology

The global methodology adopted to conduct this research is one of engineering. Engineering is known for its straightforward method: acquire needed knowledge, generate ideas and formulate hypotheses, verify these hypotheses and knowledge with prototypes and evaluate the results produced. There are a certain number of iterations done until the project gets to a satisfactory state or the formulated hypotheses gets confirmed or refuted.

This section describes the various steps involved in the approach designed to meet the three sub-objectives presented above, and therefore the project's global objective. This approach is synthesized through the workflow diagram display in Figure 1.8.

The two first steps are dedicated to the completion of sub-objective 1. They consists of, respectively, a literature review and the determination of a scale-invariant accuracy representation that takes into account the multiple variables defining the accuracy requirements of a task. The literature review aimed mainly at finding answers to the various issues with relation to accuracy requirements of AR in a mobile GIS context. The review focused on work describing the different tasks that must be accomplished on a construction site and their accuracy requirements, specifically in a AR context. A scale-invariant accuracy representation was then proposed as a way to abstract the different factors influencing the accuracy requirements of a given task.

The two following steps are dedicated to the fulfillment of sub-objective 2. They consist of a literature review of the different tracking solutions available as well as a selection, among these solutions, of those that present potential for high accuracy AR. The literature review was conducted in the fields of indoor and outdoor tracking, pedestrian navigation and geolocation, especially applied to AR. It allowed to identify the different tracking solutions currently available along with their respective accuracy. Based on this metric, approaches were selected for further investigation in this research.

Finally, sub-objective 3 is based on *agile* software development methodologies[11]. It consists of the typical agile cycle: design, implementation and revision. Once a given approach was selected for investigation, the first step was to specify the performance and accuracy objectives of the current approach based on the accuracy requirements set with the sub-objective 1. It then led to the design of both the software architecture

[11] "Agile Software Development is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams." [14]

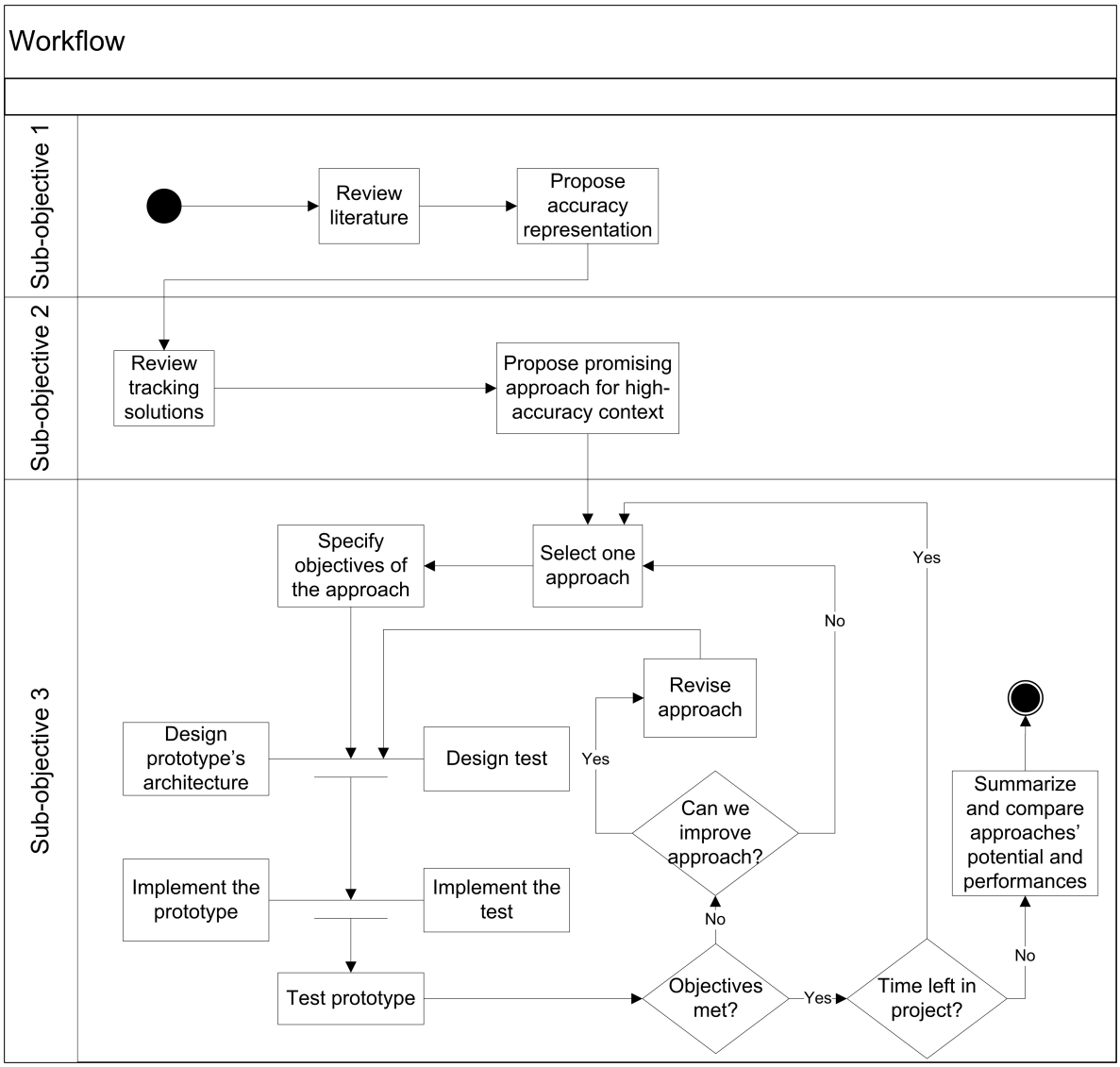

Figure 1.8 – Workflow diagram of the activities associated with each sub-objective

and the tests that were used to validate wether the approach met or not the objectives. The solution and the tests were then implemented. If the objectives were met and the time allocated for the project has not been exceeded, another approach could be considered for investigation. On the other hand, if the objectives were not met, some further investigation was required in order to see if any improvement could be made to the given approach. If some improvements could be identified, they were implemented and tested or suggested as future works.

1.5 Structure of the thesis

Chapter 2 will present the literature reviews conducted to meet sub-objective 1 and 2. These reviews identify the construction site related tasks that could benefit from AR and their respective accuracy requirements. They also propose the different available tracking solutions. The ones that are compliant with mobile AR GIS for geo-engineering are identified in this chapter.

Chapter 3 and Chapter 4 concern the work carried out to meet sub-objective 3. Chapter 3 will present the first prototype proposed in this research project: Orientation Tracker. The related work will be detailed before describing the principles of the method. The result of the prototype evaluation will then be presented. The chapter will conclude on the different identified factors that influence negatively the accuracy of the method. Chapter 4 will present the second prototype proposed in this research project: Panorama-Based Pose Refinement. The approach will first be presented and justifications for departing from an online tracking method will be provided. The method will then be presented in details before presenting and analyzing the results of its evaluation. The chapter will conclude by discussing the potential of this approach for high-accuracy AR.

Finally, Chapter 5 will propose the different conclusions drawn from the experiments presented in Chapter 3 and 4 and suggest future work.

Chapter 2

Literature Review and Mobile AR Concepts

AR is a relatively broad field of research. It relates, among others, to computer vision, robotics, visualization, positioning, artificial intelligence, and therefore makes use of many different concepts. This chapter mainly presents the different key concepts that will be involved or influence the selection of tracking solutions suitable for mobile AR GIS dedicated to geo-engineering contexts.

The first part of the chapter will focus on the typical tasks taking place on a construction site. Their accuracy requirement will be reviewed and a unified accuracy representation model will be provided towards the selection of the most suitable tracking solutions for mobile AR GIS. The second part of the chapter will be dedicated to AR. Formal definition of AR and mobile AR will be provided and the constraints brought forth by the mobility to the AR challenge will be underlined. Since tracking is at the core of the AR challenge, the third part of the chapter will provide a review of tracking methods. Based on this review, the most promising tracking approaches for mobile AR GIS will be determined. Finally, the chapter will conclude with the list of tracking solutions that will be further investigated.

2.1 Mobile AR GIS Compliant Tasks and Accuracy Requirements

As stated in Chapter 1, high accuracy is required for mobile AR GIS dedicated to different applications such as construction site monitoring or infrastructure inspection. This section aims at defining in a more tangible and quantitative manner these accuracy requirements (*i.e.* fulfillment of sub-objective 1). First, the geo-engineering tasks that could benefit from a mobile AR GIS will be presented. Then, a formal definition of the terms *precision* and *accuracy* will be provided to better highlight the different implications of each term. Because this research aims to fulfill quantitative objectives, the accuracy requirements of the different selected tasks will be quantified. Shin *et al.* [1, 13] did an extensive investigation of these issues. Therefore, the list of tasks benefiting from AR and their accuracy requirements are mostly a synthesis of their work. Finally, as it is anticipated that the required accuracy depends on different factors (*e.g.* the nature of the task, the scale of the infrastructure, ...), the accuracy requirements for a mobile AR GIS will be expressed in a unified, task-independent representation. This unified representation is necessary in order for the mobile AR GIS solutions to be evaluated on the same bases for multiple tasks, taking place at different scales.

2.1.1 Mobile AR GIS compliant tasks

Before addressing the accuracy issue and the expected performance of a mobile AR GIS on a construction site, the different tasks that can be achieved using a mobile AR GIS on such a construction site and their definitions need to be reviewed. Shin *et al.* carried out such a review and evaluated which of them might be improved by introducing AR into the workflow[1]. The tasks compliant with a mobile AR GIS system and their definitions are presented in Table 2.1.

These tasks are very diverse and require varying level of accuracy. These will be quantified in Section 2.1.3 after presenting the formal distinction between the *accuracy* and *precision* terms as this distinction is crucial to this research project.

[1] While Shin *et al.* selected only a small subset of the presented tasks, it is highly probable that most of these activities could benefit, in one form or another, of AR. Nonetheless, their selection represents a good variety of tasks and gives a good overview of the different accuracy needs on a construction site.

Work task	Definition
Layout	Determining, ascertaining, and marking dimensions
Excavation	Breaking up, turning over, removing, or filling soil
Positioning	Moving heavy objects to certain locations and orientations for installation
Inspection	Examining installed workmanship by an approved state or city personnel to verify quality and that the work installed to the pre-approved drawings and that the work meets all codes
Coordination	Organizing and determining upcoming work flows or resource allocation
Supervision	Seeing if work is performed as planned
Commenting	Conveying supplementary information regarding a task
Strategizing	Figuring out the detailed procedures for specific tasks

Table 2.1 – Definitions of construction-site related tasks that could benefit from the introduction of AR in the workflow [1].

2.1.2 Precision and accuracy definition

Precision and accuracy are two related but actually different concepts. Because this project put such a high emphasis on accuracy and precision, it is important to clearly highlight the differences between these two notions. The *Oxford English Dictionary* describes precision and accuracy as follows:

Precision The reproducibility or reliability of a measurement or numerical result; a quantity expressing this. [15]

Accuracy The state of being accurate (Exact, precise, correct, nice; in exact conformity to a standard or to truth). [16]

A visual representation of these definitions is provided using the target analogy (Figure 2.1).

Ideally, a system should be both precise *and* accurate. Precision without accuracy will result in a user *consistently* making decisions that are off. On the other hand, an accurate but imprecise system will result in a user making, *on average*, a correct decision. Mathematically speaking, precision is related to the standard deviation of a set

Figure 2.1 – The *Precision Vs. Accuracy* target analogy. (left) High accuracy and low precision and (right) high precision and low accuracy (*source: (left): http://goo.gl/tH1u2 (right): http://goo.gl/X8fiH*).

of values and the accuracy is the mean's bias compared to its true value. The problem is that decisions are individual actions that are not averaged over time. Therefore, the said system will always be off, even if, on average, the decision is correct.

2.1.3 Absolute accuracy requirements

There are mainly two concepts that must be considered when quantifying the accuracy requirement of a given task. The first one is the actual *accuracy* requirement of the task, in an absolute reference system, thus expressed in meters (*e.g.* the layout task requires a very high accuracy of 1 mm – below this threshold, the dimension measurements and validation are not considered reliable enough). The second concept is the *distance* between the observer (user) and the observed object when the task is being conveyed (*viewing distance)*. Indeed, each task typically needs to be executed at a given distance from the observed object (*e.g.* it would not make sense for the supervisor of a machinery-operated excavation activity to stand at 50 cm from the place where the hole is being dug). This second concept is central to the determination of a unifying accuracy and precision representation and will be explained further (as well as its implications) in Section 2.1.4.

As the absolute accuracy requirements and the viewing distance are interrelated, Shin *et al.* combined both metrics in a matrix they called "Matrix of Operational Spaces for Work Tasks"; it is reproduced in Table 2.2.

All the tasks span a very wide spectrum, both in term of viewing distance, ranging from 0.5 m to 108 m, and accuracy requirements, ranging from 1 mm to 50 mm. This raises an issue when trying to specify the accuracy requirements of a mobile AR GIS system. Indeed, expressing accuracy using two metrics makes it difficult to evaluate a system that must be able to convey multiple tasks because it offers no threshold to

View Dist. (m)→	Reachable (0-0.5)	Very short (0.5-3)	Short (3-6)	Medium (6-18)	Long (18-60)	Very long (60-108)	Extremely long (+108)
Req. Acc. (mm)↓							
Ultimate (0-1)	Layout Positioning Inspection						
High (1-6)		Positioning Inspection					
Intermediate (6-13)			Positioning Excavation Inspection	Excavation Inspection	Inspection		
Low (13-25)				Positioning			
Very low (25-50)				Coordination Supervision Commenting Strategizing	Coordination Supervision Commenting Strategizing	Inspection Coordination Supervision Commenting Strategizing	Inspection
Extremely low (+50)				Coordination Supervision Commenting Strategizing	Coordination Supervision Commenting Strategizing	Coordination Supervision Commenting Strategizing	Coordination Supervision Commenting Strategizing

Table 2.2 – Matrix of operational spaces for work tasks as presented in [1]. Each task is categorized by the viewing distance (view dist.) it must be conveyed at, expressed in meters, and by the required accuracy (req. acc.), expressed in millimeters.

assess if the system fulfills the *overall* requirements of engineer and architects. These values need to be expressed in a way that a single threshold in terms of accuracy and precision can be adopted as the overall accuracy need for an mobile AR GIS system. The following subsection proposes a solution to this issue.

2.1.4 Scale invariant accuracy representation

As illustrated in the previous section, the accuracy requirement of a task is specified for a given viewing distance. It makes the comparison between the accuracy requirement of tasks difficult. Indeed, assessing if achieving a 50 mm accuracy at a viewing distance of 100 m is more difficult than achieving an accuracy of 1 mm at a distance of 0.5 m is not straightforward. Thus, having a single metric to encompass both the accuracy and distance variables would facilitate the accuracy comparison of various tasks.

In this research's context, the accuracy of a given AR system is expressed as the maximal *distance* error that is supported by a given task above which the task's integrity is compromised. As specified in Section 2.1.3, any task must be realized at a given distance from the main point of interest of the said task (*e.g.* the main point of interest of an excavation task is the location where the hole is being dug). For simplicity, the camera's position is also assumed to be fixed (*i.e.* static). In this context, the ratio between the required absolute accuracy and viewing distance can be expressed as an *angular offset* because the camera's only degrees of freedom (DOF) are its orientation. An angular value is appropriate for comparison purposes since it is scale independent. Thus, a given task having an accuracy requirement (ϵ m) executed at a distance (d m) from the task's main point of interest can be converted into an angular representation (θ) using Equation 2.1 (illustrated at Figure 2.2).

$$\theta = \arcsin\left(\frac{\epsilon}{d}\right) \tag{2.1}$$

Typically, mobile AR is carried out using handheld devices and video see-through approaches (see Section 2.2). That is, the augmentation is displayed on a *representation* of reality captured by the device's camera. Therefore, it is sometime useful to express the accuracy in terms of *pixels*[2] as it is common to every video see-through mobile AR system and it is the smallest data unit used in imagery. It is especially helpful, for obvious reasons, if a tracking component of the mobile AR system is based on

[2]Pixel: "a single point in a raster image, or the smallest addressable screen element in a display device; it is the smallest unit of picture that can be represented or controlled." [17]

computer vision. The conversion between angular accuracy value (θ) and pixel value (τ) is achieved easily by using the pinhole camera model. The first step is to compute the field of view (FOV) of the camera. The FOV of the camera (ϕ) at a given focal length (f) can be calculated using Equation 2.2 (illustrated at Figure 2.3). The focal length (f) can be obtained through camera calibration[3]. Finally, the angular error tolerance (θ) can be expressed in pixels (τ) using Equation 2.3, where ρ represents the resolution of the captured image (in pixels) and ϕ is the previously determined angular field-of-view of the camera (illustrated at Figure 2.4). The opposite conversion (from pixel error tolerance to angular error tolerance) can be obtained trivially from Equation 2.3 resulting in Equation 2.4.

$$\phi = 2 \times \arctan\left(\frac{\rho/2}{f}\right) \tag{2.2}$$

$$\tau = \rho \times \frac{\theta}{\phi} \tag{2.3}$$

$$\theta = \phi \times \frac{\tau}{\rho} \tag{2.4}$$

The conversion of the accuracy requirement from a distance tolerance into a angular tolerance has been applied to all the tasks described previously. Table 2.2 proposed a categorization of each task given their accuracy requirements and their viewing distance. A similar table is presented at Table 2.3. However, each cell has been filled with a converted accuracy requirement expressed in angular tolerance (and example of corresponding pixel tolerance calculated for a generic web camera[4] in parentheses)[5].

The different values shown in Table 2.3 confirmed the intuitive notion that, for a given accuracy requirement, the further away the user is from the observed object and the more precise and accurate the mobile AR GIS solution should be. In this

[3] "Camera resectioning is the process of finding the true parameters of the camera that produced a given photograph or video." [18] Camera resectioning is often called "camera calibration". The true parameters of the camera are used to make the correspondence between world 3D coordinate system and the 2D projection of the image (in pixel coordinates). Thus, it becomes possible to associate real-world dimensions to a pixel. See Appendix A for details on the pinhole camera model and its parameters.

[4] These results have been computed assuming a camera with 640×480 pixel resolution and 721 mm focal length.

[5] For each cell, the mean value of both ranges (*viewing distance* and *accuracy requirement*) are considered for the calculation. Thus, for a range of [3,6], the 4.5 value is used.

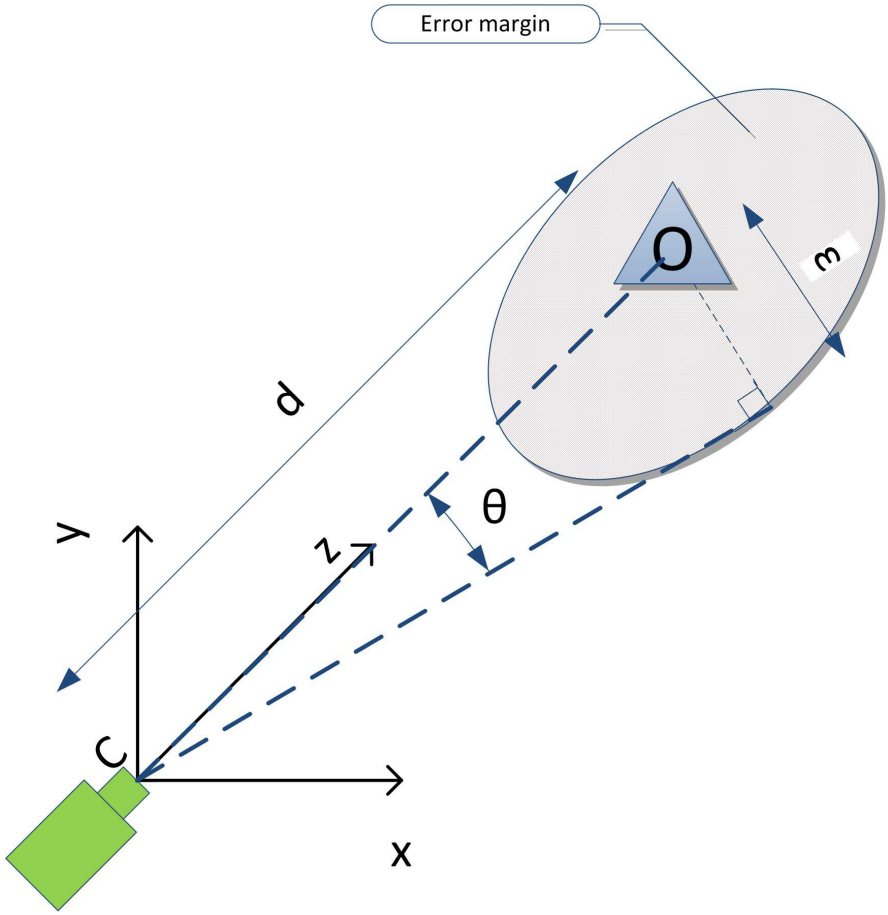

Figure 2.2 – Angular error tolerance illustration. A camera C is located at a distance d from the point of interest O where the task is being conveyed. ϵ represents the task's accuracy requirement and θ is the corresponding angular error tolerance. The red circle represents the error margin; it is here represented by a circle on the XZ plane for the sake of clarity, but the error can be aligned along any direction in the 3D space.

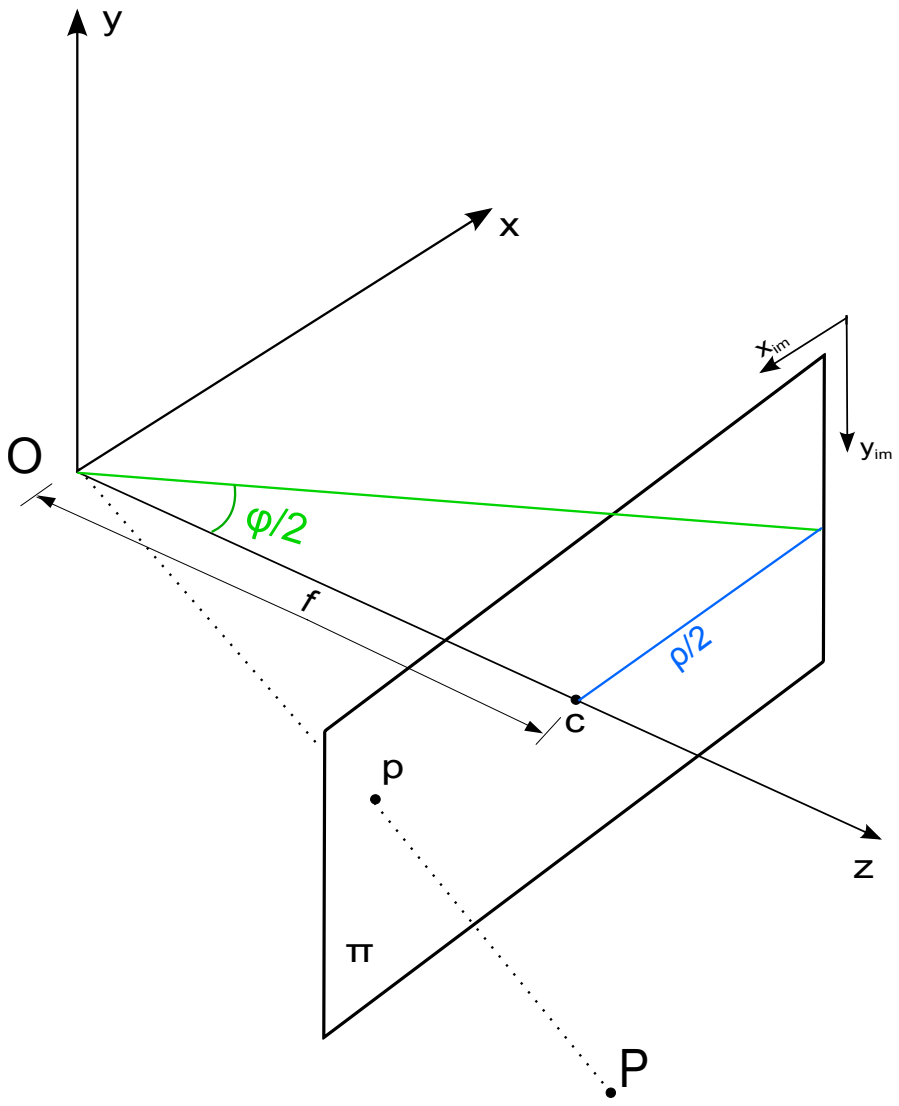

Figure 2.3 – FOV determination based on the pinhole camera model. The model consists of an image plane π and a 3D point O, representing the *camera center* or the *focus of projection*. The positive z axis that goes from O through the center c of the image plane is called the *optical axis*. A 3D point in the world P is projected on the image plane π at point p. ϕ is the angular FOV of the camera, f is the focal length of the camera and ρ is the resolution (the total number of pixels in each of the image's dimensions). Only the horizontal FOV is represented on this figure for readability purposes. The same principle applies for the vertical FOV. x_{im} and y_{im} represents the coordinate system of the image.

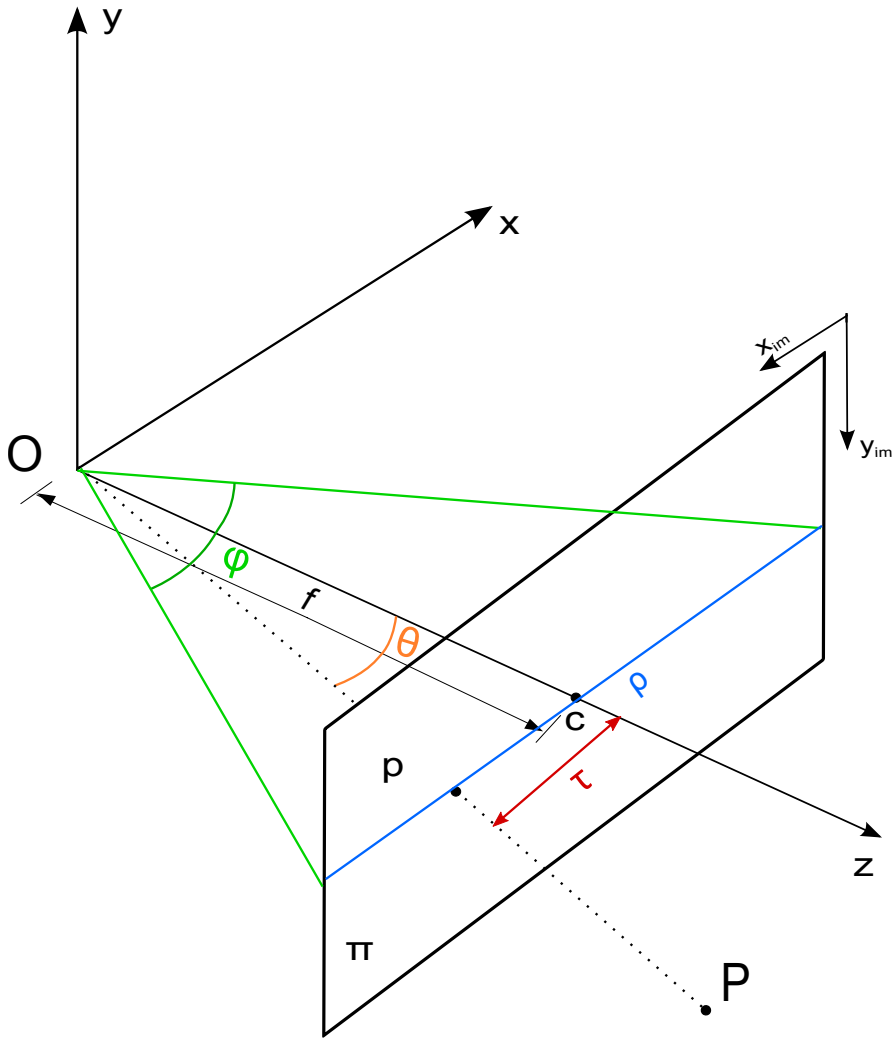

Figure 2.4 – Conversion between angular measurements and pixel measurements based on the pinhole camera model. The model consists of an image plane π and a 3D point O, representing the *camera center* or the *focus of projection*. The positive z axis that goes from O through the center c of the image plane is called the *optical axis*. A 3D point in the world P is projected on the image plane π at point p. ϕ is the angular FOV of the camera, f is the focal length of the camera, θ is an angular measurements with its projected length (τ), in pixels, on the image plane and ρ is the resolution (the total number of pixels in each of the image's dimensions). Only the horizontal FOV is represented on this figure for readability purposes. The same principle applies for the vertical FOV. x_{im} and y_{im} represents the coordinate system of the image.

VIEW DIST. (m)→	Reachable (0-0.5)	Very short (0.5-3)	Short (3-6)	Medium (6-18)	Long (18-60)	Very long (60-108)	Extremely long (+108)
REQ. ACC. (mm)↓							
Ultimate (0-1)	0.115° (1.523 px)						
High (1-6)		0.115° (1.523 px)					
Intermediate (6-13)			0.121° (1.617 px)	0.045° (0.606 px)	0.014° (0.187 px)		
Low (13-25)				0.091° (1.213 px)			
Very low (25-50)				0.179° (2.394 px)	0.055° (0.737 px)	0.026° (0.342 px)	0.020° (0.266 px)
Extremely low (+50)				0.239° (3.192 px)	0.073° (0.982 px)	0.034° (0.456 px)	0.027° (0.355 px)

Table 2.3 – Matrix of Operational Spaces for Work Tasks (Table 2.2) displaying error tolerance expressed in pixel (values in parentheses are the corresponding angular tolerance). Values have been computed with the following camera setting: 640×480 pixels ; focal length: 721 mm.

proposed representation of accuracy, the ratio between accuracy and viewing distance is more determinant than the absolute accuracy requirement of the task. It is also worth observing the required accuracy is *very high*. The less demanding tasks (accuracy of 50 mm at a viewing distance of 12 m) still requires an accuracy of 0.239° (3 pixels) while the most demanding task (accuracy of 9.5 mm at a viewing distance of 39 m) requires a 0.014° (0.2 pixel) accuracy. As a point of comparison, a relatively high-accuracy orientation sensor (XSens' MTx – see Section 2.3.1.3) can achieve an accuracy of $\approx \pm 1°$; this is a little less than five time the minimum accuracy requirement of the less demanding task. This confirms that high accuracy AR is a difficult problem that requires novel solutions.

Relying on Table 2.3, it can be observed that the accuracy requirements of a mobile AR GIS solution will vary in function of the engineering task at hand. Summarizing the values of Table 2.3 in relation to the viewing distance: tasks that must be realized at relatively short viewing distances (0-18 m) require a system having an approximate accuracy of 0.1° and tasks realized at a larger distance (18+ m) require, on average an accuracy $< 0.05°$.

2.2 Mobile AR Concepts

AR is a concept where virtual elements or objects are superposed and correctly registered to actual elements in the reality to form a seamless and coherent world. Azuma [19] proposed to define an AR system or application as a solution fulfilling the following three criteria:

1. Combines real and virtual
2. Interactive in real time
3. Registered in 3-D.

Not all AR systems can be considered *mobile*. Scholl *et al.* [20] proposes a definition for mobile AR:

> Instead of being represented by an avatar like in many traditional computer games, players are their own avatar moving in the game world by moving

in their real environment.[6]

Mobile AR system are typically comprised of a few hardware components: display, tracking, input and computer components. The display is the component where the augmentations are rendered. They most often consist of either head-mounted displays (HMD) or handheld devices (*e.g.* smartphone). The tracking component consists either of dedicated tracking hardware (*e.g.* GPS) or a combination of hardware and software (*e.g.* camera hardware used in conjunction with computer vision techniques) that aims to estimate the user's (or device's) position and orientation. The input component can be used to interact with the augmentations. These interactions can be realized by manipulating the elements directly on the display (*e.g.* touchscreen) or by using dedicated hardware (*e.g.* special gloves that can be tracked in space). Finally, the computer component is used to analyze the data captured by the system (*e.g.* camera, GPS, infrared, sonar, etc.) to synthesize and position augmentations.

In order to render an augmentation (*i.e.* merge the virtual elements and the real world), an AR system can use one of two main strategies: video see-through and optical see-through. Optical see-through systems employ half-silver mirrors to reflect the overlay information into the user's eye. Most of the time, these systems use head-mounted displays (HMD) (Figure 2.5 (left)). Video see-through systems, on the other hand, capture the reality through a camera and the captured image is then augmented with correctly positioned virtual elements before being fed to the screen for visualization (Figure 2.5 (right)[7]). All handheld devices use video see-through for AR since they are equipped with standard resolution camera. Also, most mobile AR systems, nowadays, are based on handheld devices.

Building a mobile AR system imposes constraints on different components of the system. The first challenge concerns the user or device tracking: in order to be able to merge virtual elements with the real world, the system needs to know the camera pose (*i.e.* the position and orientation of the camera capturing reality) at all time during the augmentation phase. This is not a trivial task, especially in a mobile context, where the user should be able to move freely and anywhere he needs to use the system. This challenge is further heightened if a handheld platform is used. Handheld devices are

[6]Note: [20] is a paper that has been written in the context of mobile AR *games*. This definition still is accurate for our application context by substituting the terms "player" and "game world" by "user" and "workspace world" respectively.

[7]Figure 2.5 presents video see-through display as a HMD to better illustrate the difference between *optical see-through* and *video see-through*. Video see-through can be, and most of the time is, achieved on handheld devices where the reality is captured by the integrated camera and the display is done on the actual screen of the device (see Figure 1.3 for examples of video see-through AR realized on handheld devices).

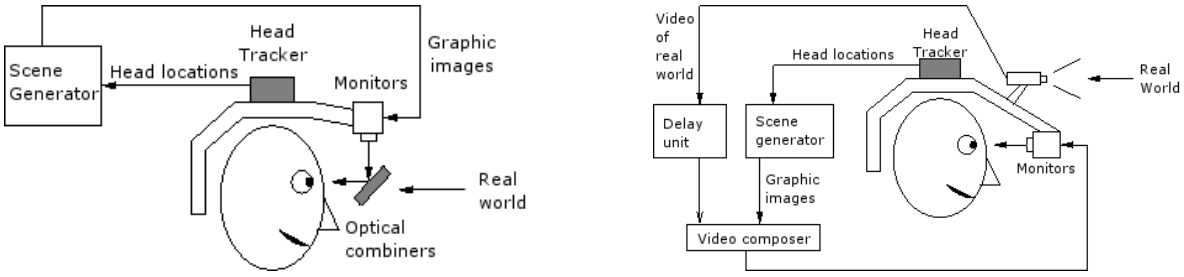

Figure 2.5 – Diagrams illustrating the different components and organization of (left) an optical see-through HMD and (right) of a video see-through HMD (*source: (left): http://goo.gl/IWQu3 (right): http://goo.gl/WwVhu*).

not very stable, mainly because these devices are hand-held and it is difficult and/or counter-intuitive for any user to keep a steady hand for any extended length of time. This context requires more adaptability and robustness out of the tracking solution.

Another component impacted by the mobility of the system is the camera. As one of the criterion of AR states that the system must be interactive in real-time (see Azuma's definition [19]), the camera must be able to sustain a minimal framerate[8]. The capability of a mobile AR system to sustain the required framerate depends on the selected tracking solution. For example, a tracking solution based on computer vision requires more processing time than a hardware-based tracking method since each frame has to be processed individually in order to extract the camera position and orientation that will be used to align the virtual and real worlds. Hardware-based tracking solutions generally feed directly the position and orientation information to the mobile AR system. Thus, in such applications, the device's processing time is mainly dedicated to the augmentation step.

Finally, any system relying on computer vision techniques for tracking the camera pose must be robust to the different "difficult" conditions inherent to outdoor mobile systems. For example, difficult illumination settings like direct sunlight, scenes that are too dark or unevenly lighted might compromise the analysis of the system in some portion of the image. Also, the system has to tackle occlusion problems. That is, there might be some object(s) that occlude, completely or partially, the scene that the system tries to track. Lastly, fast motion can be a hurdle. When the user changes the viewpoint of the camera too quickly (*i.e.* quickly enough so that the camera's framerate is not sufficient), motion blur will be added in the captured image. This blur may compromise the tracking, especially if the system uses some primitives that are sensible to blur (*e.g.*

[8]To be considered "real-time", a given framerate should not be below 15 frame per second (fps). This threshold must be adapted to each application context; when rapid motions need to be captured by the video stream, the framerate should be higher (30-60 fps or more) in order to avoid "skipped frames" in the user's perception.

corner primitives [21]).

A review of the available tracking solutions is provided in the next section.

2.3 Tracking

Tracking is one of the fundamental issues of AR. Changes in the viewer's position and orientation need to be tracked in order to properly render the graphics according to his field of view. Six degrees of freedom (DOF) tracking (3 DOF for position and 3 DOF for orientation) is a difficult problem. To provide a nice augmentation experience, tracking should be carried out in real-time and ideally no prior knowledge of the environment. Moreover, the review of the different tracking solutions presented in this section demonstrates that the difficulty inherent to the measurement of the viewer's position tend to be proportional to the level of precision and accuracy needed. Indeed, in order to increase the precision and accuracy of a tracking solution, the hardware needs to be more specialized, the algorithms (real-time or post-processing) are getting more complex and precise and/or a more thorough calibration process might be required. Tracking is an ongoing field of research on its own and has still many unresolved issues. There are mainly two different, but often complementary, categories of approaches in order to assess or track the changes inherent to the position and/or orientation of the user: *passive* solutions and *active* solutions. The following sections describe the methods belonging to these two categories respectively.

2.3.1 Passive solutions

Passive tracking solutions consist of using a given "signal" from an outer source in order to derive from it the position and/or orientation of a device. The signal can belong to various domains like radio-frequencies (RF), sound, laser, magnetic fields, gravity, etc. Many different tracking systems relying on such signals are available nowadays. This subsection will address only the systems that could be considered in a mobile AR solution. These are: Global Navigation Satellite System (GNSS), urban and indoor alternatives to GNSS and MicroElectroMechanical Systems (MEMS).

2.3.1.1 GNSS

GNSS is a positioning system based on the signal emitted from a series of orbiting satellites. The most well-known implementation of these systems is the Global Positioning System (GPS), which was developed and still is maintained by the United States government. GNSS is by far the most popular tracking solution addressing the *positioning* problem (GNSS is unable to provide any orientation information). This technology is based on the trilateration[9] of a minimum of 4 simultaneous pseudorange[10] measurements. Thus, a GNSS receiver must detect the emitted signal of at least 4 different satellites simultaneously. Pseudorange is calculated using the time-of-flight (TOF) of the GNSS signal (the time the signal takes to reach the receptor once broadcasted by the satellite). GNSS satellites and receivers use very accurate clocks in order to make these time estimates. Even so, it is still essential to estimate the clocks' error. This is crucial as an error of one microsecond ($1.0 \times 10^{-6}\, s$) yields to an absolute position error of about 300 meters[11].

The precision a typical consumer-level GNSS receiver, as can be found in cars and mobile phones, can achieve is about several meters (≈±10m) [24]. Moreover, GNSS display the following limitations:

- The Earth's atmosphere (specifically the ionosphere and the troposphere) causes delays in the signal transmission yielding additional error. Also, this error is not the same for all satellites and depends on its position: a satellite near the horizon will involve larger delays as the signal has to travel a longer distance inside the atmosphere than a satellite located near the zenith). Other GNSS receivers, like differential GPS (DGPS), rely on specific approaches to better model and estimate this atmostpheric delay and therefore can increase the resulting precision (≈±2-5m) [25] at the cost of additional constraints on the measurement context. Thus, DGPS requires the receiver to be located in a range of 200 km of a reference station – the precision decreases the further away the receiver is from the reference station.
- GNSS signal is hardly able to penetrate matter. As a consequence, it can be

[9] Trilateration: "A method of surveying analogous to triangulation in which each triangle is determined by the measurement of all three sides." [22]

[10] To determine its position, a satellite navigation receiver will determine the ranges to (at least) four satellites as well as their positions at time of transmitting. Knowing the satellites' orbital parameters, these positions can be calculated for any point in time. The pseudoranges of each satellite are obtained by multiplying the speed of light by the time the signal has taken to travel from the satellite to the receiver. As there are accuracy errors in the time measured, the term pseudoranges is used rather than ranges for such distances. [23]

[11] $c \times 1.0^{-6}\, s = 300\, m$ where c is the constant for the speed of light ($\approx 3.0^8\, m/s$)

blocked or reflected by almost any solid. For example, concrete, glass, wood, foliage and human flesh all block the signal significantly if not completely.

- Some surface will act like a mirror for the frequencies used by the GNSS. If the receiver intercepts a signal that has been reflected on a surface, it will interpret the signal as being received directly from the satellite even if the reflection has added some delays in the TOF. This is known as the multipath problem. The more objects and clutter are around the receiver, the more significant the multipath problem becomes (and thus, the precision is further decreased).

In a mobile AR context, these GNSS limitations can impact the performance of the system. First, the overall precision of standard GNSS receivers is relatively low (i.e. 10 meters is about the width of a street). It can limit the capability of a user to interact with items in the real world. However, it may be sufficient for some applications, especially those in the mild augmentation category. The inability of GNSS signals to efficiently penetrate matter also creates the "urban canyon" problem (Figure 2.6): tall buildings will significantly reduce the line-of-sight of the receiver thus making it more difficult to receive the 4 simultaneous satellite signals that are required for determining the user's position. Moreover, even if 4 satellites are visible, being able to see more satellites will improve the precision. Geo-engineering contexts may not always involve urban canyon situations, but the typical user will be around tall structures (skyscrapper, towers, bridges, etc). It is therefore sensible to assume that in such applications, the line-of-sight of the GNSS receiver will almost always be partially occluded. Finally, a construction site or urban environment is cluttered with different objects likely to produce reflection of the GNSS signal thus increasing the multipath problems (and reducing the precision).

It is worth emphasizing that GNSS receiver *cannot* be used indoor due to the inability of the signal to efficiently penetrate solid matter. Even if an indoor receiver would manage to measure a signal, the multipath problem would be so important that it would be impossible to expect a precision higher than 50-100 meters [26]. The indoor positioning problem and the pedestrian navigation problem (*i.e.* tracking a pedestrian in a urban environment, going in and out of buildings without losing the ability to retrieve its position) depend on alternative solutions to standard GNSS. These will presented next.

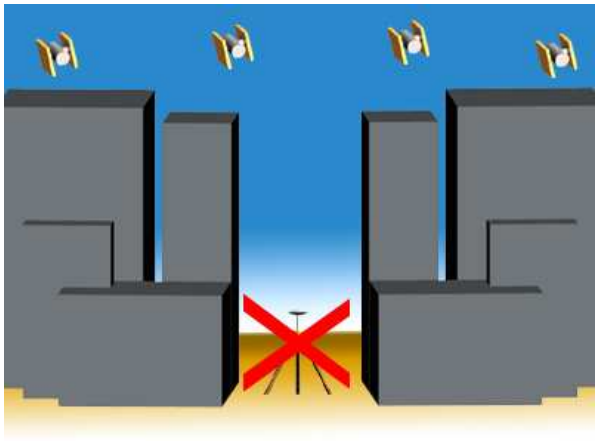

Figure 2.6 – Illustration of the GNSS urban canyon problem. As the GNSS signal is unable to penetrate matter, tall buildings greatly reduce the line of sight of the receiver and it therefore becomes more difficult to get the 4 simultaneous satellite signals that are required for determining the receiver position (*source: http://goo.gl/M66ap*).

2.3.1.2 Indoor positioning and pedestrian navigation

There are some alternative solutions to GNSS receivers for positioning in urban context which could also be considered for indoor positioning and pedestrian navigation: ultra-wideband (UWB), indoor messaging system (IMES), high sensitivity GPS (HSGPS) and GPS-repeaters.

UWB Chiu and O'Keefe [27] propose a relatively formal definition of UWB and a list of benefits:

> UWB pulses are very short, ranging from a few tens of picoseconds to a few nanoseconds and usually last only a few cycles of an RF carrier wave. Since the pulses are short, the energy is spread across a large bandwidth and produces a low power density.
>
> [...]
>
> UWB offers the following benefits:
>
> 1. With power spread over large bandwidth, frequency selective fading from materials/multipath is mitigated;
> 2. There are minimal multipath cancellation effects;
> 3. Low energy density gives minimal interference to nearby systems and minimal RF health hazards;

4. For ranging, there is very fine time resolution for precise distance measurement.

UWB-based tracking can achieve precisions of a few decimeters [28, 29].

In a mobile AR context, UWB positioning have a high potential in terms of achievable precision. The fact that it can still provides decimetric precision in cluttered environments (*e.g.* building sites) is also an important asset. However, it is still based on trilateration and it therefore requires the setup and calibration of an infrastructure, even if temporary, of at least four emitters in order to derive a unique position.

IMES IMES is a technology that is at an early stage of development. The objective is to offer a *seamless* tracking solution. A user could rely on a GNSS receiver outdoor and, when entering a building, use the *same* receiver to keep the tracking indoor. IMES aims at developing an indoor infrastructure (a network of emitters) that can override the GNSS signals when the user enters in a building. The signal sent from IMES would not contain time measurements (as it is the case with the original GNSS signals), but would instead contain the *position* of the *nearest emitters*.

Dempster [30] and Forssell [31] both propose an exhaustive list of pros and cons of this technology. Among them, the ones that requires consideration for a mobile AR system are presented below:

- Pros:
  - Good reliability
  - More precise than Assisted-GPS[12], which is the currently used technology
- Cons:
  - Requires an important infrastructure. To have a decent coverage, emitters needs to be installed at 20-30 meters from each other.
  - Because it is the location of the *emitter* that is broadcasted, high precision would require the installation of numerous emitters.
  - Precision becomes dependent of the calibration that has been done upon the installation of the emitters.

[12] Assisted GPS (A-GPS) is a support infrastructure that enable a receiver to access the satellites' information faster and more reliably in poor reception conditions. For example, in cities, the cellular network can be used for this purpose. A-GPS does not enable an higher precision than standard GPS; it only enables to use GNSS signals in very poor reception conditions.

HSGPS HSGPS aims at reenforcing the GNSS signal strength after it has been attenuated and degraded when passing through solid matter. It does so by integrating the signal over a given length of time. For example, integrating the signal over 20 ms allow for an increase of 10 dB in the signal's strength. It allows the use of GNSS receiver even under heavy occlusion (*e.g.* in a forest or inside a building). As its name implies, this tracking solution does not aim to be more precise than traditional GNSS solutions, but instead to allow a receiver to be more sensitive (and therefore to be able to derive a position measurement) to degraded signals in traditionally difficult location for GNSS signals. Lachapelle [26] has shown that this type of receiver can achieve a precision ranging from 15 to 50 meters in a forest and 45 meters inside a covered stadium.

GNSS repeaters GNSS repeaters consist of installing an additional infrastructure that *repeats* the GNSS signals inside a building so that a receiver can determine its location indoor. Jee [2] proposes the following description of the GNSS repeater system:

> The key to using a GNSS repeater for positioning is a radio frequency (RF) switching device. This device takes a single input (*i.e.*, the live signals from outdoors) and switches it among multiple re-radiation antennas, one at a time. By means of this time domain multiplexing, receivers can sequentially track GNSS signals from the multiple re-radiation antennas installed at different locations without self-interference. With this switching among the re-radiation antennas a change occurs in the signal retransmission path. The change corresponds to the time difference of arrival (TDOA) between the switched retransmission antennas and the user. Therefore, if we have four retransmission antennas connected to the switching repeater, we can obtain three TDOA measurements for three-dimensional positioning.

An example of a GNSS repeater setup is illustrated in Figure 2.7.

This technology presents some limitations. The most important one in a mobile AR context is the overall precision that can be achieved: it will, at most, be similar to the precision of a standard GNSS receiver. Moreover, since the signals are repeated indoor, which is an environment much more prone to multipath problems, the overall precision will often be lower than the precision of the outdoor antenna (1-10 meters).

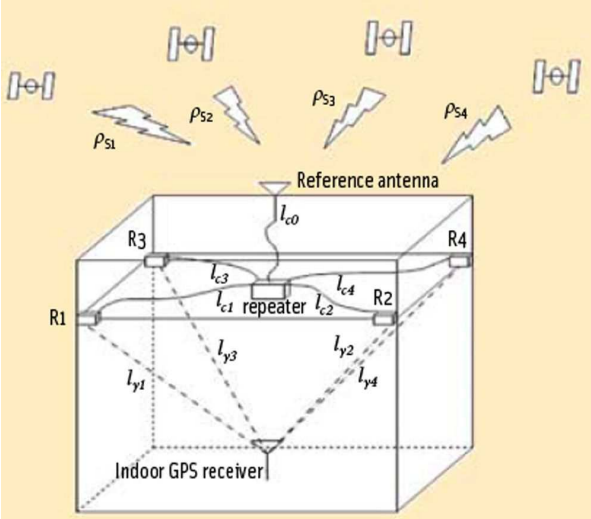

Figure 2.7 – Illustration of GNSS repeater setup. The antenna on the roof of the building receives the GNSS signal and rebroadcast it inside the building. ρ_{sk} represents the geometric range between the k-th satellite and the reference antenna, l_{c0} represents the cable length between the reference antenna and the switching repeater, l_{ci} represents the cable length between the switching repeater and i-th re-radiation antenna and l_{ri} represents the range between i-th re-radiation antenna and the receiver [2].

2.3.1.3 MEMS

MEMS stands for *MicroElectroMechanical Systems* which include, among others, gyroscopes, accelerometers and magnetometers. A gyroscope is a device used for measuring orientation. It relies on the principles of angular momentum conservation. An accelerometer measures the proper acceleration of the device. Multi-axis models are available to detect the magnitude as well as the direction of the proper acceleration (including gravity). An accelerometer can also be used to sense orientation since it can be deduced from multiple readings of the acceleration's direction (because direction of weight changes when an object rotates). A magnetometer is used to measure the strength and the direction of a magnetic field. Vector magnetometers have the capability to measure the component of a magnetic field in a particular direction, thus allowing to measure the orientation of the device.

Some orientation sensing technologies, such as the XSens MTx[13] (Figure 2.8), uses a combination of gyroscopes and magnetometers. The orientation is computed in an "absolute" manner, that is, the reference frame for each measurements is always the same (no relative measurements). A magnetometer is used to measure the direction of the magnetic north in order to ensure stability. If the device includes accelerometers,

[13] http://www.xsens.com/en/general/mtx

Figure 2.8 – The MTx orientation sensor from XSens (*source: http://goo.gl/9OjiQ*).

the position can also be calculated but only in a *relative* manner. This is achieved by estimating the device displacement in 3D space based on the variations of acceleration between two given positions. This problem is named *dead reckoning*.

The accuracy and the refresh rate of these devices are relatively high ($\approx \pm 1°$ at $120\,Hz$ in the case of the XSens MTx[14]), making them very robust and reliable systems, even in difficult contexts like very rapid motions. However, they are very sensitive to magnetic interferences and, as they are relative measurement tools (mainly applies for accelerometers), they are also very likely to drift over time; the longer the tracking system relies only on this hardware and the larger the drift is (as it accumulates over time).

2.3.2 Active solutions

Active tracking solutions aim at recognizing the device's surroundings in relation to a previously known or learned environment in order to determine its current position and orientation. The recognition is done based on a sensor that can capture the environment which is then interpreted. The most common way to capture the environment is by using a video camera. Other means include LiDAR[15], sonar, IR[16], etc. In this study, only video-based recording of the environment is considered given the mobile AR context and related usage of video cameras in the current outdoor solutions (see Section 2.2). When dealing with images or video sequences, the general approach to conduct scene interpretation (required by active tracking solutions) is to rely on *com-*

[14] See http://www.xsens.com/en/general/mtx for complete specifications.
[15] LiDAR: Light Detection And Ranging
[16] IR: Infrared

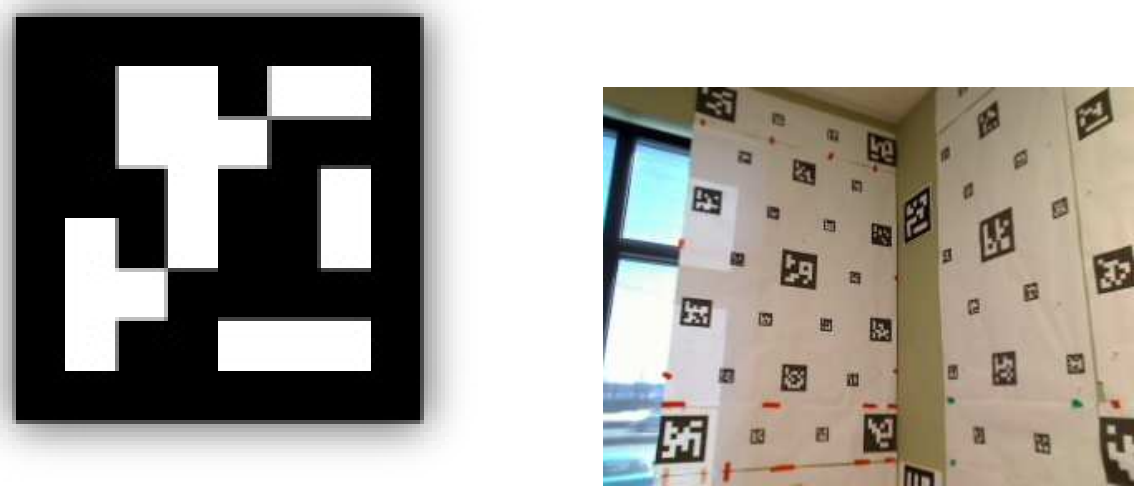

Figure 2.9 – (left)Typical symbol used by the Studierstube library [3] and (right) an example where multiple markers are used to track the position of a device inside a room (*source: (left): http://goo.gl/FE7W6*).

puter vision algorithms. Computer vision (CV) is "a set of computational techniques aimed at estimating or making explicit the geometric and dynamic properties of the 3-D world from digital images" [32].

The following sections will present the various CV methods generally used for tracking purposes.

2.3.2.1 Fiducial markers

Fiducial markers are a technology that enables a full 6 DOF tracking solution relying solely on CV algorithms. It consists of visually distinctive markers (Figure 2.9(left)) which are affixed in the environment. Their size and position in the environment must be known by the system beforehand; if several markers are used to track a same object (or environment) (Figure 2.9(right)) simultaneously, they must be calibrated and their relative position must also be known by the system.

The tracking consists in retrieving the pose of the camera in relation to the marker(s)' coordinate system (Figure 2.10). The method is explained in details by Kato and Billinghurst [4], but here is an overview of the algorithm:

1. Find the shape of the marker in the image;

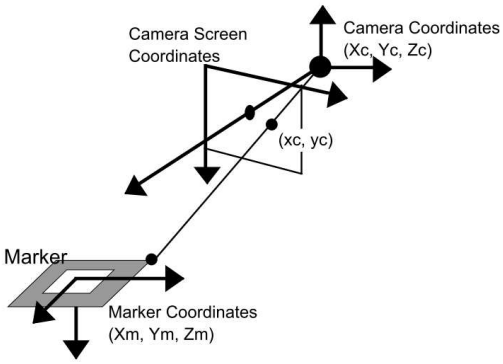

Figure 2.10 – Finding the pose of the camera consists of finding the transformation matrix between the camera's coordinate system and the marker's coordinate system. (X_C,Y_C,Z_C) are the camera's coordinates, (x_C,y_C) are the camera screen's corrdinates and (X_m,Y_m,Z_m) are the marker's coordinates. (*source:* [4])

2. Identify the marker using template matching;
3. Recover pose of the marker by analyzing the projection of the marker's edge in the image.

This technology is very easy to use (thanks to the development of libraries like ARToolKit[17] and Studierstube[18]). However, it has some constraints:

- The markers need to be installed in the environment;
- The markers need to be calibrated (the system must know exactly the location of each marker in the environment);
- In most marker libraries (*e.g.* ARToolKit), a marker must be entirely seen into the camera's field-of-view for the tracking to be effective (marker-based systems show very poor robustness to occlusion). More recently, marker libraries based on NFT (Section 2.3.2.2) show more robustness to partial occlusions;
- The marker detection step is sensitive to changes in environment (light conditions and shadows);
- Most implementations show relatively important jitter[19] and therefore, lack of

[17] http://www.hitl.washington.edu/artoolkit/
[18] http://studierstube.icg.tugraz.at/main.php
[19] Jitter: slight irregular movement, variation, or unsteadiness, especially in an electrical signal or electronic device [33].

accuracy for small number of markers[20].

In the context of mobile AR in unprepared environments, fiducial markers technology presents a major drawback: the marker needs to be affixed in the environment prior to using the system. The problem is worsen when a 360° coverage is required: an important number of markers need to be installed. In addition, it is required to carefully calibrate each marker. Finally, if the user wants to walk farther away or closer to the structures, there need to be markers of different size set up in the environment to ensure that at least one marker will be visible in the camera's FOV at all time, and that its size enables the camera to view it large enough to calculate a position out of it.

2.3.2.2 Natural Features Tracking

Natural Features Tracking (NFT) is a specific field of research in computer vision. It consists of finding, in an image, points that have high discriminative value (corners, strong gradients, etc.). These points and their surroundings are then extracted and analyzed to compute descriptors that are invariant to a range of factors (illumination changes, rotation, scaling, minor changes in point-of-view, etc.). Many solutions have been developed using this principle such as the Features from Accelerated Segment Test corner detector (FAST [34]), *Scale-Invariant Feature Transform* (SIFT [35]) and *Speeded-Up Robust Features* (SURF [36]). These solutions have proven to be really robust, even in varying outdoor conditions. Moreover, even if this type of technology can be computationally expensive, many implementations of these solutions have been optimized to be able to run in real-time (under certain conditions of resolution and minimal hardware performance). It is interesting to note that when using a tracking approach based on natural features, most of the time the precision is expressed in *pixels* instead of *meters*[21].

NFT can be used in different ways to determine the user's position and orientation. The next subsections presents the three main methods usually exploited in the AR field, namely: optical flow; SLAM; panorama.

Optical Flow Optical flow is "the distribution of apparent velocities of movement of brightness patterns in an image" [37]. In a tracking context, this technique can be used

[20]Increasing the number of markers yields an increased tracking stability; however, the installation and calibration processes get tedious and problematic rather quickly as the number of marker increases.

[21]More in-depth information about the implication of the accuracy representation is provided in Section 2.1.

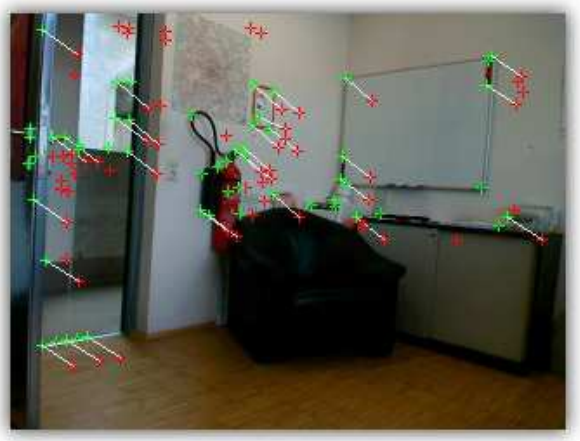

Figure 2.11 – Example of optical flow. The red crosses represent the location of the image features in the last video frame and the green crosses are the image features identified in the current frame. A link between two crosses representing the movement vector is drawn between two corresponding features. Optical flow consists of analyzing the movement vector field created between two frames (*source: http://goo.gl/CsMC2*).

to deduce the displacement (i.e. position and orientation) of the camera between two (successive) images by tracking corresponding natural features on each image. When using a video camera, this procedure is applied between two consecutive frames. This tracking solution provides *relative* measurements. It cannot compute an absolute position and orientation because it only accumulates the various displacements since the initialization of the process. Figure 2.11 shows an example of such tracking in real time.

Optical flow is a solution that is easy to implement if only the 3 DOF related to orientation need to be tracked. Displacements of pixels on screen can easily be converted to angular values given a basic knowledge of the camera's field-of-view. Optical flow can also be used as a complete 6 DOF tracking solution, but it then requires complex models to evaluate the change in depth (when the user in moving along the "Z" axis in his local reference frame – or going "forward"). In order to achieve this estimation with precision, a fine evaluation of the scale change of the objects is required. This can be a problem when the camera gets too far or too close to these objects. Additional limitations inherent to the optical flow approach are the following:

- Since optical flow methods provide only measurements in relation to the user's initial position and orientation, absolute measurements become impossible as soon as the tracking continuity is lost (*i.e.* one or more pairs of frames fail to return a displacement measurement).
- Optical flow methods display very poor resilience to occlusions or moving ob-

jects [38]; according to the moving object's size, the computer vision approach may interpret the image contents as if the user was moving instead of just the object.

- Optical flow methods display very poor robustness to rapid motions if the motion blur is not estimated and corrected using usually complex methods (or used as a cue to estimate the motion velocity – again using complex methods) [39]; if the user moves too fast, blur will be observed in the images yielding errors in the natural features' position. As the tracking is relative to the initial position, these types of errors then accumulate and create drift over time.

SLAM-based Approaches SLAM stands for *Simultaneous Localization and Mapping* and has first been introduced by Leonard and Durrant-Whyte in 1991 [40]. It consists of creating a map of an unknown environment using different types of features while at the same time keeping track of the current position and orientation (*i.e.* 6 DOF) of a robot (or user) in the said map. In fact, SLAM could be considered more like a general framework as it consists of many parts which all can be implemented using different sub-algorithms (feature extraction, association, map building, localization, etc.). It has been developed in the field of robotics but has recently showed great results for visual tracking applied to augmented reality by offering a solution that is robust to occlusions, view-point and illumination changes for small environments using recent advances in NFT [41, 42, 43]. Such SLAM applications are also called monoSLAM since they use a single camera and therefore rely only on monocular computer vision algorithms (no stereo vision, sonar, laser or inertial captors as it is often found in the robotic applications of SLAM).

This type of system presents great performances and stable tracking. They also allow to derive the camera pose based on direct measurements of the environment, either through a camera or other sensors (*e.g.* sonar, laser, etc.). However, it also displays some drawbacks:

- SLAM requires to have devices able to directly measure (or derive indirectly using calculations) the depth of the points they measure (stereo camera, laser, sonar, vision, etc) [44].
- Large environments are more difficult to handle; SLAM systems are, for now, only usable in small environments because of the number of features to maintain in the map. Moreover, if the map is too big, the localization time in the said map becomes too computationally expensive and prevent real-time usage [45].

- Estimation of the depth of the extracted features is not easy for a monoSLAM system. It can be achieved using overlapping images of a scene taken from different points-of-view. However, this requires complex algorithms and important computational power in order to sustain real-time performances [41, 42, 43].

In a mobile AR context, stereoscopic SLAM systems might not be envisioned as a possible solution due to the hardware requirements. Indeed, the two cameras of a stereoscopic SLAM need to be installed at a minimal distance from each other on the mobile device in order to correctly triangulate the depth of the features. This does not come standard on mobile devices (they come with at least a camera, often two, but not located on the same side of the device – one in the back for capturing pictures and videos and the one in the front mainly for video conferencing). MonoSLAM solutions are more adapted to mobile AR, at least in terms of hardware requirements.

Panorama-based orientation tracking

Definition A panorama is:

> A picture of a landscape or other scene, either arranged on the inside of a cylindrical surface, to be viewed from a central position (also called cyclorama), or unrolled or unfolded and made to pass before the spectator, so as to show the various parts in succession. Hence, more generally: any pictorial representation of a panoramic view. [46]

An example of a panorama captured from a construction site is presented in Figure 2.12. A panorama can be obtained in a single shot, using fish-eye lenses or using hyperbolic mirrors. A panorama can also be obtained from multiple snapshots of a scene covering 360°. These snapshots need to be assembled (i.e. stitching process) to build the panorama. This is achieved manually or automatically using software like Hugin[22] or Microsoft Image Composite Editor (MS ICE)[23].

Orientation tracking Orientation tracking based on panoramas can be considered as a very specific case of SLAM-based tracking (Section 2.3.2.2). Indeed, panorama-based orientation tracking tries to emulate the map of the SLAM system by using a

[22] hugin.sourceforge.net
[23] http://research.microsoft.com/en-us/um/redmond/groups/ivm/ICE/

Figure 2.12 – Example of a panorama captured at a construction site

brute-force approach: the whole panorama becomes a map of natural features. When a camera is at a fixed, static position, and its orientation is freely modified, the camera "perceives" the world exactly as a panorama would.

Typically, the first step consists in building the panorama. The user, equipped with a camera (the camera is either affixed on a hand-held device or connected to the main panorama-based tracking system), changes his orientation. The system then dynamically and incrementally builds the panorama while using NFT technology to match the live camera image with the panorama (Figure 2.13). Once the panorama is built, the user can change his orientation freely as the live view is continually compared to the reference system (*i.e.* the panorama) in order to calculate the orientation of the camera. The use of a panorama as a reference system allows the system to be particularly robust to drift, which was one of the main problems with optical flow, while maintaining its simplicity. Recent advances in portable computing hardware and computer vision have enabled real-time reconstruction of panoramas using a live video feed [47], getting conceptually closer to SLAM approaches where the map creation and the localization take place concurrently [48].

One potential problem when using a panorama as a reference system is the scene's dynamism. As specified in Chapter 1, it is not possible to expect a static scene for the application cases this work is interested in. However, even if the scene is dynamic, the panorama remains a good reference system in most cases. Because the panorama assembly step typically involves a large number of pictures (or a video feed) of the real world scene, the dynamic elements can be cropped out relatively easily [49]. However,

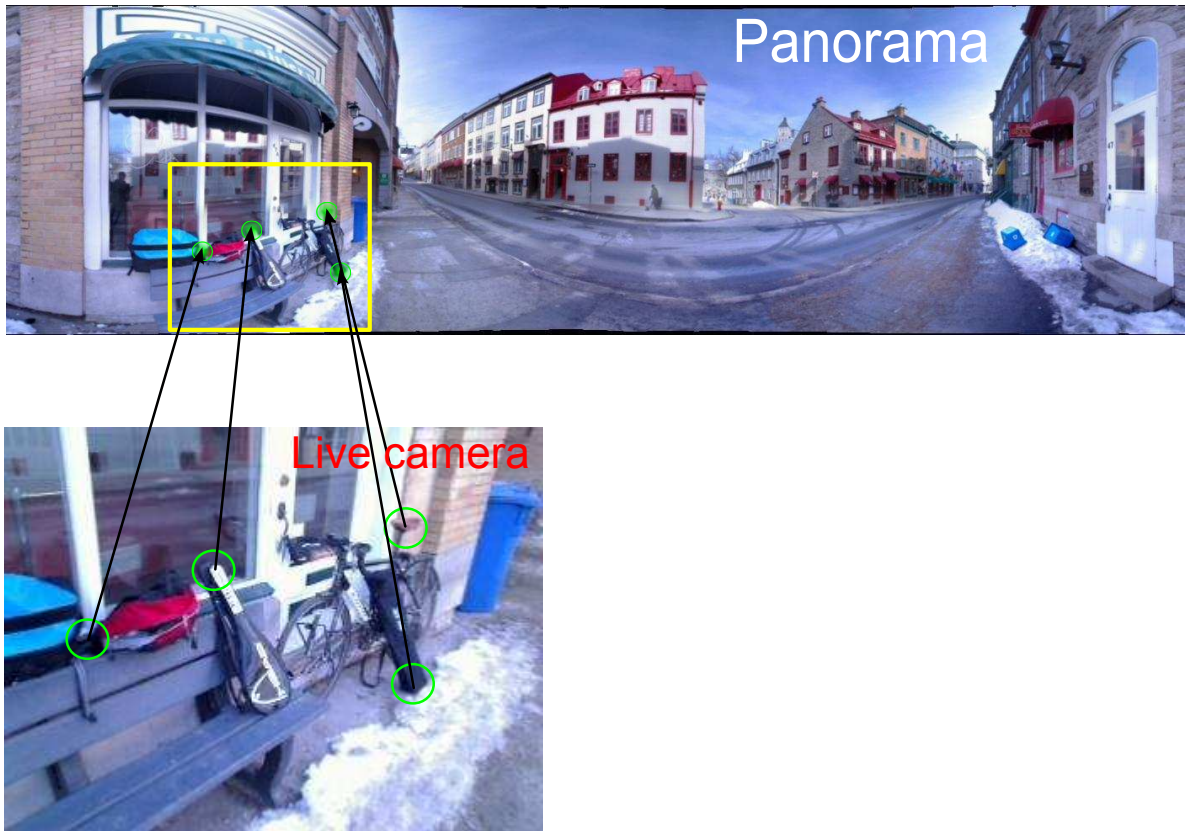

Figure 2.13 – Example of panorama and live video correspondences

there are some cases that might prevent correct use of a panorama as a reference system (*e.g.* if the dynamic objects move too slowly and are therefore captured at the same location on multiple images, they will appear in the final panorama – cars stopping at a red light).

The idea behind panorama-based orientation tracking is a compromise between optical flow and SLAM solutions. Optical flow is simple to implement but lacks the reference system of SLAM approaches. On the other hand, SLAM solutions, while having an "absolute" reference system and being very robust to occlusions and rapid motions, presents limitations for "large" environment and are too complex to integrate and adapt given the scope of the project.

Projection types Panorama images, as it is the case with any geographical maps, must be projected onto a surface. The choice of the projection surface must be done carefully as not all projections are appropriate for all uses. This section will detail the main projections that can be considered and their implication in a context of panorama-based tracking.

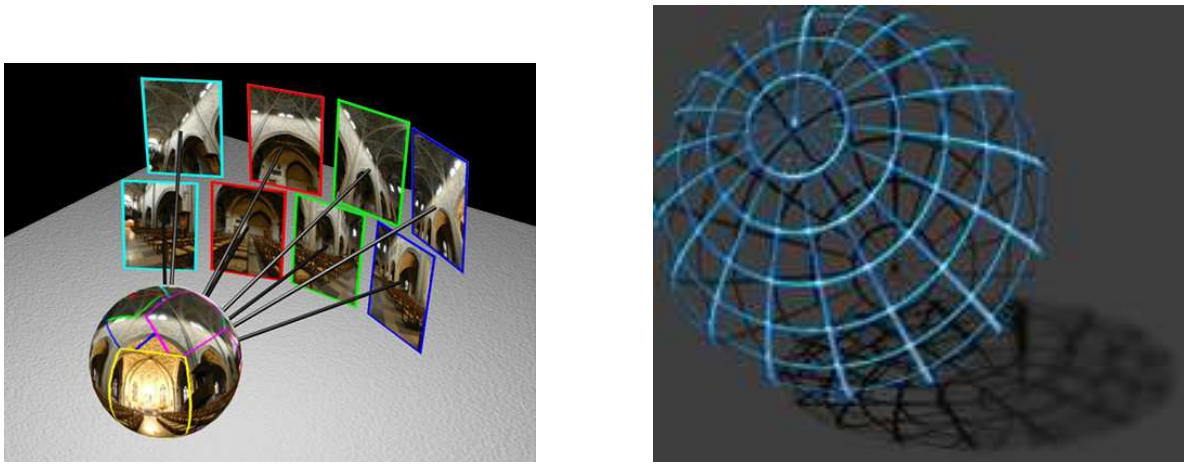

Figure 2.14 – (left) The image stitching process consists in finding the spherical coordinate of each image on a base sphere and (right) a grid representing the field of view of an observer standing at its center (*source: (left): http://goo.gl/vA6sd (right): http://goo.gl/vWMgT*).

A "perfect" or "complete" panorama can be seen as a $(360° \times 180°)$ field of view. It corresponds to a sphere (Figure 2.14 (right)) where the observer is standing at the exact center. However, a panorama is also an image which is, by definition, a projection of the reality onto a *plane*. The different projections therefore correspond to the different ways that a sphere can be mapped to a planar representation. For each projection, a projection of the meshed sphere represented in Figure 2.14 (right) will be presented in order to highlight the different implications (deformations, FOV, etc) of the projection along with an example of the projection applied to a real panorama.

In the context of this project, the panorama images are created through the registration (*i.e.* matching) and merging of a set of individual images (as captured by standard mobile devices' camera). This matching process tries to retrieve the spherical coordinate (*roll*,*pitch*,*yaw*) of each captured image on a sphere (Figure 2.14 (left)). This result in a spherical, intermediate representation that needs to be projected on a surface.

The first projection surface to consider is the plane. This projection is called *rectilinear projection*. In theory, an infinite plane tangent to the sphere can be used to project a FOV of only 180° . However, in practice, an horizontal viewing angle of 120° or more will be very difficult to project because, for a given viewing angle, the projection surface on the plane expands the further away the projection angle is from the plane's normal (Figure 2.15). Moreover, it will induce strong distortions as the horizontal FOV increases as it can be observed in Figure 2.16.

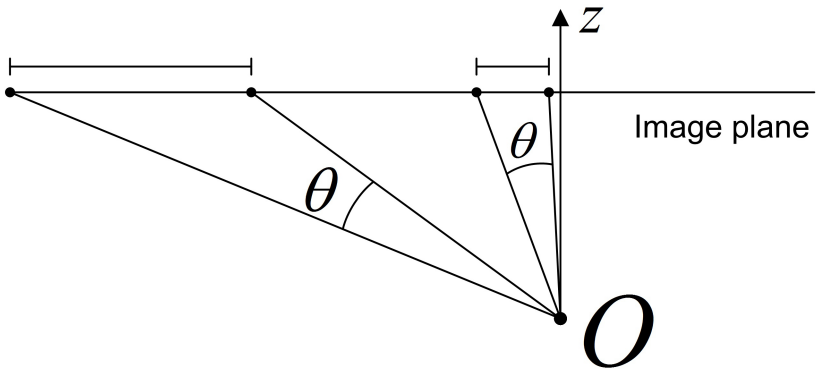

Figure 2.15 – Illustration of the projection surface expansion of a given viewing angle θ the further away the projection angle is from the plane's normal (represented by the z-axis in the figure). O is the camera's center of projection.

Figure 2.16 – (left) Illustration of the deformations generated by a rectilinear projection. (right) Example of a panorama projected using the rectilinear projection. (*source: (left): http://goo.gl/sI5oO (right): http://goo.gl/jWRBp*)

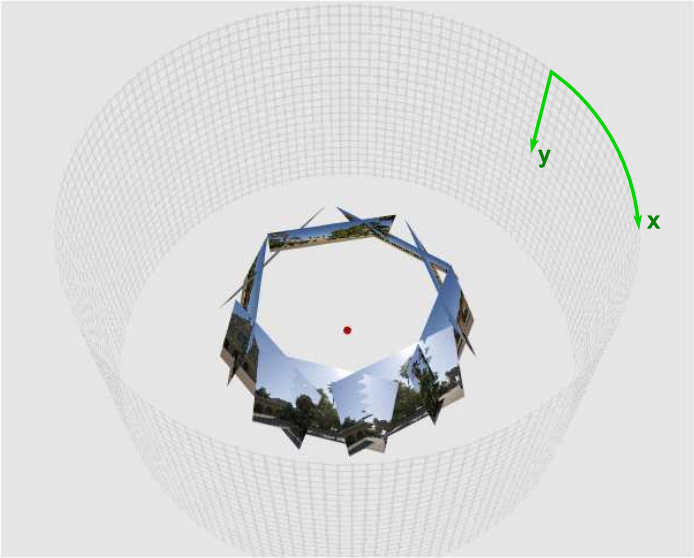

Figure 2.17 – Individual images are aligned properly and displayed in the 3D space. The cylindrical projection plane is located behind the images. The green arrows represent the coordinate system of the projection plane (the x axis is illustrated as a curve to highlight the fact that this projection surface will be unrolled to yield a planar surface). (*source: Modified from http://goo.gl/9U3ep*)

A way around the limitations of the rectilinear projection is to project the aligned pictures on the surface of a cylinder, through *cylindrical projection*. The cylinder can then be unrolled to yield a flat plane. The aligned pictures are displayed in a 3D space, each located at a distance equivalent to their focal length f from the camera's center of perspective (3D point where the images were taken from – represented by a red ball in Figure 2.17). The value of the pixel on the unrolled projection surface (*i.e.* the cylinder) at position (x,y) can be retrieved by calculating the corresponding angular position (ϕ,λ) (in radians) (Figure 2.18) in the aligned image space (Equations 2.5 and 2.6 are based on a cylinder radius of 1):

$$\phi = \tan^{-1}(y) \tag{2.5}$$

$$\lambda = x + \lambda_0 \tag{2.6}$$

where λ_0 is the longitude for $x = 0$ (this value is set by the creator of the map) [5].

Cylindrical projection is often used for landscape and architecture, as well as for

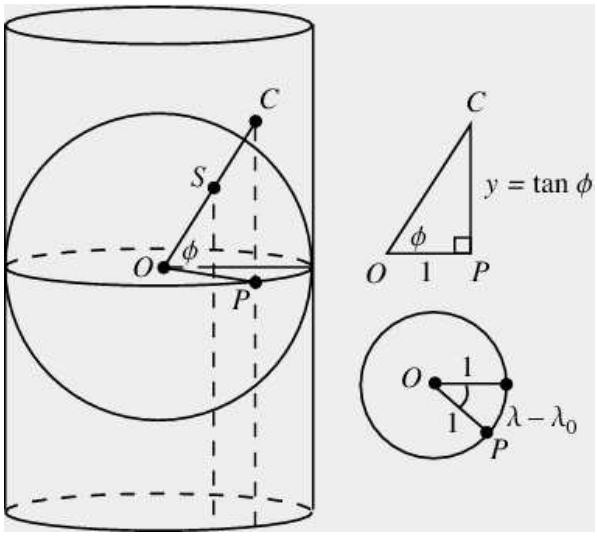

Figure 2.18 – A cylindrical projection of points on a unit sphere centered at O consists of extending the line OS for each point S until it intersects a cylinder tangent to the sphere at its equator at a corresponding point C. If the sphere is tangent to the cylinder at longitude λ_0, then a point on the sphere with latitude ϕ and longitude λ is mapped to a point on the cylinder with height $\tan\phi$. [5]

geographical maps[24]. Projecting pixels on a cylinder will cause deformations (straight lines will be bent). Also, proportions are not maintained when moving along the central meridian of the cylinder (*i.e.* fixed λ and variable ϕ) (Figure 2.19). Finally, the cylindrical projection involves blind spots at the poles as illustrated in Figure 2.20. From the definition above, a panorama is described as an image projected on a *cylindrical* surface. While this is how panoramas are presented most of the time, this is not the only projection that can be used. In the context of panorama as a constituent of a tracking solution, the main other projection surface is the sphere.

The projection where images are projected onto the surface of a sphere is named *equirectangular projection*. It maps meridians to equally spaced vertical straight lines, and circles of latitude to evenly spread horizontal straight lines. Similarly to the cylindrical projection, the equirectangular projection introduces deformations (Figure 2.22). The value of the pixel on the projection surface at position (x,y) on the unrolled sphere can be retrieved by calculating the corresponding angular position (ϕ,λ) in the aligned image space (Figure 2.21) (the Equations 2.7 and 2.8 are based on a sphere ray of 1):

$$\phi = y \tag{2.7}$$

[24]Maps ofen use the Mercator version which is mathematically more complex but allow to keep some important properties for maps.

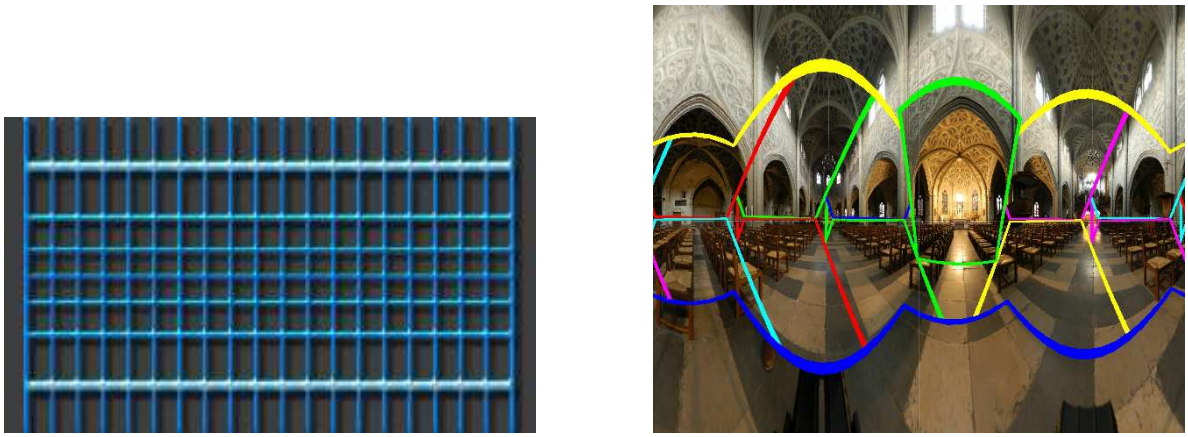

Figure 2.19 – (left) Illustration of the deformations generated by a cylindrical projection. (right) Example of a panorama projected using the cylindrical projection. (*source: (left): http://goo.gl/gDyzq (right): http://goo.gl/cOHcW*)

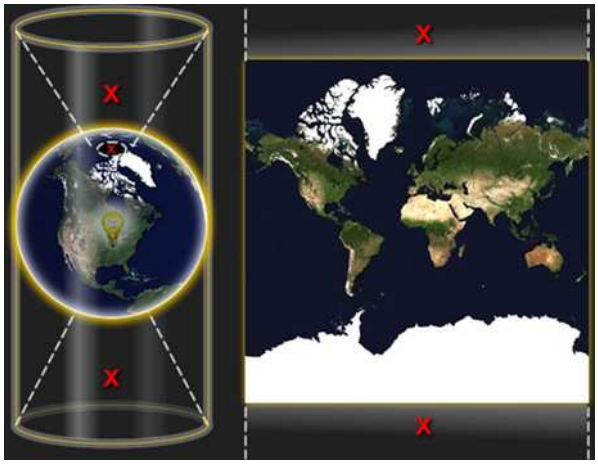

Figure 2.20 – Example of cylindrical projection. This illustration help to grasp the limitation of this type of representation: deformations at the poles and blind spots (indicated by a red "x") (*source: http://goo.gl/1V2v3*).

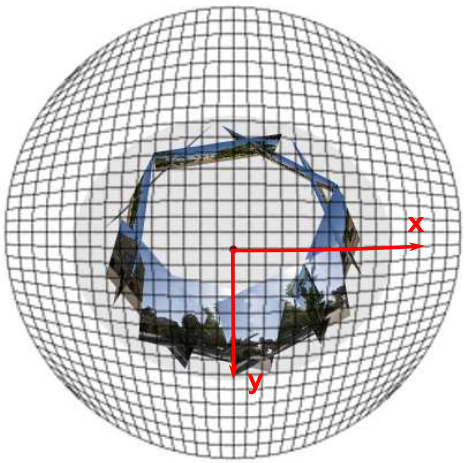

Figure 2.21 – Individual images are aligned properly and displayed in the 3D space. The spherical projection plane is located behind the images. The red arrows represent the coordinate system of the projection plane (the (x,y) notation is used to highlight that the projection surface will be unrolled to yield a planar surface). (*source: Modified from http://goo.gl/9U3ep*)

$$\lambda = \lambda_0 + x \sec \phi_1 \tag{2.8}$$

where λ_0 is the longitude for $x = 0$ and where $\phi_1 = 0°$ for the equirectangular projection (this projection is a specific case of many cylindrical equidistant projection for which the value of ϕ_1 varies) [50].

The main advantage of the equirectangular projection over the cylindrical projection is that there is no blind spots at the poles, thus allowing a complete 360° view.

2.4 Tracking Methods Compliant with Mobile AR GIS for Geo-engineering

The previous sections described the constraints inherent to AR in a mobile, engineering context, the different tracking solutions that can be used to assess the camera positioning and orientation and also the accuracy requirement for a system to be sustainable in a architecture and geo-engineering context. Relying on this knowledge, the main tracking approaches relevant to the present research work can now be selected.

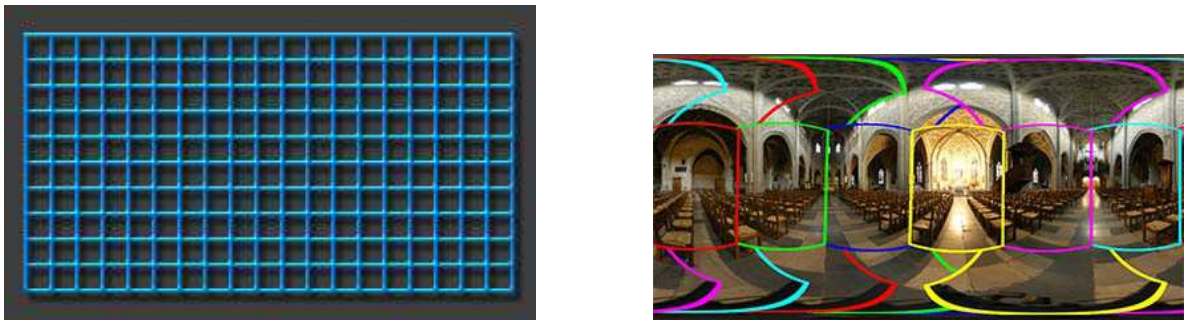

Figure 2.22 – (left) Illustration of the deformations generated by an equirectangular projection. (right) Example of a panorama projected using the equirectangular projection. (source: (left): http://goo.gl/Uy1Da (right): http://goo.gl/oB0U8)

Tracking could be carried out using a passive or an active approach. The passive approach comprised of GNSS, indoor positioning and MEMS. The active approach comprised of computer vision-based methods relying on fiducial markers and NFT like optical flow, SLAM-based approaches and panorama-based tracking. GNSS is not very well suited for high accuracy tasks. Indeed, the signal used in this technology degrades quickly when interacting with solid matter or when in presence of reflective materials (multipath problem). These problems become very limiting especially when the system evolves indoor or in very cluttered environments such as urban environments (*e.g.* urban canyon) or construction sites. Alternatives to GNSS systems exist (indoor positioning), but most technologies belonging to this category require the installation of an additional infrastructure (UWB, IMES or GNSS repeaters all require the setup of a network of emitters). This is not acceptable in the context of this research since installing additional infrastructure requires lengthy set-up and often precise calibration. For a relatively mobile user evolving in a highly dynamic environment, this constraint would hinder any extensive usage of the mobile AR GIS solution. Moreover, these systems only provide the user's position and any positioning system should be complemented with a device to capture the orientation in order to achieve 6 DOF tracking. MEMS are often used to measure device or users orientation. However, they mainly provide relative measurements and thus, they are often prone to drift. Compasses can also be used for this purpose, but the ones that comes with standard mobile devices such as tablets are very sensible to magnetic interferences and are often limited both in terms of the number of tracked DOF and of the precision they can achieve (even when using computer vision techniques to improve the accuracy of the compass, Park and Park [51] could only achieve a precision of 2.2°).

Tracking solutions based on computer vision can also be used in mobile AR (active approaches). Markers-based approaches are very easy to use in short-range applications but are not adapted to the geo-engineering context of this project since they require

to be installed on site and calibrated before usage (as it is the case with some of the previously mentioned passive solutions). Optical flow approach suffers from similar limitation as MEMS devices, namely it is a relative orientation approach which is prone to drift.

HSGPS, SLAM-based approaches and panorama-based tracking stem as tracking solutions suitable for a geo-engineering context. Nonetheless, HSGPS, even though it allows the reception of a GNSS signal in suboptimal conditions, does not offer very high positioning precision. In fact, this technology aims at improving the reception of the GNSS signal but does not offer any improvement in terms of precision compared to traditional GNSS receiver (which precision range is 5-10 m). This is too low for the requirements of most of the tasks occurring on a construction site as detailed in Section 2.1. SLAM-based systems are a very promising tracking solution for mobile AR, but they are very complex to develop, mainly because each step of the algorithm can be implemented in many different ways and using many different sensors. Moreover, they face problems when tackling tracking in large environments, due to the quickly expanding map size. Panorama-based tracking, on the other hand, presents the same central advantages of the SLAM-based approaches but is a simpler system. This however comes at the cost of a more constrained mobility, namely 3 DOF instead of 6 for SLAM-based approaches.

Given the strengths and weaknesses of the tracking approaches, panorama-based tracking was selected because it is the approach that presents the less critical limitations. However, this force the adoption of a 3 DOF (orientation only) solution. Nevertheless, given the time span of the M.Sc. project, it is a reasonable alternative because it allows to reduce the overall complexity of the problem at hand. This scope is still aligned with the objectives of this research. Solving the 3 DOF problem represents a first step in the direction of high accuracy mobile AR GIS. Measurements on construction site are often carried out using instruments mounted on tripods (ex. total station). A user requiring accuracy and precision will rarely use a measuring device while changing its position very often. Moreover, working with panorama images present strong advantages, namely:

- They are built using cheap and readily available devices: a standard camera and a computer is all that is required;
- It enables the use of already existing data (*i.e.* externally produced panorama images – Google StreetView or Earthmine for example);
- It enables the use of computer vision algorithms, which has the potential of achieving pixel or sub-pixel precision;

- For the visualization process:
  - it enables the user to "pause" the reality by displaying the corresponding portion of the panorama image instead of the live video feed if the scene's dynamism hinders the augmentation experience or the accuracy of the selection;
  - it allows a user to remove occlusions (*e.g.* passersby, cars, etc.) through post-processing or by merging newly captured images with the current panorama;
  - it gives context by displaying a larger field-of-view compared to a standard camera; this context is most useful if the visualization is done offline or remotely because the user cannot see the surrounding.

Therefore, given the advantages of the panorama-based approach, its simplicity in comparison to SLAM approach, its robustness to drift in comparison to optical flow or orientation trackers, its capability to potentially achieve pixel precision, it has been selected as the tracking method to investigate in the proposed research project.

Chapter 3 will detail the panorama-based tracking introduced in Section 2.3.2.2. It consists of using the panorama as a map of landmarks that will be used to calculate the orientation of each frame of the video camera during the augmentation phase. Chapter 4 will present a revised version of this approach in an attempt to increase the accuracy and precision of the *overall* system through the improvement of the initialization process (the initialization task consists in finding the pose between the virtual elements and the panorama).

Chapter 3

Orientation Tracker

Orientation Tracker (OT) is the first approach that was developed and investigated to achieve high-accuracy AR in the context of the proposed project. It consists of a panorama-based tracking system. A panorama is first created and the tracking phase is then initiated. The tracking consists in retrieving the orientation of each live image from the video feed (relative to the panorama) by matching it with the corresponding area in the panorama. Therefore, as its name implies, it is a system aimed at calculating the *orientation* of a given frame; thus, it is limited to 3 DOF. The user's position is assumed to be stable (*i.e.* no translation movements). If the device is actually hand-held, the assumption that objects are far enough from the user so that a slight change in position will not disturb the orientation values in a very noticeable manner [52] can usually be made. However, if the user is doing a close-up task (*e.g.* layout), this cannot be assumed and the user would be better off using a tripod or any stable installation.

This chapter presents the result of the experiments aimed at determining whether this tracking approach is able to fulfill the accuracy requirements of geo-engineering and architecture tasks. First, a review of the related work is provided, giving an overview of where OT stands in relation to other panorama-based systems. Then, a more in-depth view of the algorithm used by the Orientation Tracker system is proposed. Finally, the results of the accuracy evaluation are presented and a review of the different elements that influence the accuracy of this system is performed.

3.1 Related Work

Lately, panorama creation and tracking begins to spark interest in the AR community and several systems using panorama images as the main mean of tracking have been developed. The following paragraphs provide a review of the main work carried out on that topic.

Envisor, a system developed by Diverdi *et al.* [52], is a solution that is conceptually close to Orientation Tracker. It aims at creating a panorama, in real-time and incrementally, for which a tracking system is required. The tracking is based on a hybrid approach: a combination of frame-to-frame tracking (*i.e.* optical flow – see Section 2.3.2.2) and image descriptors (*e.g.* SURF or pixel patches) to add robustness. The image descriptor extraction process is dynamic, creating and deleting features on the fly, similar to SLAM-type systems. In a further improvement of Envisor, adding relocalization upon tracking failure, Kim *et al.* [53] also provide an in-depth evaluation of the accuracy of the overall system. In very controlled contexts, the system achieves an accuracy ranging from 0.5° to 3°. In a free motion setup, the accuracy mainly ranges from 0.5° to 10°, even when relying on the relocalization functionality.

The main difference between OT and Envisor concerns the targeted objective of the systems: OT's main objective is to determine, with high accuracy, the orientation of the camera based on panorama tracking while Envisor aims at creating a panorama for which a tracking system is required. Also, the orientation assessment is not the sole purpose of OT, since, ultimately, it aims at using the panorama for the augmentation of the physical environment instead of augmenting the live view (probably at the cost of reduced accuracy as the scene is likely to have changed since the panorama has been built – this remains to be tested). Finally, for simplicity purposes, OT does not handle the computation and deletion of the image features on the fly as does Envisor. Therefore, OT needs to project part of the panorama on smaller planes to generate rectilinear projections that are comparable to the live feed images (see Section 3.2.1.2).

Another state-of-the art system proposed by Wagner *et al.* [54] aims to port a tracking system similar to Envisor to mobile platforms. The panorama is projected onto a cylindrical surface, thus creating "dead zones" at the top and bottom of the panorama (Figure 2.19). It also introduces distortions that are proportional to the distance from the horizon line. While it can operate at real-time speed, its accuracy and precision remains in the range of the results presented by Envisor (*i.e.* about 4° in horizontal direction).

These ranges of accuracy are not fulfilling the required accuracy calculated in Section 2.1. To fulfill the accuracy constraint, the angular error tolerance should be in the interval [0.01°,0.20°]. The proposed approach, OT, aims to satisfy this constraint. The next section will describe the main components of this system.

3.2 Orientation Tracker System

Orientation Tracker is composed of two main components: *Feature Map Construction* and *Tracking*. These two components do not work in parallel, unlike SLAM-based solutions. The user must first build the feature map before initiating the tracking phase. Figure 3.1 proposes an overview of OT workflow including the various steps executed when operating the system. The following sub-sections provide in-depth information about, respectively, the map construction and the tracking component.

3.2.1 Map Construction

Map Construction is the preliminary task to be completed before any orientation can be computed by the system (tracking). Because OT is based on panorama images, the first step consists in assembling the panorama. This panorama image is then processed further to build a map of features used to conduct the tracking.

3.2.1.1 Panorama construction

Principles The panorama construction process is very important. As the panorama image will be the reference system of OT, the accuracy of the computed orientation is positively correlated to the overall quality[1] of the panorama. That is, if the panorama construction process introduces a spatial uncertainty (σ_x,σ_y) on the pixels, any spatial information deduced from the panorama will have, assuming a perfect matching algorithm, an uncertainty of (σ_x,σ_y).

[1]The term *quality* can be interpreted in different ways in this context (*e.g.* visually pleasing or coherent (no pedestrian "cut in half" at the junction of two different images)), but it is here understood as the property that states that each pixel is spatially accurate. For example, if the reality is sampled at orientation (θ_{World},ϕ_{World}), the corresponding information should be located at orientation (θ_{Pano},ϕ_{Pano}) in the panorama.

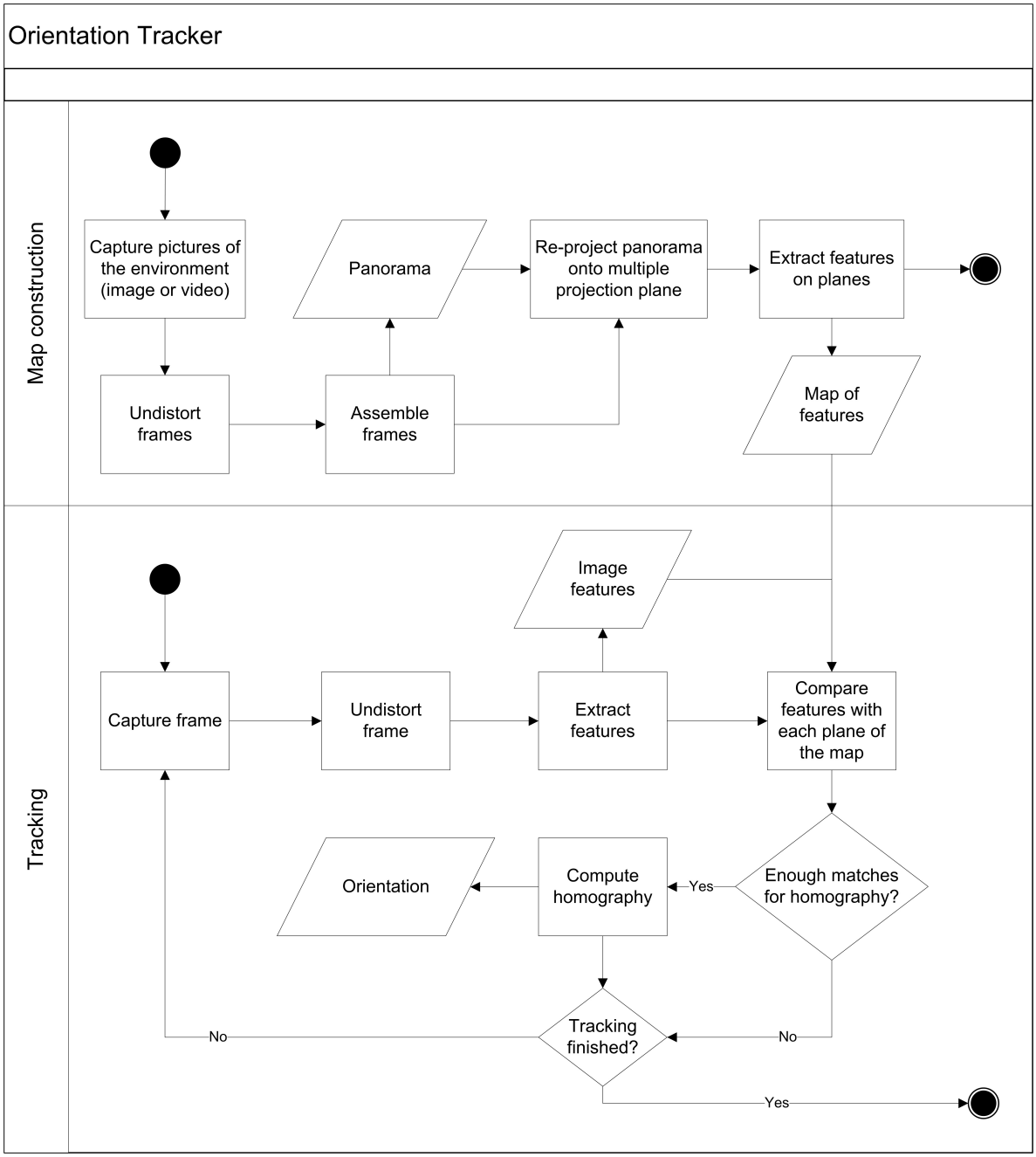

Figure 3.1 – Workflow diagram of the proposed Orientation Tracker system

The first step of the panorama construction process consists in capturing the pictures of the environment. The camera is affixed to a tripod and pictures covering at least a 360° field of view in the horizontal dimension are recorded (it is important to "close the loop" when stitching all the images together). The pictures should have a minimal overlap region of approximately 20% to 50% [55]. Increasing the overlap between the pictures results in an improved stitching at the cost of additional processing time (since there are more images to process).

Implementation In order to stitch all the images together, an external program called *Microsoft Image Composite Editor*© (MS ICE) was used. Experiments were also realized with *AutoStitch*© [56, 57], but MS ICE proved to be more adapted to the context's accuracy needs (MS ICE uses a fixed focal length while AutoStitch makes small adjustments in order to make some pictures fit in the panorama – adjusting the focal length introduces local deformations). MS ICE also allows the use of different projection types and adjustments of the said projection of the panorama before rendering it to its final form, providing additional flexibility. Figure 2.12 was rendered using MS ICE. To have a complete representation of the environment without deformations at the poles, the equirectangular (spheric) projection was used for rendering the panorama.

3.2.1.2 Feature extraction

Principles Once the visual representation of the map is built, features characterizing the image content have to be computed. Such features allow associations to be carried out between the live video feed images and the panorama, and thus matching each image with its corresponding area in the panorama.

Feature extraction is a process that is *not* invariant to image distortions. For this reason, if the flattened panorama is simply fed to the feature extraction algorithm, the extracted features from the panorama will not be "compatible" with the features extracted from the live camera feed (Figure 3.2 highlights the difference in term of distortion between the flattened portion of the panorama in equirectangular projection and the live input from the camera). In other words, two features extracted from the same corresponding location on both the flattened panorama and the live image will bear very little similarity and mismatches are very likely to occur. In order for the features to be comparable, they must be extracted from a *planar* projection, as it is the projection natively used by a camera. To meet this constraint, the projection sphere of the panorama was discretized in a uniformly distributed number of planes (which

resolution is similar to the input camera's resolution) (Figure 3.3)[2].

Once the sphere is discretized, the panorama is reprojected on these smaller planes (Figure 3.3). Feature extraction is applied on these planes and image features are stored. The "feature map" consists of these image features. This feature map is not "global"; each plane references only its own image features. Each plane's position is stored as a (θ,ϕ) position on the sphere.

Implementation The image descriptor that was used to represent image features is, like the original Envisor system [52], *Speeded-Up Robust Features* (SURF) [36] technology. Specifically, the implementation proposed by the OpenCV library[3] was chosen since all the other components of OT were already using OpenCV.

The process of discretizing the panorama into a number of planes required that the said planes had to be positioned in an uniform manner on the surface of the sphere. For this purpose, a solution based on physics formula for charge repulsion (inverse square law) was used to diffuse random points in an uniform arrangement. An implementation developed by Jonathan D. Lettvin[4] was used.

The number of planes, their size and resolution are adjustable parameters of OT. However, in practice, only two of them are of interest for the experiments presented in this thesis: the *number of planes* and the *plane size* expressed as a ratio of the FOV of the current camera[5]. According to our experiments, the number of planes must be high enough to cover the entire surface of the panorama and the plane size must be a little larger than the camera's FOV (a ratio ranging from 1.2 to 1.4 provided good results

[2]While it may seems like discretizing the panorama into smaller planes comes down to using individual images, using a panorama presents many advantages: the panorama creation process tend to remove outliers that appear only on one image (in the image set used to assemble the panorama), it greatly reduces the search space for matching against the live camera image by generating planes that have a larger FOV than the one offered by the camera and it has also the ability to generate the planes at any location on the sphere (this is useful if the current live image is located at the boundaries of two images).

[3]http://opencv.willowgarage.com/wiki/

[4]The code is available at http://local.wasp.uwa.edu.au/~pbourke/geometry/spherepoints/

[5]Such representation of the plane size allows to automatically compute the correct resolution for any given plane, reducing sources of error. In practice, it is possible to set the plane size and its resolution independently, but in order to provide good results, they must be proportional to the corresponding parameters of the live camera (the live camera feed will be compared to the panorama projection on these planes).

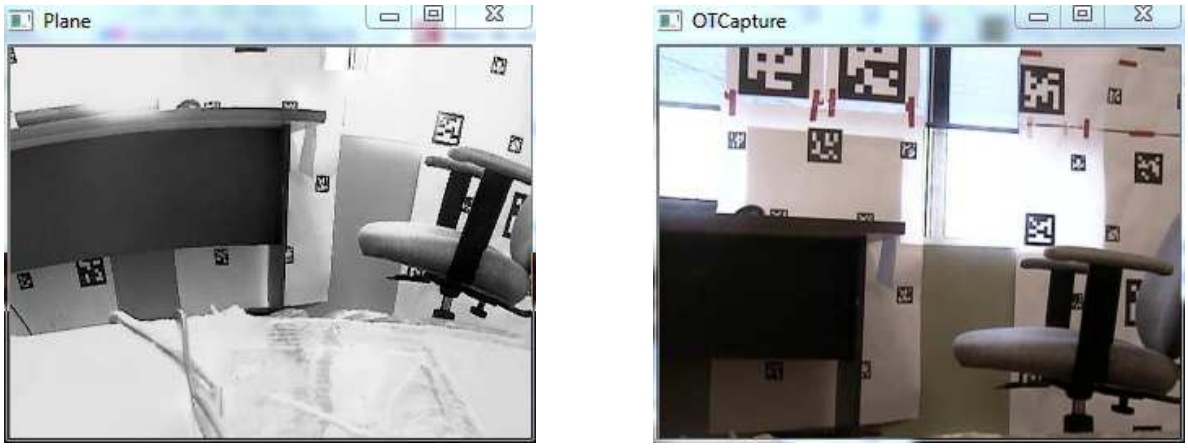

Figure 3.2 – Difference between the (left) flattened spherical projection and the (right) planar projection. The spherical projection does not keep the lines straight and distort the image; it is therefore impossible to compare SURF features extracted from each image respectively.

without affecting the performances too much when using between 20 and 30 planes[6]). Having a plane size larger than the camera's FOV allows more robustness when selecting the best plane to be matched with the input camera image. An illustration of the panorama coverage is proposed in Figure 3.4. Two panoramas are presented for which the number of planes is equal but the plane size ratio is different.

3.2.2 Tracking Phase

Once the feature map has been built, the orientation tracking phase begins (see Figure 3.1 for the workflow of the tracking phase). First, a frame is captured from the live video feed. The frame is then corrected for radial and tangential distortions[7] before applying the image feature extraction algorithm (SURF in this case). The features are then compared to the features previously extracted on the different planes of the map. The plane where the largest number of matches is found is selected. If the number of matches is sufficient (the minimum number of matches required to compute an

[6] As a rule of thumb, the plane size ratio should always be higher than 1.0 to ensure that a frame captured by the camera can fit on one plane, but not too high in order to prevent any important decrease in performance. The number of planes is then adapted to ensure a full coverage of the equirectangular panorama.

[7] Radial distortions is caused by the camera's optics. This has the effect of bending straight lines into slight curves. Tangential distortions is caused by misalignment of the camera's sensor with the camera's optics. This introduces slight skew in the image. To correct these distortions, the camera's distortion parameters must be determined through camera calibration. The image is then remapped to correct the previously calculated distortions. [32]

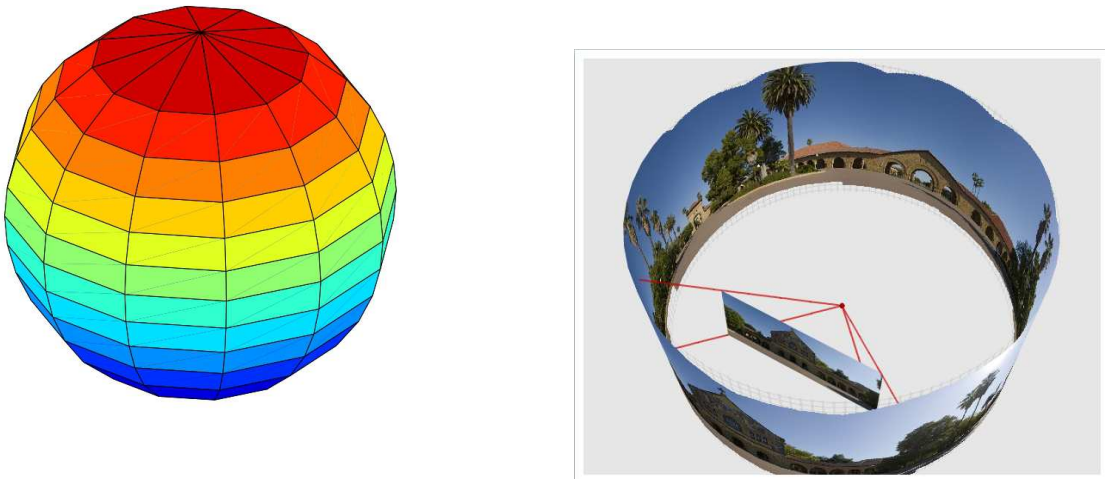

Figure 3.3 – (left) An example of a discretized sphere. (right) An illustration of the panorama reprojection process. The pixels of the generated panorama are reprojected on the individual planes of the discretized sphere. (*note: (left): It must be noted that in OT's specific case, the planes of the discretized sphere will all be the exact same size and may overlap each other in some regions (unlike what is seen in the illustration)*) (*source: (left): http://goo.gl/h5lKJ (right): http://goo.gl/IQFLR*)

homography is 4), the homography is computed using RANSAC [58] for selecting the subset of points that minimize the back-projection error. In this specific experiment, algorithms available in OpenCV library were used to compute the homography. The homography allows to determine the camera frame's center on the plane. This position is then reprojected on the virtual sphere (*i.e.* the panorama projection surface). This position on the surface of the sphere represents the orientation of the camera in relation to the panorama.

Optionally (not shown in the workflow diagram), a pose refinement feature is proposed. This pose refinement technique is useful when the live camera image is located at the junction of different planes. In this specific case, there may be enough features to compute an homography using a given plane of the discretized sphere, but not enough to make sure this homography is very accurate. The pose refinement principle consists in using the best fitting plane, initially selected (cf. previous paragraph), to compute an approximate homography and then using this first orientation to refine the result further. A temporary plane, centered at the camera's orientation (previously computed), is created. The panorama is then reprojected on this plane resulting in a "synthetic" image that is as similar as possible to the image captured from the live camera. SURF features are extracted again on this new image and compared to the features extracted from the live camera image. Finally, the homography is once again computed from the

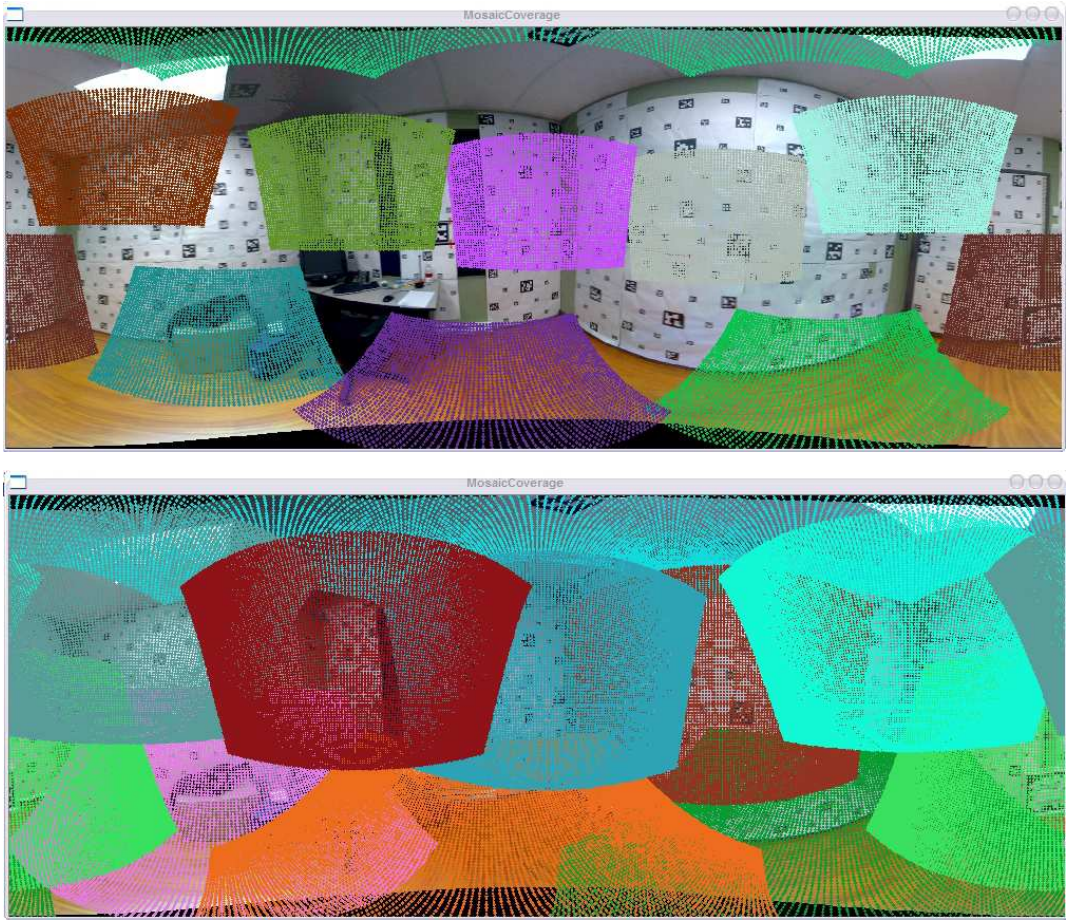

Figure 3.4 – Example of plane coverage using an identical number of planes but a different plane size ratio, respectively equal to (top) 1.0 and (bottom) 2.0

new set of matches.

3.3 System Evaluation

Sub-objective 3 aims at validating the potential of tracking approaches for mobile AR GIS applied in an architecture and geo-engineering context. In order to carry out such validation for OT approach, its accuracy (and indirectly, the precision[8]) had to be measured. OT's accuracy assessment was carried out with two external measurements tools used as ground-truth references: a MEMS-type orientation tracker from the company XSens[9] and a PTZ camera from the company Axis[10].

The data sets involved in the accuracy assessment experiments consisted of two components: a panorama (used to build the feature map) and a video. Both components were recorded from the exact same location, using a tripod. The video consisted of a sequence of frames captured using a camera setup on the tripod. The external measurement tool measured an orientation value for each frame (information about the setup will be provided below – Section 3.3.1.1). The experiments consisted in feeding OT with the video sequence and in recording the orientation computed for each frame of the video. Comparison of orientation values provided by OT and those measured by the external measurement tool allowed assessing the accuracy performance of the proposed orientation tracking solution.

3.3.1 Using a MEMS -type orientation tracker

3.3.1.1 Evaluation setup

To assess the accuracy of OT, the XSens MTx[11], a 3DOF orientation tracker, was used. This device has a static accuracy of 1° and a dynamic accuracy of $2° RMS$[12][13].

[8] As seen in Section 2.1.2, accuracy and precision are two distinct but intertwined concepts. However, even if accuracy remains the principal focus of this research, accuracy without precision is to be avoided.

[9] www.xsens.com

[10] www.axis.com

[11] http://www.xsens.com/en/general/mtx

[12] RMS: Root Mean Square. It is a statistical measure of the magnitude of a varying quantity.

[13] It is worth noticing that, even if the MTx is less accurate than the required accuracy calculated at Section 2.1, it was used primarily to validate OT and have an overview of what could be achieved with this approach.

The web camera (a Logitech QuickCam for Notebooks Pro) and the MTx were setup on the same platform (Figure 3.5) and for each frame captured by the camera, the orientation value returned by the MTx was saved. To avoid extensive inter-sensor calibration, the tripod's rotational movement was locked in order to free only one axis (*heading* – also called *yaw* (Figure 3.6)) and an absolute rotational representation (called Rodrigues parameters[59]) was used. Using such representation, the rotation is expressed as a rotation axis in 3D space (a unit 3D vector) and an angle of rotation[14] (Figure 3.7). Given the operational context where only one rotation axis is free on the tripod, the rotational representation allows dismissing the vector's orientation and keeping only the angular value as the significant measurement for the subsequent comparison with the reference data set[15].

Tests were conducted in a controlled environment including only static components (i.e. no moving objects; constant lighting). The panorama created for this test is shown at Figure 3.8. The resolution of the camera was set to 320x240; this is quite low, but is here justified by the hardware limitations of this project (Section 1.2): native video resolution of the cameras that come equipped with mobile devices (such as smartphones or tablets) are often of resolution of 320x240 or 640x480. OT's internal panorama representation was set to 20 planes with a plane size ratio (expressed as a ratio of the FOV of camera) of 1.2 (see Section 3.2.1.2). The experiments consisted in rotating slowly the platform supporting the camera and XSens sensor up to a specific angle while pausing a few time (arbitrarily) along the way before bringing the platform back to its starting position. This rotation process was repeated four times.

3.3.1.2 Results

The graphs representing the recorded data sets is shown in Figure 3.9 and 3.10. The graphs show that the orientation values measured by OT and the XSens sensor respectively are consistent and follow the same progression. However, some differences are noticeable:

- Several obvious outliers are located near the peaks (frames $\sim$ [350,600], $\sim$ [1050,1150] and $\sim$ [1300,1400]). These erratic values are related to OT only and they are not

[14] Rodrigues parameters can also be expressed using a compact form where only a 3D vector is used. The vector's norm then represents the angle of rotation.

[15] Keeping only one free axis on the tripod helps keeping the rotation comparison simple by removing the need of error-prone calibration steps. The rotation axis can be oriented in any way, as long as the rotation is kept on this single axis for the evaluation. It is therefore generic and suited for the accuracy evaluation of OT.

Figure 3.5 – The evaluation setup using XSens's MTx

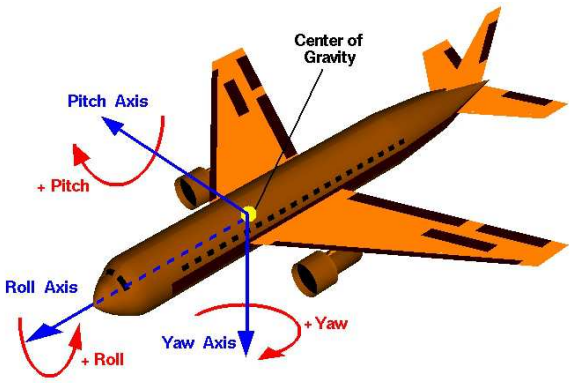

Figure 3.6 – Illustration of the *roll*, *pitch* and *yaw* rotation axes using the airplane example (*source: http://goo.gl/bNNuO*).

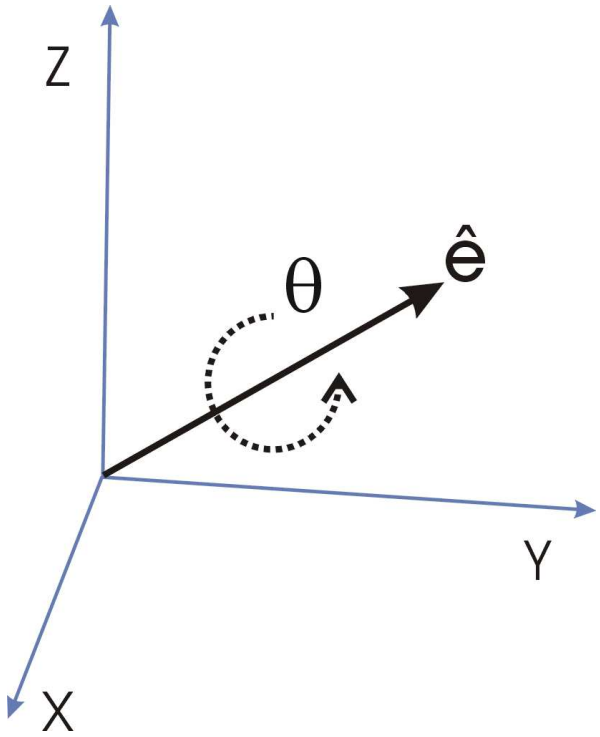

Figure 3.7 – A visualization of a rotation angle of θ by an Euler axis $\hat{e}$.

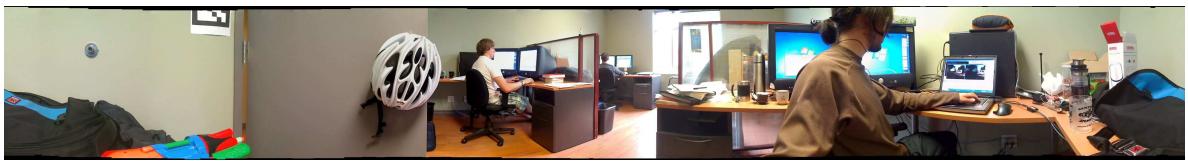

Figure 3.8 – Panorama captured in office to evaluate the accuracy of Orientation Tracker using the MTx.

negligible. However, given the large discrepancy between these erroneous values and the a priori relevant orientations, these outliers could be removed using, for instance a Kalman filter [60, 61];

- The error amplitude does not seem constant throughout the graph. The first peak (frames $\sim$ [150,800]), for example, shows an error range that evolves almost linearly up to orientation discrepancy between OT and MTx of 0.17 rad (*i.e.* about 10°). One hypothesis explaining this observation would be a bad calibration between the camera and the MTx; indeed, as shown in Figure 3.5, the setup has not been calibrated correctly since the camera's center MTx's center are not located at equal distance from the center of rotation of the tripod's head. This would be sufficient to create a parallax error explaining the offset observed;

- When observing Figure 3.10 more closely, a small drift is observable at the beginning of each plateau (caused by the pauses introduced in the rotation of the platform – if the platform stops moving, the orientation values should remain constant). While small in these observations, mainly because of the shortness of the pauses, this drift could easily explain the variations of the error amplitude. To validate this hypothesis, the MTx orientations was recorded for a very small number of orientation variations. Results can be observed in Figure 3.11. These results highlight the fact that the sensor experienced a strong drift in its orientation measurements[16]. To validate such phenomenon was not due to a malfunctioning of the device, the drift tests were conducted with another MTx sensor which yielded similar results. Drift tests were also ran outdoor in order to move the sensor away from potential magnetic fields in the experiment environment, but the same observations were recorded (having no equipment to measure the strength of the magnetic fields in the test environment, it was not possible to assess the impact of the magnetic field). Since according to the specifications, the MTx is drift-free, the support team at XSens was contacted. Unfortunately, they were not able to identify the source of the drift, beside perhaps a bad initial calibration of the internal MTx components.

These first results allowed to confirm that OT was indeed in working order; it was able to track the orientation and the orientation values were coherent. However, the MTx, that acted as our ground-truth measure for this experiment, showed important drift problems preventing any quantification of the accuracy of the orientation values returned by OT.

[16] This should not be observed since XSens claims that that this technology should be drift-free.

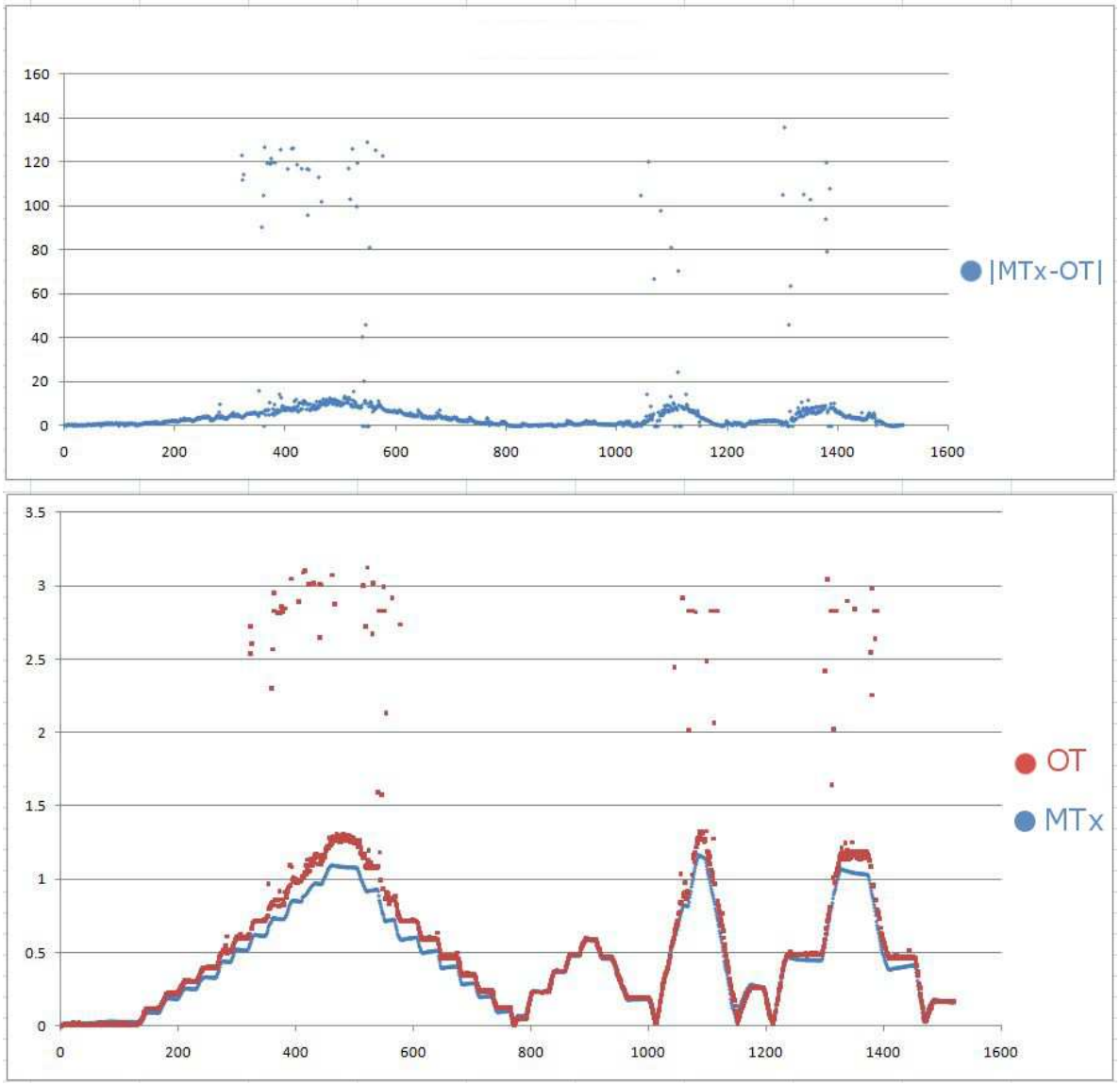

Figure 3.9 – Graphs representing the orientations provided by both the MTx and OT. The top one represents the absolute difference, in degrees, between the orientation values provided by both systems. The abscissa represents the number of frame captured and processed by the system and the ordinate is the absolute difference, in degrees, between the orientation values returned by the MTx and OT. The bottom graph shows the actual, raw orientation, in radians, of each system. An enlargement of this graph is proposed at Figure 3.10.

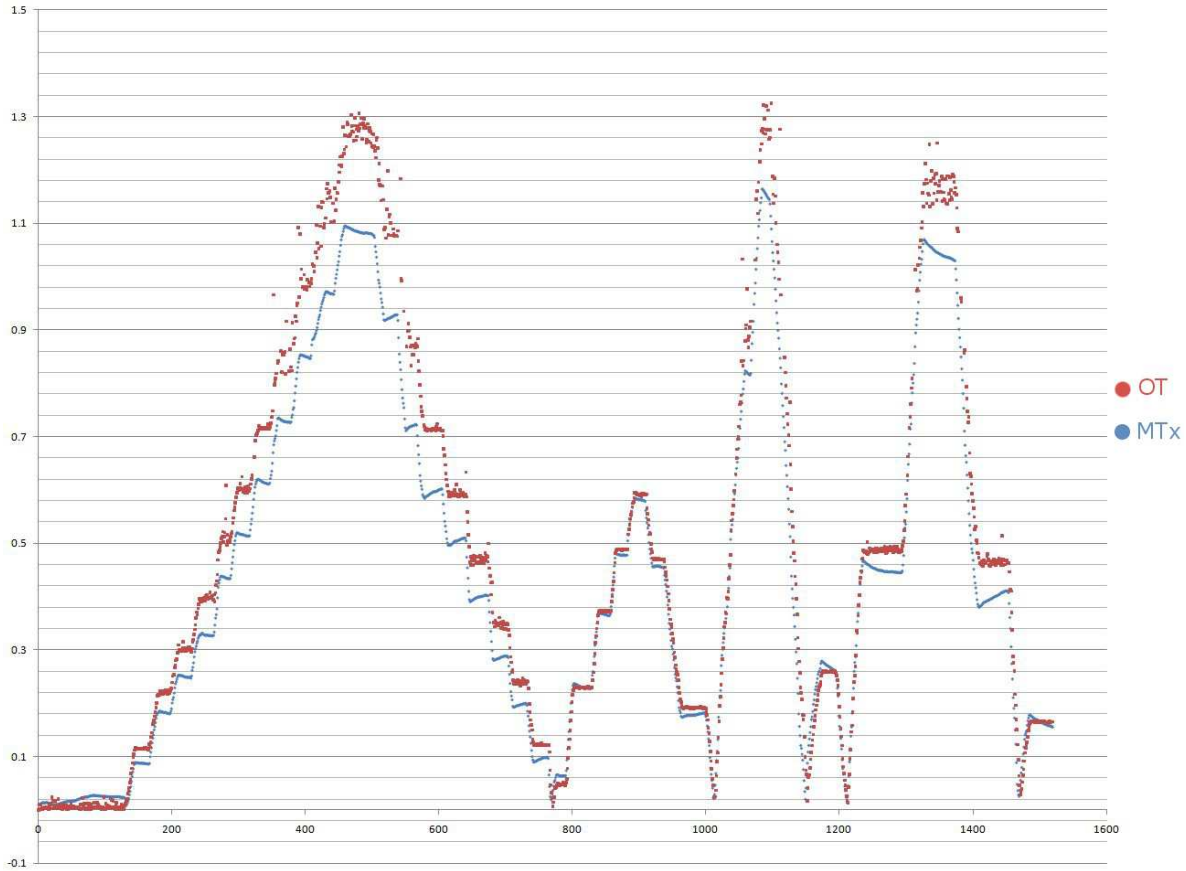

Figure 3.10 – Enlargement of the captured orientations by the MTx and OT shown at the bottom of Figure 3.9. The abscissa represents the number of frame captured and processed by the system and the ordinate is the returned orientation value, expressed in radians, of each system.

Figure 3.11 – Graph showing the drift of the MTx measurement, observed on a recorded session involving a limited number of orientation changes. The abscissa represents the number of successive orientation measurements done with the MTx. The ordinate represents an angular value, expressed in radians. The blue, red and green lines each represents one axis of rotation of the MTx (Roll, Pitch, Yaw). The purple distribution (theta) represents the absolute value of the rotation angle (using only the angular component of the Rodrigues parameter – see Sub-section 3.3.1.1). The drift zone are particularly obvious at ~3500 and ~5500 on the abscissa axis (highlighted by the two red squares).

3.3.2 Using the "Pan-Tilt" camera

The second set of tests was carried out using a PTZ (Pan-Tilt-Zoom) camera. Such a camera is often used for surveillance and security purposes as it can be controlled remotely. Typically, hardware consists of a camera mounted on a support which can be rotated through given axes (the PTZ camera can rotate about two different axes – pan and tilt[17]) using dedicated motors. Commands and requests can be sent to the camera via a LAN[18] connection.

The Axis 213 PTZ[19] (Figure 3.12) manufactured by the company Axis was used in this experiment. The model specifications involved a precision of $\pm 0.23°$[20] in both relative and absolute displacements which fulfilled the accuracy and precision requirements for OT performance assessment.

However, it stemmed from the preliminary tests that the PTZ camera motors were indeed precise, but not accurate. Also, image deformations in the range of several pixels were observed (up to 7 pixels in some images). The main hypothesis explaining this behavior would be that the CCD was slightly off-center relative to the camera's rotation center.

Consequently, the PTZ camera was not considered reliable and accurate enough to be used as a ground-truth measurement tool for testing OT's accuracy and precision.

3.3.3 Stability test

A stability test was conducted to further explore the capabilities of panorama-based tracking. A stability test does not aim at measuring the accuracy, but instead is more focused on the precision. Given that ground-truth measurements have proven to be difficult to get, precision results can be obtained more easily and can help in assessing the limits of the system. Indeed, as OT is a real-time system, measures are not accumulated over time; this prevents the system to be *statistically* accurate. OT therefore needs to be accurate for each of its measurements. Precision then becomes a good measure quantifying the *potential* of OT to be accurate. In other words, if OT has a precision below its required accuracy, it means this solution will not meet the accuracy

[17] Corresponding respectively to *yaw* and *pitch* angles of Figure 3.6.
[18] LAN: Local Area Network
[19] http://www.axis.com/products/cam_213/
[20] Specified by a technical support person of the Axis company in an email communication.

Figure 3.12 – The "pan-tilt" camera Axis 213 PTZ (*source: http://goo.gl/CQ5vM*).

Figure 3.13 – Panorama of the room used to run the stability test of OT

requirements of a mobile AR GIS dedicated to geo-engineering[21].

The stability test consisted in building a panorama (Figure 3.13) of an experimentation lab, involving series of surveyed targets on the walls, and then capturing a sequence of 200 frames from the exact same position, without moving the camera at all. The experiment was ran with input videos at two different resolutions (320x240 and 640x480) in order to assess if higher resolution resulted in important error reduction. Ideally, the computed orientation values should be perfectly stable and show little to no variation. Also, the variations should be lower than the precision required for OT; if OT cannot retrieve its orientation in a reproducible manner when presented with a video consisting of frames recorded from the exact same point of view, it is very unlikely that the precision would improve when moving the camera around.

[21] See Section 2.1.2 for a reminder of the difference between *accuracy* and *precision*.

3.3.3.1 Results

The stability results are synthesized in Table 3.1. The metric used to assess OT's stability is the standard deviation (stdev). The number and type of error made by OT during the tests are also included in Table 3.1. When OT was unable to determine the orientation of a frame, the error was considered as a "true negative". Such error, while important to minimize, is not critical as the system can make the correct diagnosis about the value it provides; OT can therefore reject it on its own. A "false positive" error, on the other hand, is far more problematic since a completely false value is considered as a correct orientation. The standard deviation calculated without the false positive errors that were relatively obvious (noted "stdev*") is also included in Table 3.1. This computation was carried out by considering that these false positive values would be easy to remove with a filtering algorithm. In the proposed context, any values greater than 60° (the average is around 51°) are considered as outliers.

The first stability experiments demonstrated high variability. Indeed, the standard deviation of the measurements was 0.543°, even when the outliers were manually removed. One hypothesis justifying this variability was the presence of noise generally included in all video capturing devices. To assess this hypothesis, the same experiments were conducted again while applying simple Gaussian filters of varying sizes (i.e. none – also noted as 1x1 –, 3x3 and 5x5) to both the panorama and the input frames extracted from the captured video. Next section provides a detailed analysis of Table 3.1 results.

3.3.3.2 Analysis

It stems from the results in Table 3.1 that the number of errors can be very high, reaching 71 in the worst case[22], especially when using video images at the lower resolution setting. The highest error rates occur when Gaussian filters applied on the panorama are of larger size than the ones applied on the video (in which case the panorama is more blurred than the input video). The panorama is therefore more blurred than the input video. Indeed, filtering the panorama with larger Gaussian filter means removing possible smaller features. Thus, only large features are detected and included in the map. Considering that the input video is not as blurred, many lower scale features, that can be noise in the image, are extracted. However, they will have no possible match in the map. This greatly increases the possibility for mismatches to occur. Also, error rate is two to three time higher with the lower resolution input video. Indeed, Gaussian filter impact is stronger on the lower resolution images. A 3x3 filter has halved

[22]Third row: 67 true negative errors and 4 false positive errors.

Resolution	Video blur	Pano blur	Avg	Avg*	Stdev	Stdev*	True negative	False positive
320x240	1x1	1x1	52.117	51.645	6.635	0.543	4	1
320x240	1x1	3x3	51.467	51.467	0.466	0.466	62	0
320x240	1x1	5x5	52.623	51.556	8.740	0.554	67	4
320x240	3x3	1x1	53.063	51.651	12.226	0.321	1	3
320x240	3x3	3x3	51.536	51.536	0.374	0.374	8	0
320x240	3x3	5x5	51.715	51.715	0.411	0.411	0	0
320x240	5x5	1x1	54.233	51.632	14.887	1.319	12	6
320x240	5x5	3x3	51.527	51.527	0.427	0.427	31	0
320x240	5x5	5x5	51.753	51.753	0.522	0.522	20	0
640x480	1x1	1x1	52.003	52.003	0.565	0.565	14	0
640x480	1x1	3x3	52.115	52.115	0.526	0.526	14	0
640x480	1x1	5x5	51.886	51.886	0.699	0.699	2	0
640x480	3x3	1x1	51.848	51.848	0.486	0.486	0	0
640x480	3x3	3x3	52.028	52.028	0.413	0.413	0	0
640x480	3x3	5x5	51.857	51.857	0.567	0.567	1	0
640x480	5x5	1x1	51.697	51.697	0.541	0.541	5	0
640x480	5x5	3x3	52.009	52.009	0.424	0.424	0	0
640x480	5x5	5x5	52.083	52.083	0.526	0.526	0	0

Table 3.1 – Results from the stability test using a completely still video consisting of 200 frames. OT implementation relies on the SURF descriptors (from points identified by the SURF detector) computed using the library OpenCV. Results are expressed in degrees. The *Resolution* is the resolution of the video, *Video blur* and *Pano blur* are the size of the Gaussian filter applied to the video and panorama respectively, *Avg* and *Avg** are the average orientation values, with and without outliers respectively, provided by OT and *Stdev* and *Stdev** are the standard deviation of the orientation values, with and without outliers respectively, provided by OT. Finally, there are the *True negative* and *False positive* count.

the smoothing effect on a 640x480 image than on a 320x240 image. Higher resolution images also have four times more pixels, resulting in potentially more features. Higher number of features raises the probability for a correct match to be identified.

As regard the standard deviation (*with* outlier rejection – Stdev*) metric, some variations, in the range of 0.321° to 1.319°, can be noticed. They appear to be a function of the blur applied to the panorama and the input video. From the data collected, it is hard to identify a causality effect between the size of the Gaussian filter (i.e. the ratio of blur) and the best performance (lowest values of "Stdev*"). Even so, looking only at the standard deviation values, the "best" Gaussian filter configuration yields a standard deviation of 0.321°. This is higher than the minimal accuracy and precision requirements calculated in Section 2.1.4 (the application requiring the lowest accuracy still has a threshold value of 0.239°). It means that, with a perfectly stable camera, with only the noise of the CCD altering the input image, and in an ideal environment (enough features, no occlusions, good illumination of the scene), the orientation computed by OT is varying by almost 0.5°. This is too high for this project's requirements.

These high variations highlight the fact that there are still error sources that need to be controlled. The next section will provide a review of the various sources of errors influencing the stability.

3.4 Error Sources

Orientation Tracker proved to be a tracking system that works but that presents some stability issues (Section 3.3.3). These results means that some error sources were not accounted for when designing the system and that these error sources influenced the results negatively. This section provides a list of the error sources that could have influenced the results.

3.4.1 SURF implementation

The SURF implementation used in the proposed project could be the first source of OT relative instability. Indeed, as reported by Gossow *et al.* [62], there seems to be a performance variability between the different SURF implementations that cannot be easily dismissed. Furthermore, "performance" does not only include the processing speed, but also the matching rate. To test this hypothesis, the stability test detailed at

Resolution	Video blur	Pano blur	Avg	Avg*	Stdev	Stdev*	True negative	False positive
320x240	1x1	1x1	52.281	51.745	7.603	0.517	0	1
320x240	1x1	3x3	51.851	51.851	0.516	0.516	0	0
320x240	1x1	5x5	52.035	52.035	0.468	0.468	0	0
320x240	3x3	1x1	52.219	52.219	0.729	0.729	0	0
320x240	3x3	3x3	52.077	52.077	0.743	0.743	0	0
320x240	3x3	5x5	51.924	51.924	0.707	0.707	0	0
320x240	5x5	1x1	57.096	51.704	22.770	0.683	2	11
320x240	5x5	3x3	51.856	51.489	5.190	0.476	2	1
320x240	5x5	5x5	51.442	51.442	0.440	0.440	3	0
640x480	1x1	1x1	52.315	51.714	7.208	0.573	9	2
640x480	1x1	3x3	54.156	51.160	13.739	0.725	1	11
640x480	1x1	5x5	53.281	51.538	12.305	0.599	9	5
640x480	3x3	1x1	51.860	51.860	0.575	0.575	9	0
640x480	3x3	3x3	51.657	51.414	3.481	0.595	0	1
640x480	3x3	5x5	51.773	51.773	0.588	0.588	0	0
640x480	5x5	1x1	51.599	51.438	2.265	0.247	5	1
640x480	5x5	3x3	51.395	51.395	0.362	0.362	0	0
640x480	5x5	5x5	51.534	51.534	0.374	0.374	0	0

Table 3.2 – Results from the stability test using a completely still video consisting of 200 frames. OT implementation relies on the SURF descriptors (from points identified by the SURF detector) computed using the library OpenSURF. Results are expressed in degrees. The *Resolution* is the resolution of the video, *Video blur* and *Pano blur* are the size of the Gaussian filter applied to the video and panorama respectively, *Avg* and *Avg** are the average orientation values, with and without outliers respectively, provided by OT and *Stdev* and *Stdev** are the standard deviation of the orientation values, with and without outliers respectively, provided by OT. Finally, there are the *True negative* and *False positive* count.

Section 3.3.3 (which relied on the OpenCV implementation) were conducted again using the OpenSURF library [7]. While, according to OpenSURF author's, less optimized for real-time performance, the implementation is compliant with the original SURF algorithm as proposed by Bay *et al.* [36]. The stability results are displayed in Table 3.2.

When comparing these results with the ones in Table 3.1, the main observation concerns the error rate. OpenSURF performed much better in that standpoint (decreasing the numbers of error by 100% in many cases). Indeed, OpenSURF seems much more consistent than the implementation proposed by OpenCV[23]. This tends to confirm the results found in Gossow *et al.* [62] study: different SURF implementations yield different results and the difference is hardly negligible.

This variability between the different implementations of the same technology def-

[23]That is, at the time of this writting (OpenCV version 2.1.0).

initely is an uncertainty factor in the proposed tracking system. Gossow *et al.* found that the latest version of OpenSURF yields better results than other implementations. Nevertheless, stability score of OT using OpenSURF still is insufficient (a standard deviation averaging 0.5°), showing that this is not the only error source.

3.4.2 Accuracy of the point of interest detector

Performing image matching or object recognition based on high level feature descriptors like SURF or SIFT, generally consists of two main operations: detection of points of interest and description of these points using high level features. Assuming that incorrect matches between the features can be filtered out, any inaccuracies in the position and/or orientation computed by a NFT solution should be related to the *detection* process. Indeed, when a match is found between two features, the *location* of the features in the images are used to compute the transformation involved in the determination of the position and/or orientation of a frame in relation to its reference system (as for OT, this transformation is an homography and the reference system is a panorama). The feature location is determined solely by the detection algorithm. Therefore, to increase the accuracy of a solution based on natural features in image (as is OT), efforts should be focused on the point of interest detector.

The main criteria used in the literature to evaluate the performance of a detector is the *repeatability*. This criteria needs to be investigated in further details to determine its relationship with the accuracy of the currently evaluated detection algorithm.

3.4.2.1 Repeatability

Given two images representing, in part or in whole, the same scene under different recording conditions, a detector can be qualified as "good" if it can identify the same points on the two images in a reliable and accurate manner. In order to measure this performance, Schmid *et al.* [6] first proposed the *repeatability rate* as a metric. This metric was later used as the main performance criterion in the most widely used benchmark for detector evaluation, which was proposed by Mikolajczyk *et al.* [63]. This benchmark was also used by Evans [7] when evaluating the OpenSURF library.

Schmid *et al.* [6] defines the repeatability as follow:

> *Repeatability* signifies that detection is independent of changes in the

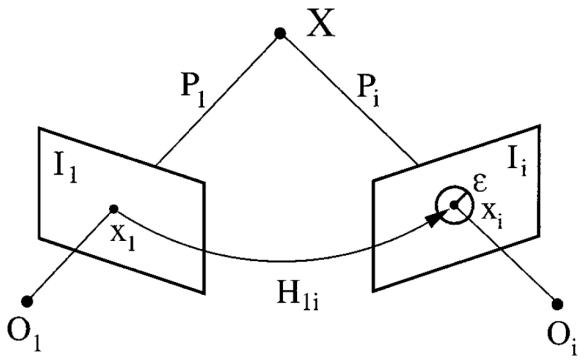

Figure 3.14 – The point x_1 and x_i are projections of the 3D point X into images I_1 and I_i: $x_1 = P_1X$ and $x_i = P_iX$ where P_1 and P_i are the projection matrices. A detected point x_1 is repeated if x_i is detected as well. It is ϵ-repeated if a point is detected in the ϵ-neighborhood of x_i. In the context of planar scenes the point $x_1$and x_i are related by the homography H_{1i}. (*source: [6]*)

> imaging conditions, i.e. the parameters of the camera, its position relative to the scene, and the illumination conditions. 3D points detected in one image should also be detected at approximately corresponding positions in subsequent ones.
>
> [...]
>
> *Repeatability rate* is defined as the number of points repeated between two images with respect to the total number of detected points.

The repeatability concept is illustrated in Figure 3.14. Given a 3D point X that is projected on two different images I_1 and I_i using projection matrices P_1 and P_i, resulting in projected points $x_1 = P_1X$ and $x_i = P_iX$. In a planar scene, the unique relation that exist between x_1 and x_i can be represented by an homography H_{1i} so that $x_i = H_{1i}x_1$. Given the error involved in the projection process, an error range ϵ need to be included in the equation. Therefore, for x_1 to be correctly repeated, x_i must be detected in image I_i as well. It is ϵ-repeated if a point is detected in the ϵ-neighborhood of x_i.

In terms of accuracy, the concept of ϵ-repeatability is fundamental as ϵ is the threshold value that defines what is a repeated point and what is not. Therefore, repeatability rate is bound to increase as ϵ increases. However, that also means that *a higher repeatability rate does not imply a higher accuracy of the detected points.*

3.4.2.2 OpenSURF performance evaluation

When he proposed the OpenSURF library, Evans [7] used the image detector benchmark proposed by Mikolajczyk *et al.* [63] to evaluate his library (Bay *et al.* [36] used the same benchmark to test the SURF algorithm when it was first introduced). The results are presented in Figure 3.15 and Figure 3.16 (as OT is a system that will mainly deals with rotation, scaling and light changes – viewpoint changes are not considered as it is taken for granted that the user is static or that he uses a tripod – only these results are presented). It can be observed that OpenSURF performs relatively well, substaining a repeatability rate higher than 50% for all the studied conditions. Sustaining a repeatability rate of 50% is critical; below that threshold, no statistical method can distinguish inliers from outliers.

The problem with the results provided by the benchmark is that it uses an error range value of $\epsilon = 1.5$ pixels . This means that each image feature that is matched (and by assuming a 100% matching rate for the description and matching algorithm), the position of the said feature is accurate up to 1.5 pixels, most of the time. This threshold value is too high for the accuracy requirements of this research as most of the task compliant with an AR GIS require an accuracy higher than 1 pixel. A smaller threshold should therefore be used ($0.2 < \epsilon < 1$ pixel as determined in Section 2.1). Tests should be conducted again with the Mikolajczyk *et al.* [63] data set with a smaller threshold. Schmid *et al.* [6] have already run tests using an error range value of $\epsilon = 0.5$ pixel, but at the time of publication of Schmid *et al.*'s work, modern detectors such as Fast-Hessian or FAST [34], two detectors commonly used in conjunction with the SURF descriptor for the point-of-interest detection step[24], were not developed. Supplementary tests should include these new detectors to assess if they improve former detectors such as the Harris corner detector [64] which, while presenting the best overall results in Schmid's study, showed a repeatability rate below 50% with a threshold ϵ equal to 0.5 pixel. That being said, it is not very likely that the SURF detector would provide a good enough accuracy. Indeed, Remondino [65] showed that both Harris corner detector and the Hessian operator (SURF uses an *approximation* of the Hessian matrix, mainly for performance purposes) consistently returned a shift of 1 pixel in position.

[24] Fast-Hessian is actually the point-of-interest detector proposed by Bay *et al.* [36] when they introduced the SURF algorithm.

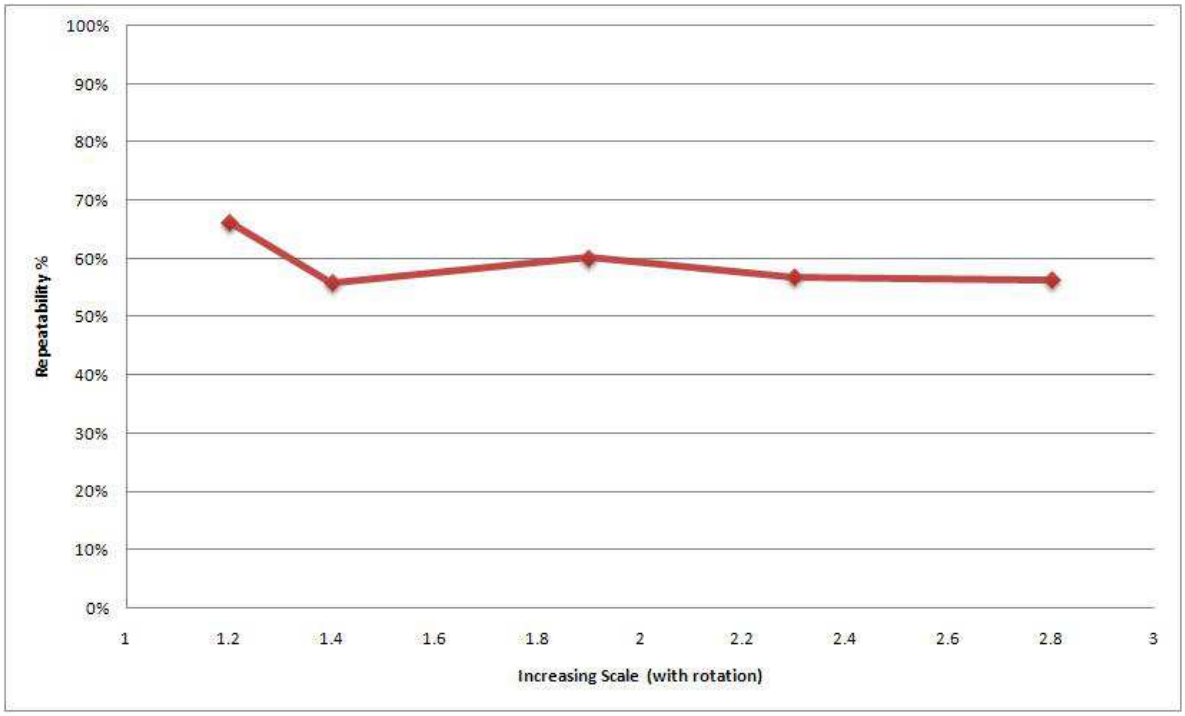

Figure 3.15 – OpenSURF repeatability rate under increasing rotation and scaling changes (*source: [7]*).

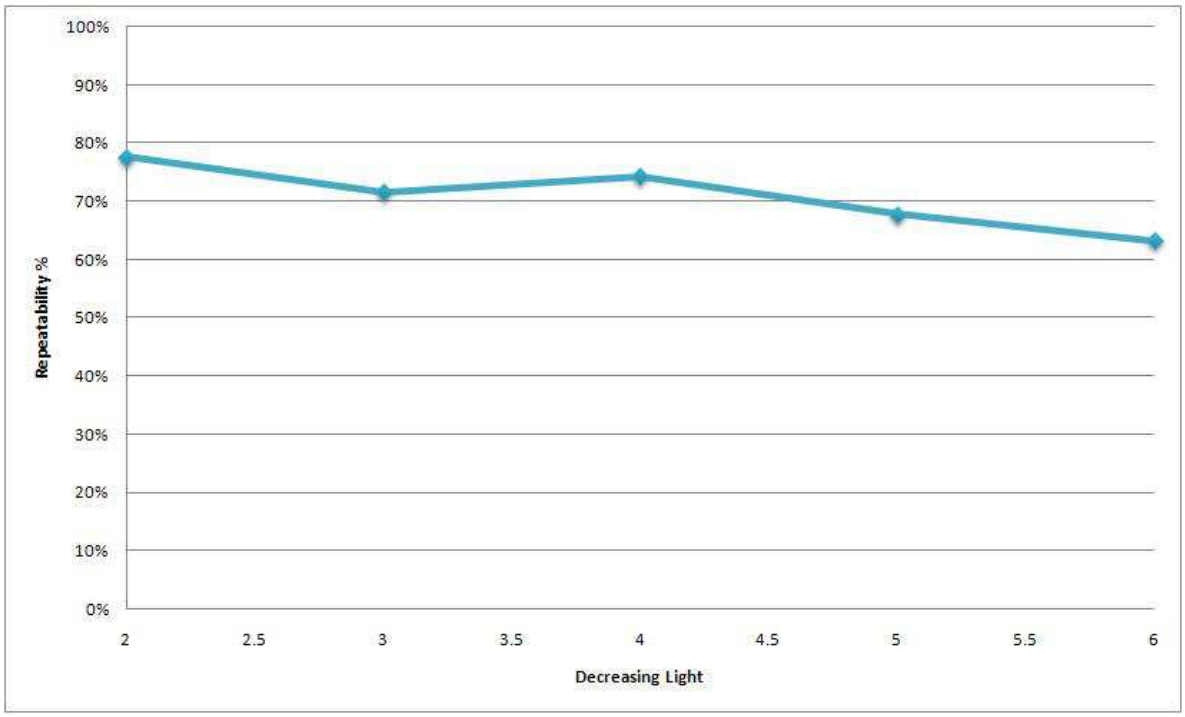

Figure 3.16 – OpenSURF repeatability rate under decreasing light conditions (*source: [7]*).

3.4.3 Hardware imperfections/limitations

When building panorama images in order to use them as an accurate representation of the reality, the hardware used to record the image matters. Even sophisticated cameras record images with some imperfections. These imperfections must be corrected. One example is the radial and tangential distortions induced by the lens of the camera. This type of distortions can be "easily" corrected, as it has been done with OT, through the calibration of the camera (*e.g.* using OpenCV library and a checkboard-like pattern printed on paper [32]). Once the intrinsic parameters of the camera are known, each image recorded by the camera can be corrected and used as if they were captured using a pin-hole camera[25].

However, there are other factors that may distort the images. When investigating accurate panorama building with the Axis PTZ camera (Section 3.3.2), Bilodeau [66] noticed that the hardware introduced variations that could not easily be dismissed. Namely, he observed that hardware misalignment could yield to distortion of many pixels in the images.

Also, one of the most limiting factor for the accuracy of the OT system is the *camera resolution*. It is obvious, at this point, that increasing the resolution of the captured images would bear important improvements of the accuracy of the system. Indeed, because OT uses the SURF algorithm that has an accuracy level expressed in pixels, doubling the resolution of the camera effectively reduces the angular value represented by one pixel in half and thus, doubles the accuracy of the system.

3.5 OT conclusion

Orientation Tracker is a system that retrieves the orientation of a user by comparing the live video feed of the camera with a previously captured panorama. The experiments conducted in this project and described in this chapter allowed to investigate the accuracy and stability of panorama-based tracking in various contexts where a high accuracy is required.

The overall accuracy for OT, when in motion, has not been measured quantitatively

[25] Calibration is not a perfect process and error will always remain. However, when realized carefully, the remaining error is usually very small and can be dismissed for the type of application that this project is interested in.

because of different hardware as well as ground-truth problems. However, stability tests were conducted and a variation of about 0.5° was measured. Such variation is very high considering that the accuracy threshold for the least demanding task should be approximately 0.3°. Three potential causes have been identified as the reason for such variation: the SURF implementation, the point of interest detector accuracy definition and camera hardware imperfections.

This section highlighted some of the limitations and challenges of panorama-based tracking. It also highlighted the difficulties to reach a very accurate and real-time tracking solution. The next chapter will consider another approach to high accuracy augmentation system while still using panorama images as its core.

Chapter 4

Panorama-Based Pose Refinement

Panorama-based pose refinement (PBPR) is an additional approach for high accuracy AR. This work is a direct follow-up to the experiments involving OT. This approach, like OT, relies on panorama images. However, it differs from OT in one main aspect: while OT is a *tracking* method, aiming to retrieve the rotational pose of the user's head for each frame of a video stream, the PBPR approach aims to produce high accuracy *static* augmentations on panoramic images produced from multiple images. Its main purpose is to assist the user in finding the virtual camera pose of the 3D model *corresponding* to a captured panorama as accurately as possible in order to augment it. Put simply, the virtual camera pose is the virtual counterpart to the real-world camera pose used to capture an image of a scene. The virtual camera pose determines from which point of view a given 3D model will be rendered. The challenge of AR is to determine the relation between the real-world camera pose used to capture an image of a given element and the virtual camera pose of a 3D representation of this element so that both representations can be aligned to produce an augmented view of a scene. Thus, in the PBPR approach, the goal is to determine which virtual camera pose will produce a point of view of the 3D element that will allow the best alignment possible with the already captured panorama. This approach is also a solution to some of the shortcomings identified in the experiments using OT as described in Section 3.4.

In this chapter, the general PBPR approach will first be presented along with the context that justified a change of strategy from tracking to offline virtual camera pose refinement. Then, the method will be described as well as the preliminary experiments conducted to assess its potential as a high-accuracy mobile AR GIS approach. The results of these tests will be analyzed before concluding. It should be noted that, due to time constraints, it was not possible to thoroughly and quantitatively test the accuracy of the approach. However, preliminary tests were conducted and enough data

was collected to draw *qualitative* conclusions on the potential of the overall approach as well as to identify potential problems and future work.

4.1 A Different Approach to High-Accuracy Panorama-Based Augmentations

In Chapter 3, a panorama-based tracking solution was proposed (OT). Experiments with this system highlighted the many limitations that prevented reaching the accuracy needed for architecture and engineering tasks. OT addressed the accuracy issue of mobile AR GIS as a tracking problem, since the tracking component is fundamental to AR and it strongly influences the mobile AR system performance. However, the tracking accuracy is not the ultimate goal but a step towards what really matters to a mobile AR GIS, that is, the *accuracy of the actual augmentation*. In other words, what is important is that the architect or the engineer, while observing an augmentation, can have the confidence in the decisions he takes based on the information that is displayed. PBPR is an approach that considers only the augmentation aspects: it only operates on static panorama images, mainly for simplicity and problem isolation purposes. Indeed, considering the relative complexity of the real-time and dynamic tracking problem, it was decided to isolate only the *augmentation* component of the problem. This simplification does not prevent useful conclusions to be drawn as the augmented panorama images, even if static, can still provide important decision support information (*e.g.* allowing to visually assess a given situation with its projected state in the case of a construction site monitoring task).

Panorama images present great potential for high accuracy AR: they provide context, they are relatively easy and cheap to produce and they could yield to pixel-precise augmentations (using computer vision algorithms, which can achieve pixel or sub-pixel precision). However, as identified in the last chapter (Section 3.2.1.1), there is an accuracy bottleneck related to the panorama construction process. That is, the overall accuracy of the augmentation cannot be higher than the accuracy with which the panorama has been built (*i.e.* the degree at which the panorama accurately represents the reality). The panorama creation process is flawed in that, when built from multiple individual images, it introduces distortions caused by some adjustments which are required to correctly "close the loop" of a 360° panorama. These distortions are hard to both calibrate and correct contrarily to distortions in a single image. Indeed, distortions in an individual image, which are mostly related to the imperfections of the hardware used to acquire the panorama, are relatively easy to correct through camera calibration

Figure 4.1 – Multiple images referenced together using Microsoft Photosynth without merging them into a single panorama image. Each image is accurate on its own. (*source: http://goo.gl/A8e7L*)

and image undistortion. Once undistorted, an individual image can be considered an accurate projection of the world (at least for the accuracy and precision requirements of this research). Therefore, a solution aiming to work around the inaccuracies caused by the assembly of images into a panorama could do so by only manipulating the individual images *composing* a panorama. This is the strategy used by the PBPR approach.

The simplest way to work at the image level while preserving the different advantages provided by the panorama image would be to cross-reference the individual images while skipping the merging process altogether. The result would be a set of individual images referenced together (Figure 4.1). Such "mosaics" can be assembled automatically with a system like Microsoft Photosynth[1] that also provides an intuitive interface to navigate such image mosaics.

The use of such mosaics, however, adds some constraints. The first one is that it is not possible to work with a single panorama image anymore since the individual images are not merged together. This also implies that there will be various degrees of overlap

[1] http://photosynth.net/

and slight offsets among the images (see Figure 4.1). The offsets among the images can add up to a fair amount. If a 3D model is overlaid over this, the augmentation will be very accurate within a small subset of individual images[2], but will display strong inconsistencies within others; this will result in a very poor overall augmentation. The proposed solution to this issue consists in determining the virtual camera pose of the 3D model on *each* individual image composing the mosaic (*i.e.* a slightly different virtual camera pose will be associated with each individual image to ensure that the augmentation is as accurate as possible for *each* single image). Therefore, it would ensure that when a user is inspecting a specific part of the mosaic (*i.e.* an individual image), he would always look at an accurate augmentation (even if only locally) (see Figure 4.2 for an example). This constraint is not a major hurdle in the geo-engineering and architecture context of this research. Indeed, the field of view offered by a single image is generally big enough to display all the important elements required to take a decision. This is mainly due to the fact that the accuracy requirements of all the tasks identified in Section 2.1.1 are relatively proportional to the distance between the user and the augmented object. Moreover, it would be possible to use an interface similar to Photosynth that would provide the user with an overall view of the project area and thus some context while enabling interaction with individual images.

The central problem of this approach can be stated as follow: how to determine the virtual camera pose for *each* individual image of the mosaic, considering that a typical panorama is composed of 50-100 individual images? The proposed approach relies on a manual initialization step that matches elements of the model with corresponding items in the mosaic. As mentioned previously, the virtual camera pose resulting from an initialization over a mosaic will be accurate for some individual images while some inconsistencies will be observed in others. The idea is then to use this first virtual camera pose approximation (realized on the mosaic) as a starting point to further refine the virtual camera pose automatically for *each* single image of the mosaic so that the virtual camera pose is adjusted specifically for each single image. This is achieved by using an adaptive form of line tracking [68][3]. In a nutshell, the PBPR approach allows a user to generate as many individually refined virtual camera pose as there are images composing the mosaic by realizing a single manual initialization step. A summary of the proposed workflow for the PBPR is illustrated in Figure 4.3.

[2]Most likely, the images that will diplay the most accurate augmentation will be the ones on which the initialization points were selected during the manual registration process [67].

[3]Usually, vision-based markerless tracking uses line tracking or texture (this includes texture patches, SURF, SIFT, etc.). Line tracking has some advantages over texture patches in that, if we want to use the information provided by the 3D model, we do not need a textured 3D CAD model (which is not the norm in the geo-engineering industry). Moreover, lines of the model are much more likely to be steadily observed in a real-world capture over time as the texture information is more unreliable over time (season changes, daylight, occlusions, etc.).

Figure 4.2 – Example of an AR visualization using individual images referenced in a system like Microsoft Photosynth. It can be noticed that the augmentation stops at the different edges of the current image due to the offsets between the images thus avoiding inconsistencies in the augmentation. (*source: Modified from http://goo.gl/LRDJT*)

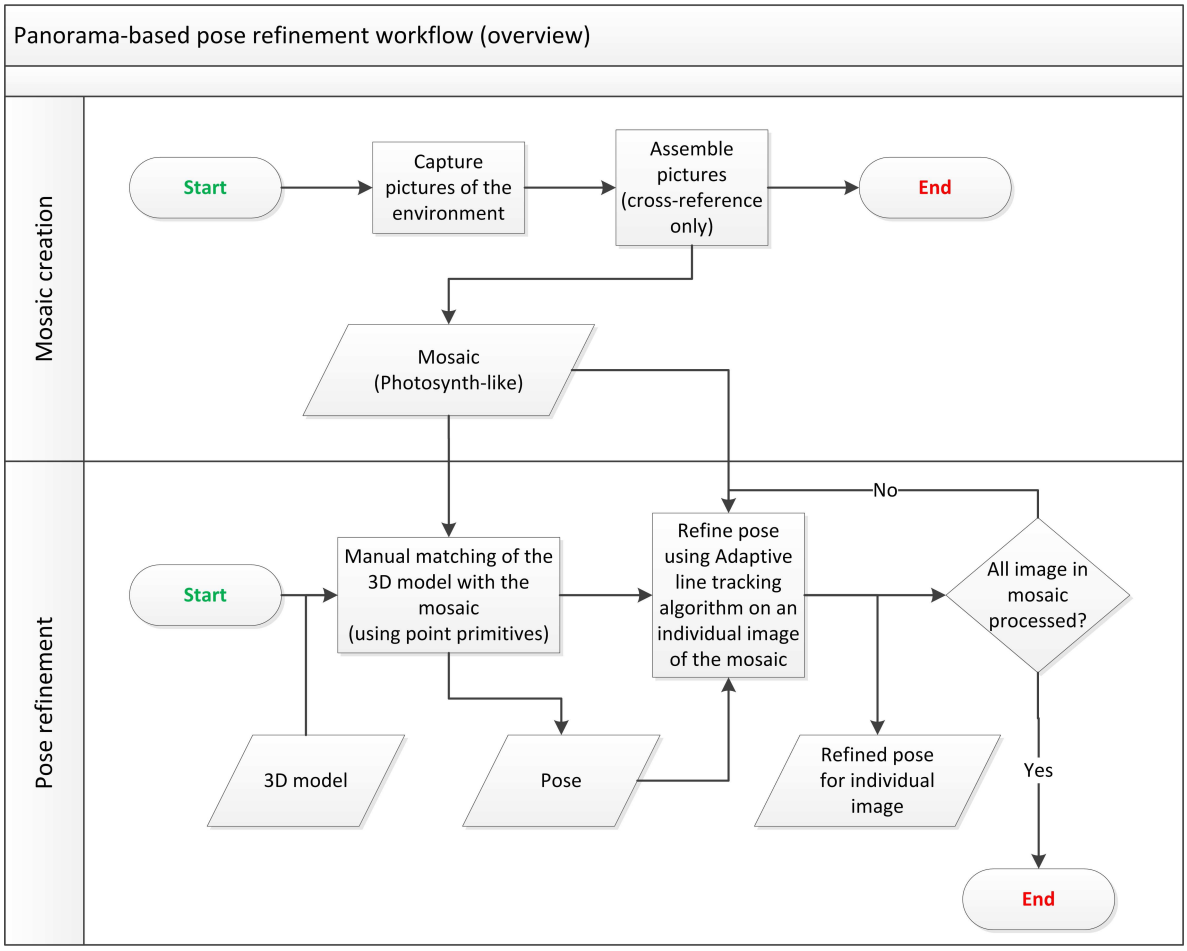

Figure 4.3 – Overview of the proposed workflow for the PBPR approach.

The investigation of the central problem of this approach, however, required to make some assumptions along with an overall simplification of the initial problem as stated in Section 1.2. The first simplification consists in putting aside the real-time tracking component to focus exclusively on the accurate augmentation of *static* images; the tracking component is no longer the component defining the accuracy of the overall system. The second simplification implies that, because this approach is based on *line* primitives, the focus of this specific experiment is shifted from *accuracy* to *coherence* (Section 1.2). Indeed, line tracking algorithms (or any computer vision-based algorithm) aim at matching an image and a corresponding model. In this context, a system can be accurate in relation to the real world only if *the model is accurate and representative of the reality*. If the 3D model is accurate, then a *coherent* system becomes an *accurate* one. Finally, as the experiment is focused on computer vision methods, a last assumption consists in assuming that the augmented structures are visible in the image (*i.e.* no underground infrastructure models can be overlaid).

The adaptative line tracking method used in the proposed PBPR approach is detailed by Wuest *et al.* [69] and is based on the method proposed by Vacchetti *et al.* [68]. This method is presented in details in the next section.

4.2 Adaptive Line Tracking with Multiple Hypotheses

Wuest *et al.* [69] propose a method to achieve AR by tracking, on each of the camera frames, the visible lines of a 3D CAD model. From an approximate registration of the model with the image, the line tracking algorithm ensures that the image-model registration is constantly updated. The application context is different: while Wuest *et al.* used the 3D model in order to track the virtual camera pose, it is used here to augment the panorama. Even if the end result is not the same the aim of both cases is identical: determine the virtual camera pose which will produce a point of view of the 3D model coherent with the reality (or representation of reality – using panoramic images in the context of this work). Therefore, for the application case of this project, only one iteration of the algorithm is required since the augmentation is done on a static image. This algorithm (adapted to this specific context) consists of the following steps:

1. Initialization and first pose estimation: the user provides a set of correspondences between the image and the 3D model to create an approximate virtual camera pose;

2. Line model generation: creation of a *Hidden Lines* 3D model from the approximate registration of the image and the 3D model, done through the manual initialization step, so that only edges that are potentially visible on the image are kept;

3. Identification of the multiple edge hypothesis:

   (a) Discretization of each line of the projected model into points along the line; the number of points should be proportional to the line length *on the image plane*;

   (b) For each line point, selection of the g gradient maxima along the line's normal, where g is the number of hypothesis[4];

4. Refining the registration using a numeric non-linear minimization algorithm like Levenberg-Marquardt [70] with Tukey's estimator [71] as the error function.

Each of these steps are explained in greater details in the next subsections.

4.2.1 Initialization and first pose estimation

The first step of this method consists in manually associating 3D points of the model with 2D positions in the image. In fact, any initialization method can be used for this step (*e.g.* POSIT [72] or the 3-points method proposed by Ameller *et al.* [73]), as long as an approximate camera pose is computed and the resulting pose is close enough to the pose of the camera when recording the image.

4.2.2 Line model generation

Most 3D CAD models of buildings consist of polygons [74]. As the pose refinement process is based on lines, the model needs to be converted to a wireframe representation [75]. This process simply consists in removing any polygon while retaining their edges; most CAD software provide this model representation. Subsequently, only the lines that are expected to be visible on the image on which the pose refinement will be applied must be considered. Therefore, some of the lines of the model need to be truncated or deleted so that only the expected visible parts are left. This can be achieved automatically

[4] An *hypothesis* is a point to be tested in the image as an edge point potentially corresponding to the projected line.

through standard raycasting [76] from the provided camera pose; any portion of an edge that cannot be reached by a virtual ray coming from the virtual camera's center because it is occluded by one or multiple polygon(s) of the model should be removed.

4.2.3 Identification of the multiple edge hypotheses

Once the model is in a wireframe representation and involves only the visible edges in the image (cf. Section 4.2.1), it can be projected onto the image using the first estimation of the virtual camera pose[5]. Then, the potential matches between the lines of the model and their counterparts in the image need to be found. This is achieved by first discretizing the lines and then finding the potential matches between the sampled points. These two steps are described below. The matching process yields multiple hypotheses (*i.e.* to each sampled point corresponds several potential matches). The advantage of using multiple hypotheses has been demonstrated by Vacchetti *et al.* [68]. The multiple match hypotheses are then used in the pose refining step.

4.2.3.1 Line discretization

First, each line of the model must be discretized into a set of sample points for which potential matches in the image will be searched. It is important to note that the number of sample points must be chosen relative to the projected (*i.e.* 2D) line length and not the actual length of the model line (3D). If this technique is used in a real-time application (thus is used for tracking), these points must also be distributed evenly along the projected line in order for the tracking to be robust. Again, in the context of real-time tracking, it is also important that the same sample points are used for each frame. To discretize the model lines, Wuest *et al.* [69] sample directly the 3D line model in a recursive manner (see Figure 4.4) and adapt the number of points to project on the image relatively to the projected line length at each frame. They then select the n first points of the 3D line to calculate the 2D/3D correspondences where

$$n = \frac{projected\,line\,length}{sample\,point\,density} \tag{4.1}$$

[5] See Section 4.2.3.1 for details.

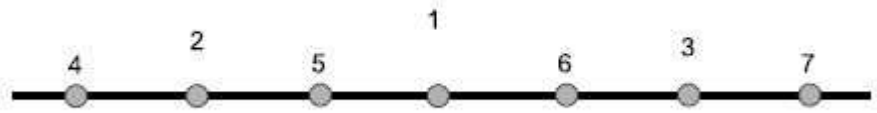

Figure 4.4 – Example of recursive line sampling. The numbers represent the order in which the line is discretized into points. For example, if 3 sample points are required, the points 1-2-3 will be selected. If 6 points are required, the points 1-2-3-4-5-6 will be selected. This ensure a relatively uniform distribution of the sample points along the line to be discretized.

where *sample point density* is a value determined manually[6].

Each of these sample point p is then projected on the image plane (resulting in p_i):

$$p_i = KMp \tag{4.2}$$

where K represents the matrix comprised of the camera's intrinsic parameters (focal length, principal point), M is the camera pose such that

$$M = \begin{bmatrix} R_{3\times3} & T_{3\times1} \end{bmatrix}_{4\times3} \tag{4.3}$$

where R is the rotation matrix and T is the translation vector[7].

4.2.3.2 Search for gradient maxima along line's normal

Once every visible lines of the model have been discretized in a number of sample points, potential matches for each of these points must be identified. Traditional line tracking methods usually search for the closest gradient maximum along the line's normal. In the context of Wuest *et al.* [69] approach, the search is expanded to multiple possible hypotheses for increased robustness. To be selected as a potential match (and thus to be added to the list of hypotheses), a point (i.e. a pixel in the image) needs to have a strong intensity gradient value; in addition, the gradient must be oriented in approximately the same direction as the normal of the line that is currently considered.

[6]The density should be function of the type of lines in the model and the processing power available. If the lines are straight, the density of sample points can be lowered. However, if the lines are splines, a higher density should provide a better accuracy. The density also has an impact on processing power since more points will be considered when analyzing the image. Therefore, when setting the sample point density, a tradeoff between real-time processing needs and accuracy requirements should be found.

[7]See Appendix "Pinhole Camera Model" for more details on the camera model and projection.

Such analysis is carried out using anisotropic Gaussian filters applied on the region of interest (*i.e.* along the line's normal).

Anisotropic Gaussian filters are simply non-isometric 2D filters that can be oriented. They can be used to smooth an image in a given direction to preserve the gradients in only one direction (*e.g.* if an anisotropic Gaussian filter oriented at 45° is applied to an image, only the edges displaying the same orientation will be preserved while the others will be smoothed out). For efficiency purposes, these filters can be pre-computed using various discrete angle values (belonging to the range $[0 - \pi]$) and the following general equation:

$$f(x,y) = Ae^{-(a(x-x_0)^2+2b(x-x_0)(y-y_0)+c(y-y_0)^2)} \tag{4.4}$$

where A is the maximum amplitude of the function and

$$a = \frac{\cos^2\theta}{2\sigma_x^2} + \frac{\sin^2\theta}{2\sigma_y^2} \tag{4.5}$$

$$b = -\frac{\sin 2\theta}{4\sigma_x^2} + \frac{\sin 2\theta}{4\sigma_y^2} \tag{4.6}$$

$$c = \frac{\sin^2\theta}{2\sigma_x^2} + \frac{\cos^2\theta}{2\sigma_y^2} \tag{4.7}$$

where θ is the rotation to apply to the Gaussian function and σ_x and σ_y are the standard deviation of the Gaussian distribution (*i.e.* level of smoothing) in the x and y dimension respectively.

Once the image region has been smoothed using the appropriate anisotropic Gaussian filter, the final step consists of applying a gradient maxima filter ($\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$) along the line's normal and to select the g highest values, where g is the number of hypotheses considered. An example of the anisotropic Gaussian filtering and hypothesis identification is presented in Figure 4.5.

4.2.4 Pose refinement using multiple hypotheses

When each line point has been associated with multiple hypotheses, the virtual camera pose of the model must be refined so it better matches the point of view represented by

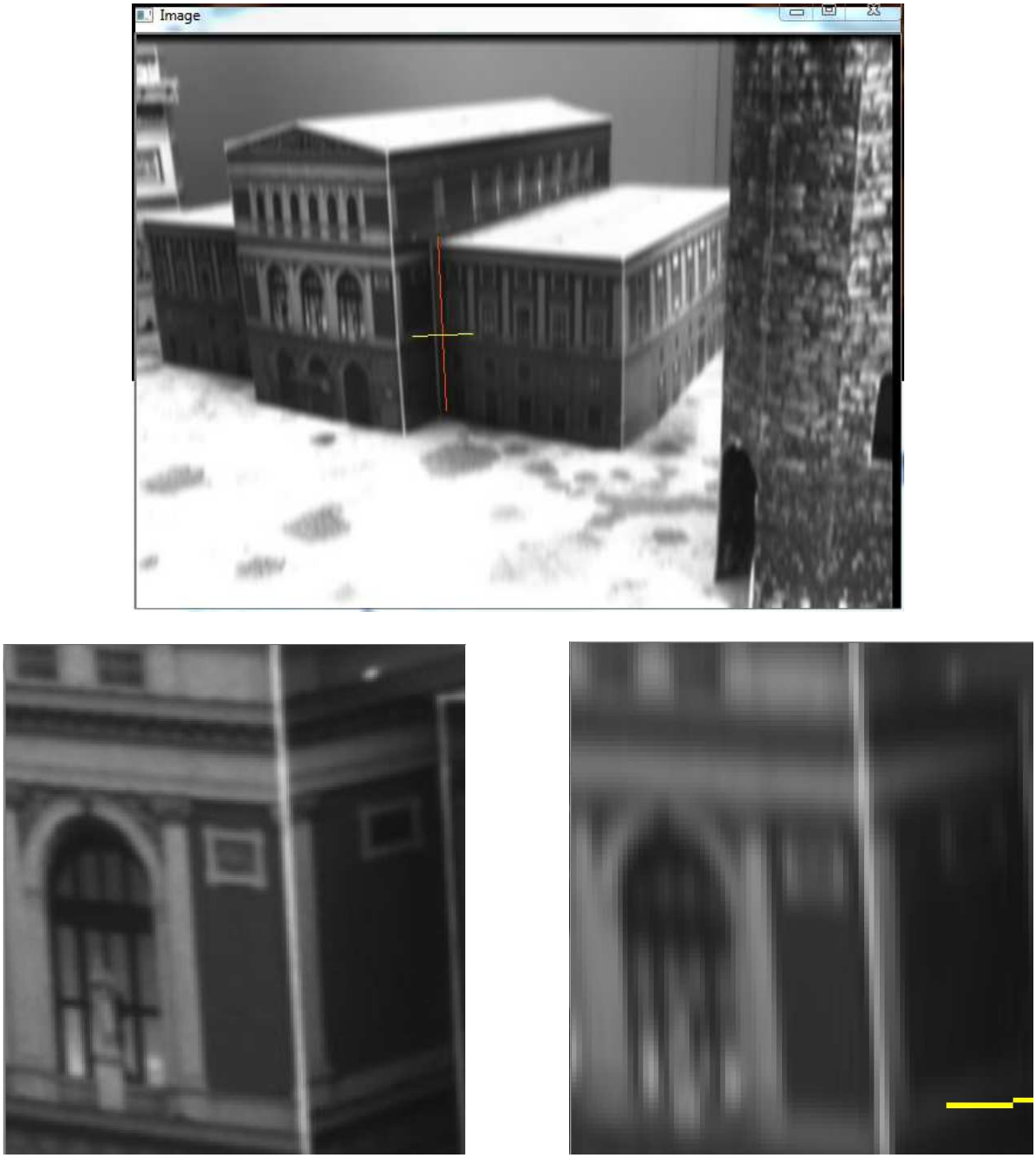

Figure 4.5 – (top) Example of an image filtered with an oriented anisotropic Gaussian filter and hypothesis identification for a specific line point. The current model line considered is displayed in red and the line's normal at the current position is displayed in yellow. It can be observed that gradient parallel to the line in red are preserved while gradients in other direction are degraded and blurred. (left) Zoomed-in crop of the original image and (right) the same crop on the image filtered with the anisotropic Gaussian filter. It can be observed that the gradients are only preserved in the vertical direction.

the image. The goal is to minimize the distance between the 3D model points projected on the 2D image and their 2D image correspondences. The minimization is done using the Levenberg-Marquardt algorithm which is a numeric and non-linear minimization method. The function to be minimized is the weighted distance between the current projection of the 3D model line point on the 2D image p_i and the hypothesis q_i along the line's normal $\vec{n}$. The function involve all the 3D model line points projected on the 2D image. Considering all the i line points of the 3D model for which there are j selected hypotheses, the function to be minimized is the following:

$$err = \sum_i \rho_{Tuk}\left(\min_j \Delta(p_i,q_{i,j})\right) \tag{4.8}$$

where ρ_{Tuk} is the Tukey estimator function, Δ is the distance function and $q_{i,j}$ is the j^{th} hypothesis corresponding to point p_i. The distance function can be computed as follow:

$$\Delta(p,q) = |(q-p)\cdot\vec{n}| \tag{4.9}$$

Tukey's estimator is used as both a weighting function and a outlier removal [68] (*i.e.* it is not necessary to apply a statistic-based outlier removal algorithm like RANSAC [58] when using Tukey's estimator). It can be expressed as:

$$\rho_{Tuk}(x) = \begin{cases} \frac{c^2}{6}\left[1-\left(1-\left(\frac{x}{c}\right)^2\right)^3\right] & \text{if} |x| \leq c \\ \frac{c^2}{6} & \text{if} |x| > c \end{cases} \tag{4.10}$$

where $c = k\cdot\sigma$, σ is the standard deviation of the estimation error and k is a constant representing the resulting rate of outliers.

4.3 Experiments, Results and Discussion

A prototype has been developed implementing the previously described algorithm. This section describes the experiments conducted using the developed prototype. First, details about the prototype implementation will be provided. Then the two data sets involved in the experiments will be described. Afterward, the contents of the experiments will be presented and explanations about the test rational and purpose will be provided. Finally, the results of these tests will be described and analyzed and the overall potential of the proposed approach for high-accuracy AR GIS will be discussed.

4.3.1 PBPR prototype implementation

For the manual initialization step, the method developed by Poirier [67], named *SphericalPnP* was used. This method is specifically designed to work on omnidirectional images, but is very well suited for standard images as well. It is summarized in the following three steps:

1. Manual selection of the 3D model-image correspondences by the user;
2. Computation of an approximate virtual camera pose from a random sample of correspondences (*e.g.* using RANSAC);
3. Refining of the pose using a non-linear minimization algorithm (*e.g.* using Levenberg-Marquardt)[8].

To convert the 3D model into a "visible edge only" model, Wuest *et al.* [69] tested the visibility of each point using an extension of OpenGL called GL_OCCLUSION_TEST_HP that directly uses the video card hardware to carry out the calculations. This has a definite advantage for real-time usage. In this experiment, Bentley MicroStation (CAD software) was used[9] to build a "visible edges"[10] line model using the camera pose provided by the manual initialization step.

The other steps of the algorithm described in Section 4.2 were implemented in a prototype developed in the C++ programming language in a Microsoft Windows environment.

4.3.2 Description of the data sets

The prototype was tested using two data sets with different levels of complexity: the first data set involved 3D models with simple geometry (planes) and a very low number of lines while the second data set consisted of 3D models where the level of detail

[8]This pose refinement is unrelated to the subsequent line-based pose refinement. In this step, an *approximate* pose for the whole mosaic is needed. The non-linear minimization is used to determine the best pose possible considering the manual point input of the user. When a rough pose has been determined, the line-based pose refinement algorithm will be applied to each single image that is part of the mosaic so that the pose fits each individual image with the best achievable precision.

[9]Mainly for simplicity purposes since it was easy and that there was no need for automation (one time process for each data set).

[10]This is the name of the Microstation function that was used.

Figure 4.6 – Musikverein model (textured and wireframe) from the "City of Sights" [8] data set used for the first tests of the "Adaptive Line Tracking" algorithm.

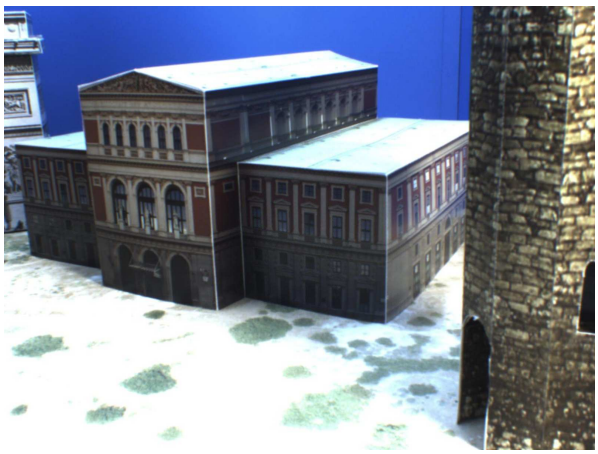

Figure 4.7 – Sample frame (undistorted) representing a paper reconstruction of the Musikverein model. A part of the building is slightly occluded on the right side of the frame.

and complexity was more aligned with engineering and architecture use cases. The 3D models of the second data set were built from mobile LiDAR point clouds.

The first data set consisted of two 3D models provided by the "City of Sights" data set [8]. The first model, representing the Musikverein in Vienna (Austria), consists of planes and a limited number of lines (Figure 4.6). The image used for the pose refinement process was extracted from a video of a scene containing a paper reconstruction of the 3D model. On this image, the model is partially occluded at the right side (Figure 4.7). The second model, representing the Berliner Dom, i.e. the Cathedral of Berlin (Germany), is slightly more complex as it presents domes of circular shapes (consisting of multiple straight lines) that might prove more difficult to augment (Figure 4.8). The image used for the pose refinement process was extracted from the same video file as the first model. In this image, the model is not represented entirely (some parts are cropped by the frame of the camera) and there is no occlusion.

Figure 4.8 – (left) Berlin Cathedral wireframe model from the "City of Sights" data set [8] and (right) sample frame (undistorted) representing a paper construction of the Berlin Cathedral model.

The second data set consisted of two 3D models of buildings of Laval University campus (Quebec City, Canada). These models were created by Mahmoud Alwa, M.Sc. [77, 78] using terrestrial LiDAR point clouds. CAD models based on point clouds are good candidates for a semi-automatic pose refinement process as they are accurate 3D representations of the real environment and therefore present several characteristics similar to those visible in the images of this physical world. The 3D models and a picture of the corresponding building on the campus are displayed in Figure 4.9.

4.3.3 Description of the experiments

The purpose of the PBPR approach (Section 4.1) is to determine the virtual camera pose for *each* individual image of the mosaic in order to "augment" it (in the augmented reality sense). Given the geo-engineering context of the proposed research work, the coherence of the augmentation is a key performance criteria. Thus, the experiments have been designed towards the assessment of such a criteria. They consist in assessing and comparing the coherence of the augmentation achieved using two different methods: the PBPR pose computation and a reference pose computation method. The latter is the SphericalPnP method proposed by Poirier [67]. This method allows for coherent augmentation as long as the manually input points are carefully selected. Also, because the PBPR pose computation is actually a *refinement*, it requires an initial virtual camera pose as an input. To emulate the fact that the initialization process realized on the mosaic will output locally flawed virtual camera poses for each image, the input of the PBPR method will be produced using SphericalPnP to which *flawed* manual registrations will be fed (see Point 2a). While it may seems strange to use SphericalPnP to produce the input virtual camera pose for the PBPR method *and* for its evaluation, it is

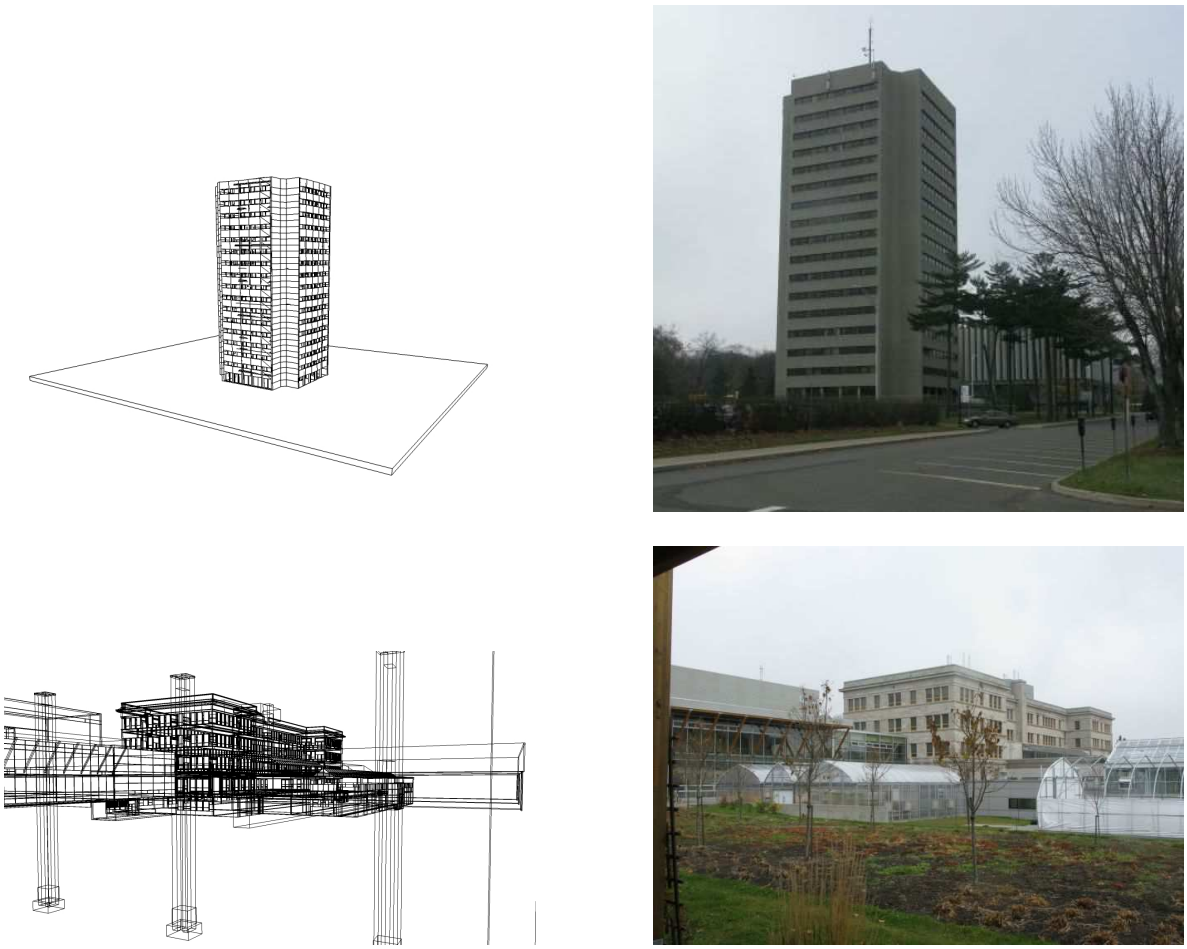

Figure 4.9 – CAD models built from LiDAR point clouds and associated images recorded on Laval University campus (Quebec city, Canada).

here completely justified since SphericalPnP is a method that outputs a virtual camera pose which precision and accuracy is dependent of the quality of the manually inputted associations between the 3D model and the image. Only qualitative visual comparisons of the two method performances have been carried out during these experiments, mainly due to time constraints.

The various steps involved in these experiments are described below:

1. Computation of the reference virtual camera pose
   (a) Manual registration of the 3D model with the image by carefully and precisely associating four[11] vertices of the model with their corresponding locations on the image.
   (b) Computation of the reference virtual camera pose.
2. Computation of the refined virtual camera pose using PBPR method
   (a) *Flawed* manual registration of the 3D model with the image by associating four vertices of the model with their corresponding locations on the image with an error of 5 to 20 pixels. These error values were estimated based on a qualitative study of a few data sets where an initialization was realized on a relatively local zone (approximately the size of a single camera frame) of the panorama and the error was visually observed on other zones of the panorama.
   (b) Computation of the flawed virtual camera pose.
   (c) Application of PBPR method to refine the virtual camera pose
3. Visual comparison of the image augmentations (i.e. superimposition of the 3D model on the image), using respectively:
   (a) the flawed virtual camera pose and the refine virtual camera pose;
   (b) the reference virtual camera pose and the refine virtual camera pose.

[11] Four is the minimal number of association required to compute a camera pose without ambiguities. However, since robustness and accuracy are the main concerns for this experiment, it would be more advised to consider more than the minimum number of correspondences between the 3D model and the image. However, since the selected models are relatively simple (at least in the City of Sights data set), four correspondences should be enough for the computed virtual camera pose to be correct.

4.3.4 Results and analysis

4.3.4.1 First data set: the City of Sight models

The first test involved the Musikverein model. Figure 4.10 shows the comparative result of the augmentation using, respectively the flawed virtual camera pose and the refined virtual camera pose computed by the PBPR method. Figure 4.11 shows the comparative result of the augmentation using, respectively the reference virtual camera pose and the the refined virtual camera pose. The differences are then displayed and highlighted in greater details in Figure 4.12.

In general, the generated virtual camera pose of both methods (*i.e.* SphericalPnP and PBPR) are very similar. Essentially two differences have been noticed. The first difference, highlighted by the red circle in Figure 4.12, tends to indicate that the virtual pose computed by the PBPR method shifted the model to the right. There is a discrepancy of several pixels between the model and the image on the upper right side of the image. This could be explained by the fact that one line of the Musikverein model is still being considered in the pose refinement even though it is occluded in the scene (Figure 4.13). The line could not have been removed in the hidden line model creation process as there is no way to know that this line will be occluded in the real-world scene. The PBPR method searched for potential matching lines in the image and found one (highlighted in green in Figure 4.13), erroneously shifting the model towards this false positive line. The second difference, highlighted by the pink circle in Figure 4.12, shows that it is the manual initialization that produced shifted results. This is probably due to the fact that no initialization point is located in this zone of the model[12] when the manual initialization was conveyed.

The second test involved the model of the Berliner Dom. Figure 4.14 shows the comparative result of the augmentation using, respectively the flawed virtual camera pose and the refined virtual camera pose. Figure 4.15 shows the comparative result of the augmentation using, respectively the reference virtual camera pose and the refined virtual camera pose.

Even though the pose refinement seem to have improved the virtual camera pose slightly, the results show both augmentations remain very incoherent. This is obviously an important problem. The problem is even more noticeable when comparing the

[12] In order to obtain the best virtual camera pose using a manual initialization method, the "anchor" points needs to be spread out as uniformly as possible on the 3D model. Local errors may occur if no points are located in a given area.

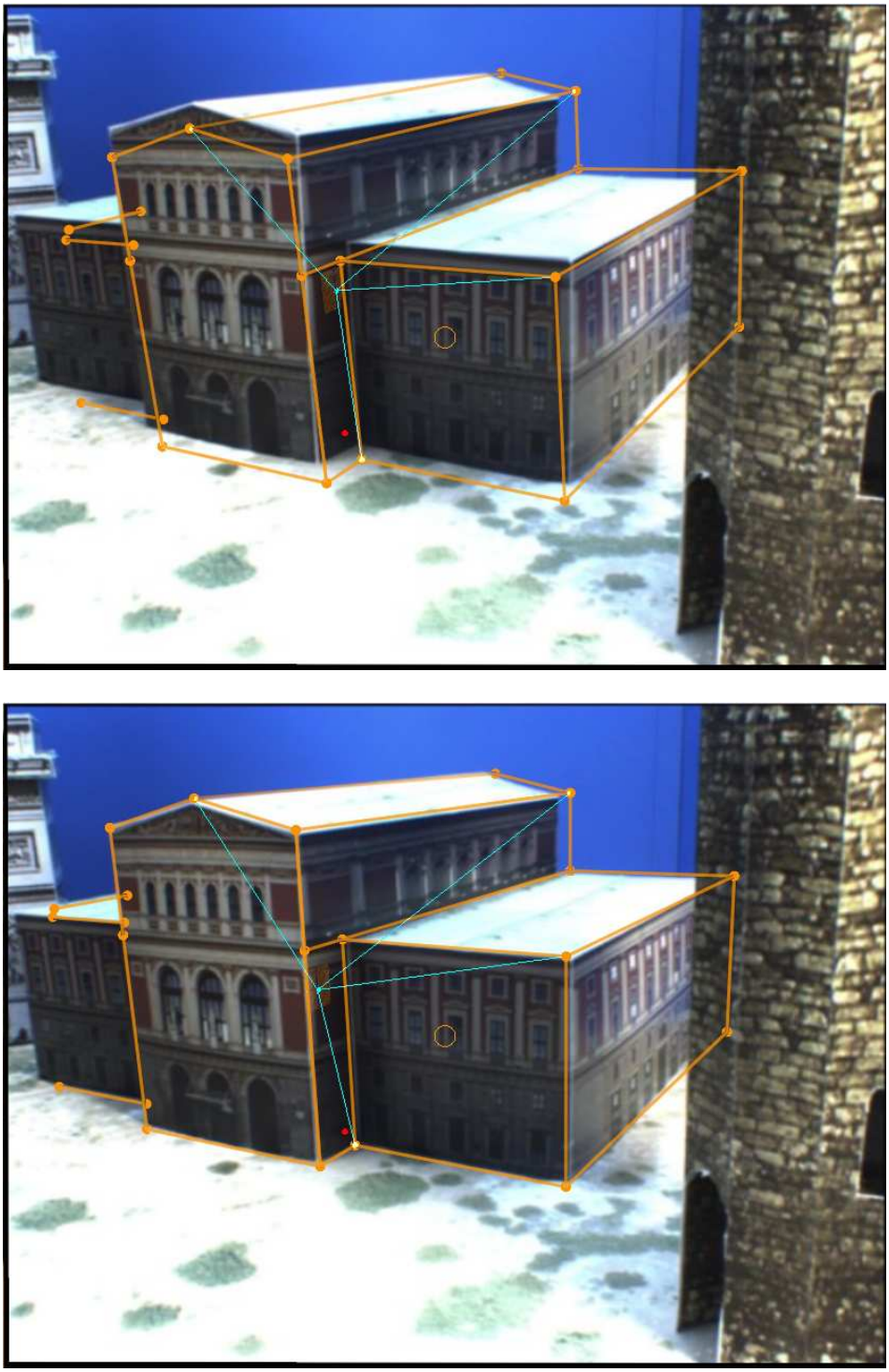

Figure 4.10 – Augmentation of the Musikverein model using two different virtual camera poses. Comparison between the (top) flawed virtual camera pose result and the (bottom) refined virtual camera pose result. The orange lines represents the 3D model in wireframe and the blue lines simply highlight the correspondences between the 3D model and the image.

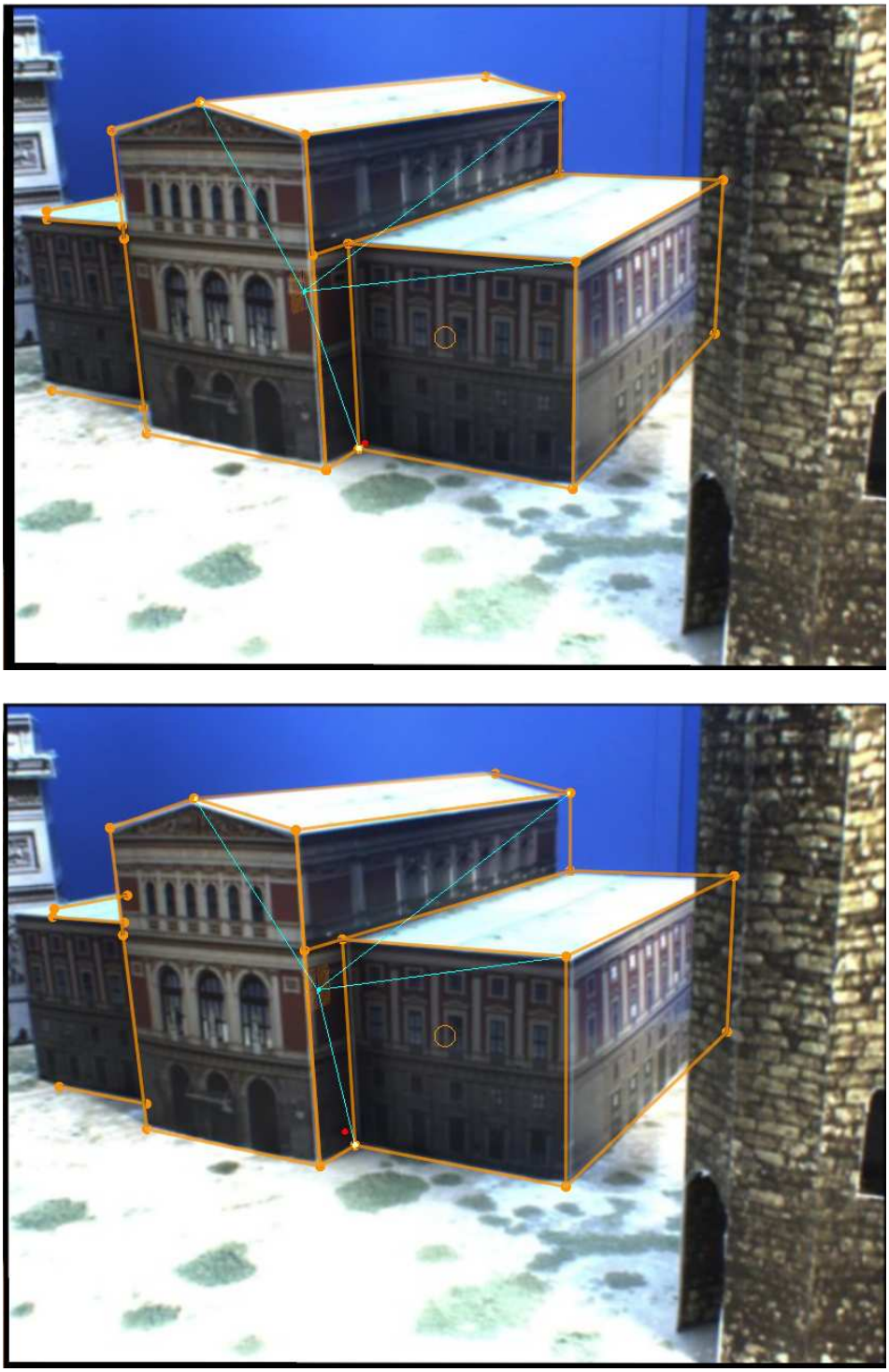

Figure 4.11 – Augmentation of the Musikverein model using two different virtual camera poses. Comparison between the (top) reference virtual camera pose result and the (bottom) refine virtual camera pose result. The orange lines represents the 3D model in wireframe and the blue lines simply highlight the correspondences between the 3D model and the image.

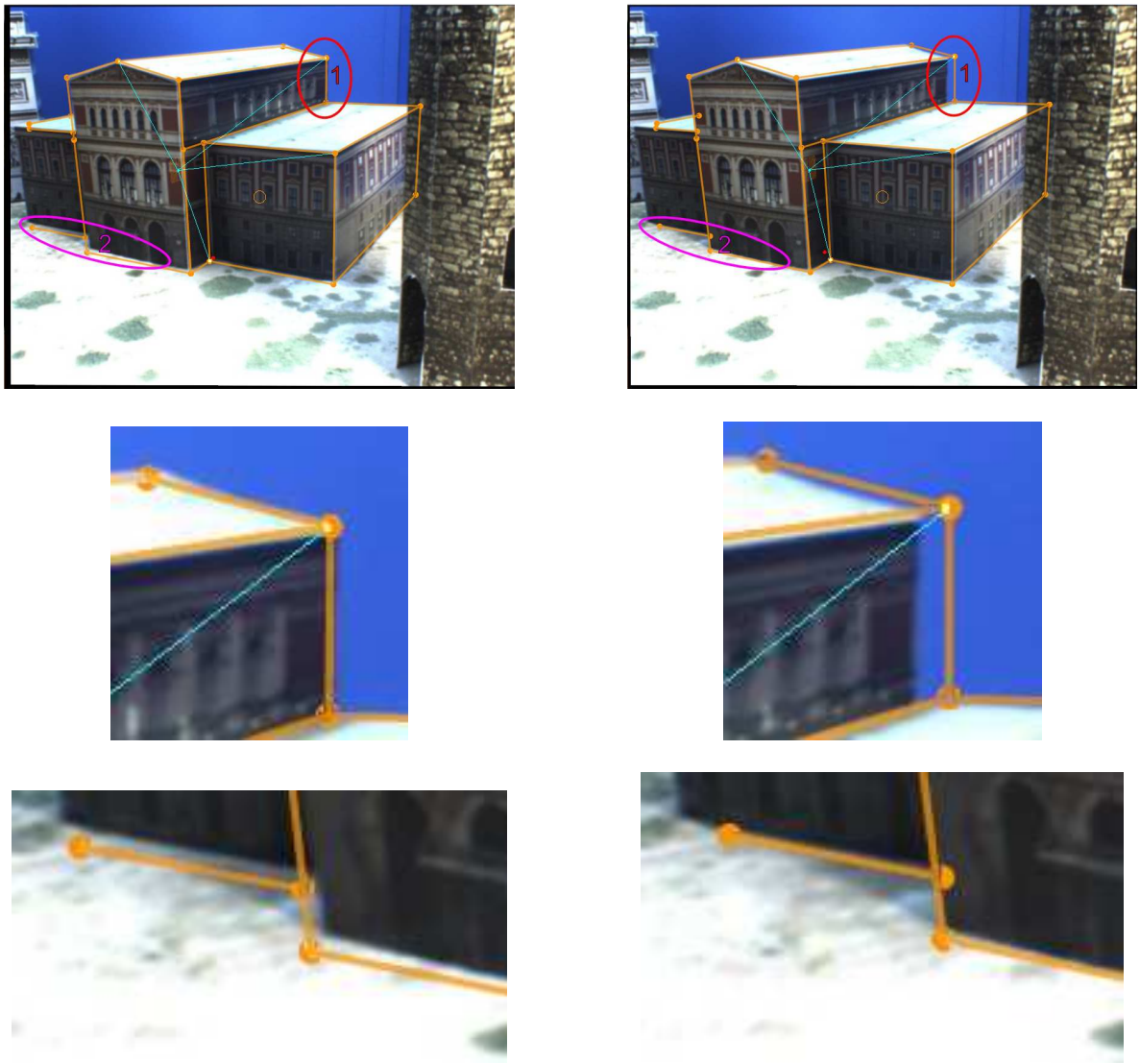

Figure 4.12 – Highlight and details of the differences between the augmentation shown in Figure 4.11. Comparison between the (left) augmentation based on the reference virtual camera pose and the (right) augmentation based on the refine virtual camera pose. The highlighted zone numbered “1” is depicted in the second line of the figure and the zone numbered “2” is depicted on the third line of the figure.

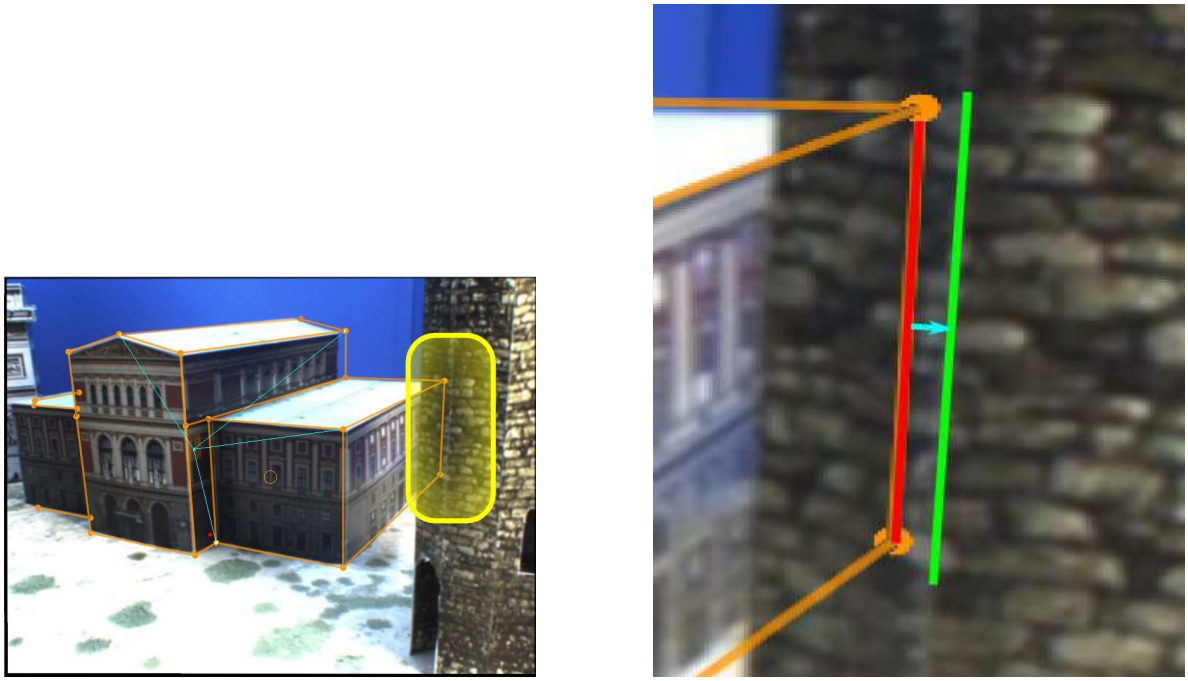

Figure 4.13 – Highlight of a potential cause of the shift of the model to the right: a line of the model (in red) that is occluded by the tower to the right may introduce errors since the PBPR tried to match this line with the nearest visible edge on the image (highlighted in green).

refined virtual camera pose to the reference virtual camera pose. The main hypothesis justifying these results is that many lines of the model (especially located in the different domes of the model) were kept in the hidden line version of the model even though these lines are not visible in the real-world image.

To confirm this hypothesis, the virtual camera poses were computed after manually removing the lines of the domes. Figure 4.16 shows the augmentation using respectively the flawed virtual camera pose and the refined virtual camera pose. Figure 4.17 shows the augmentation using respectively the reference virtual camera pose and the refined virtual camera pose. The differences are then displayed and highlighted in greater details in Figure 4.18.

There are again two main differences between the reference virtual camera pose result and the refined virtual camera pose result. However, they are very subtle and consists of shift of only few pixels (highlighted by the red and pink circles in Figure 4.18). In general, the reference virtual camera pose provide slightly better augmentations but they are visually very similar.

The results of the test with the Berliner Dom model without the domes' lines tend

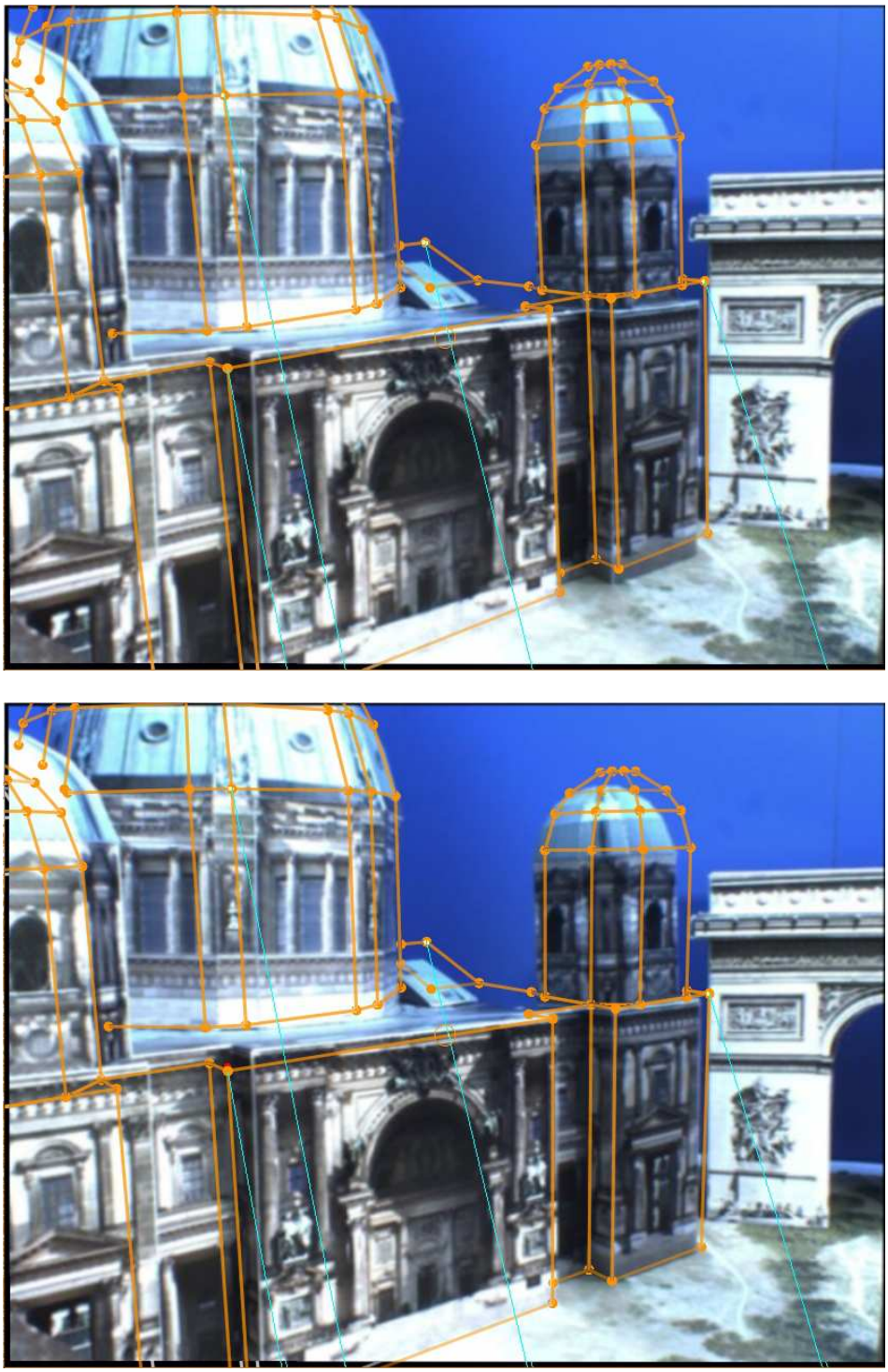

Figure 4.14 – Augmentation of the Berliner Dom model using two different virtual camera poses. Comparison between the (top) flawed virtual camera pose result and the (bottom) refined virtual camera pose result. The orange lines represents the 3D model in wireframe and the blue lines simply highlight the correspondences between the 3D model and the image.

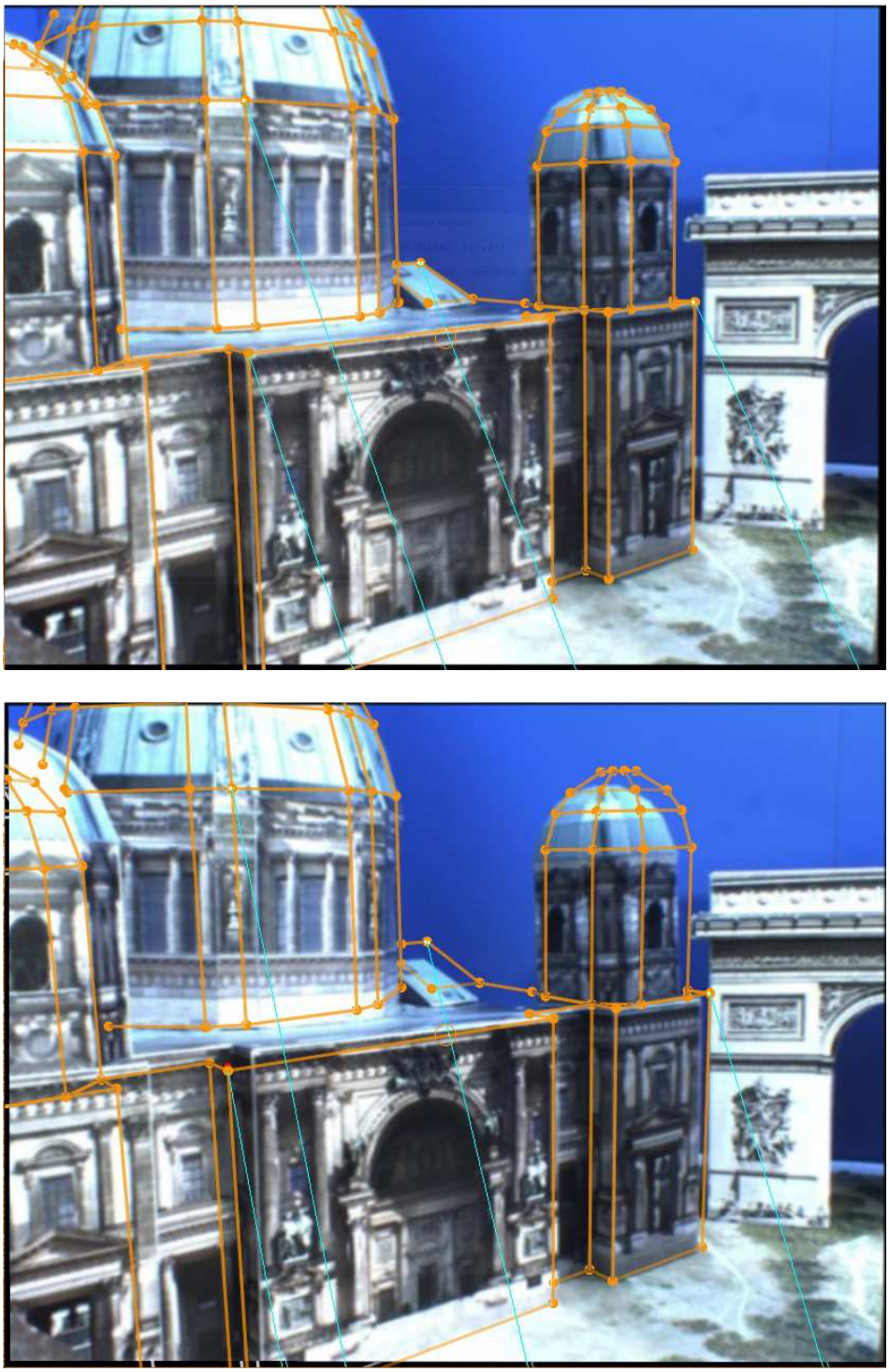

Figure 4.15 – Augmentation of the Berliner Dom model using two different virtual camera poses. Comparison between the (top) reference virtual camera pose result and the (bottom) refined virtual camera pose result. The orange lines represents the 3D model in wireframe and the blue lines simply highlight the correspondences between the 3D model and the image.

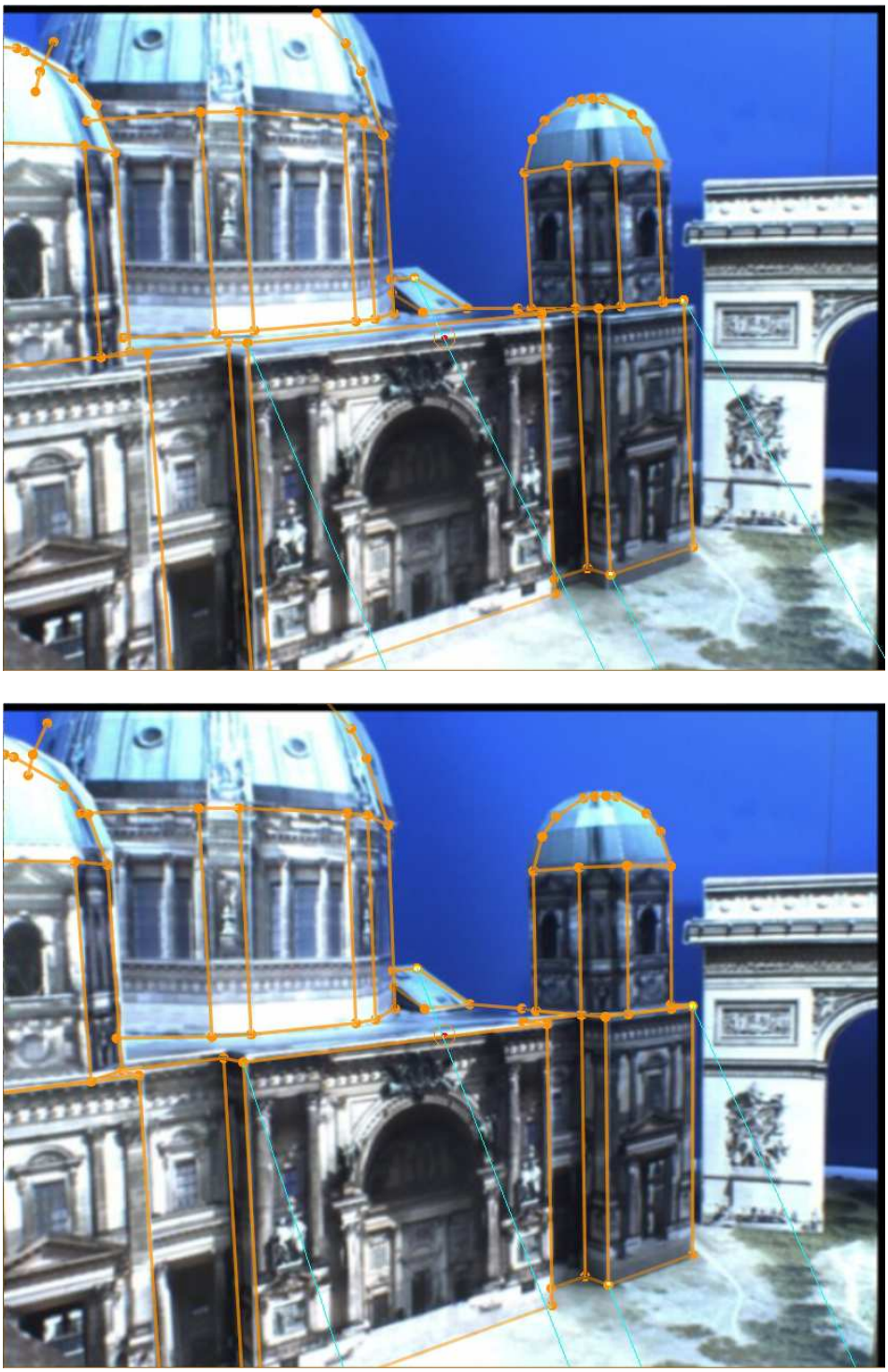

Figure 4.16 – Augmentation of the Berliner Dom model, without the dome lines, using two different virtual camera poses. Comparison between the (top) flawed virtual camera pose result and the (bottom) refined virtual camera pose result. The orange lines represents the 3D model in wireframe and the blue lines simply highlight the correspondences between the 3D model and the image.

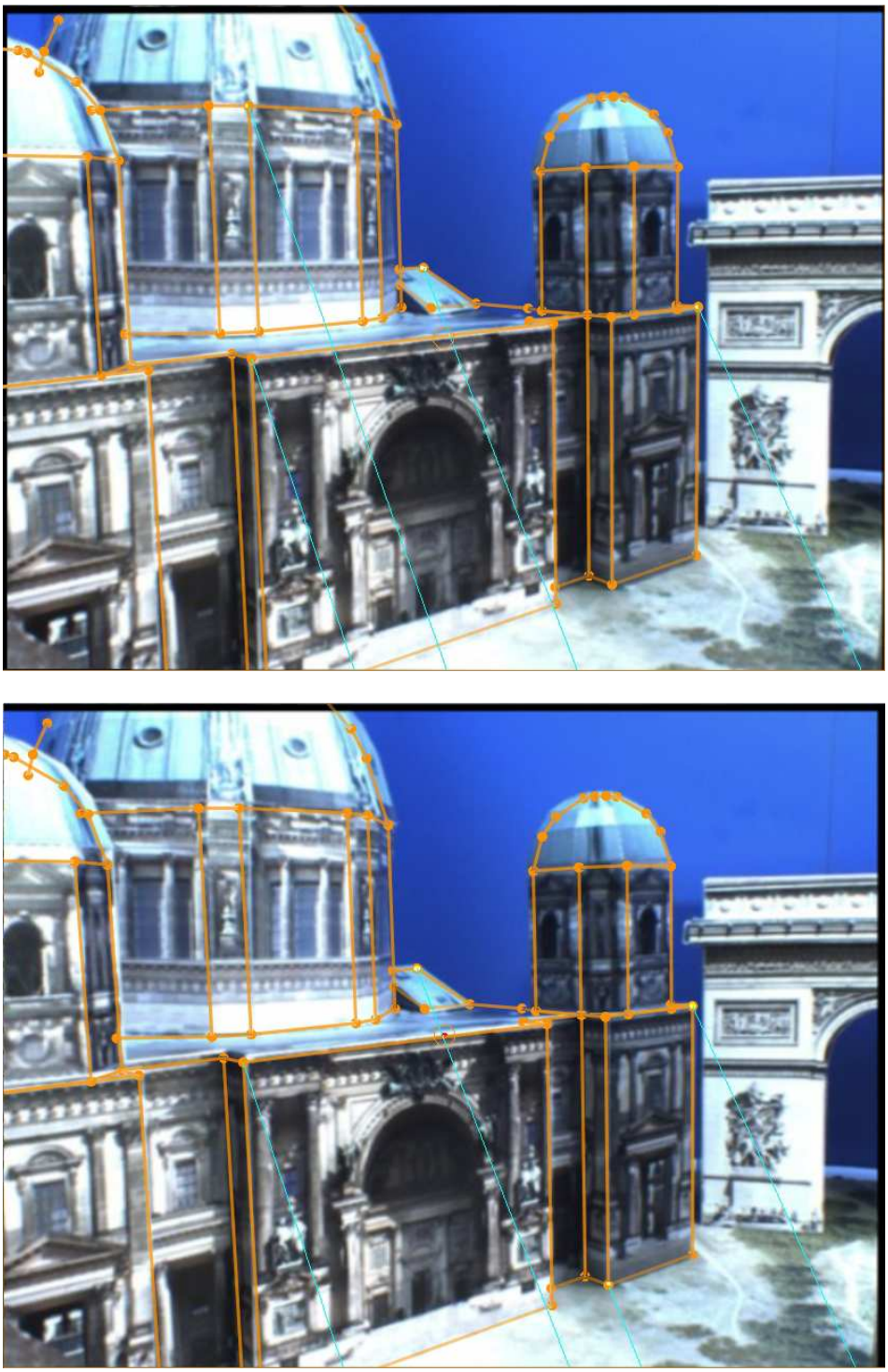

Figure 4.17 – Augmentation of the Berliner Dom model, without the dome lines, using two different virtual camera poses. Comparison between the (top) reference virtual camera pose result and the (bottom) refined virtual camera pose result. The orange lines represents the 3D model in wireframe and the blue lines simply highlight the correspondences between the 3D model and the image.

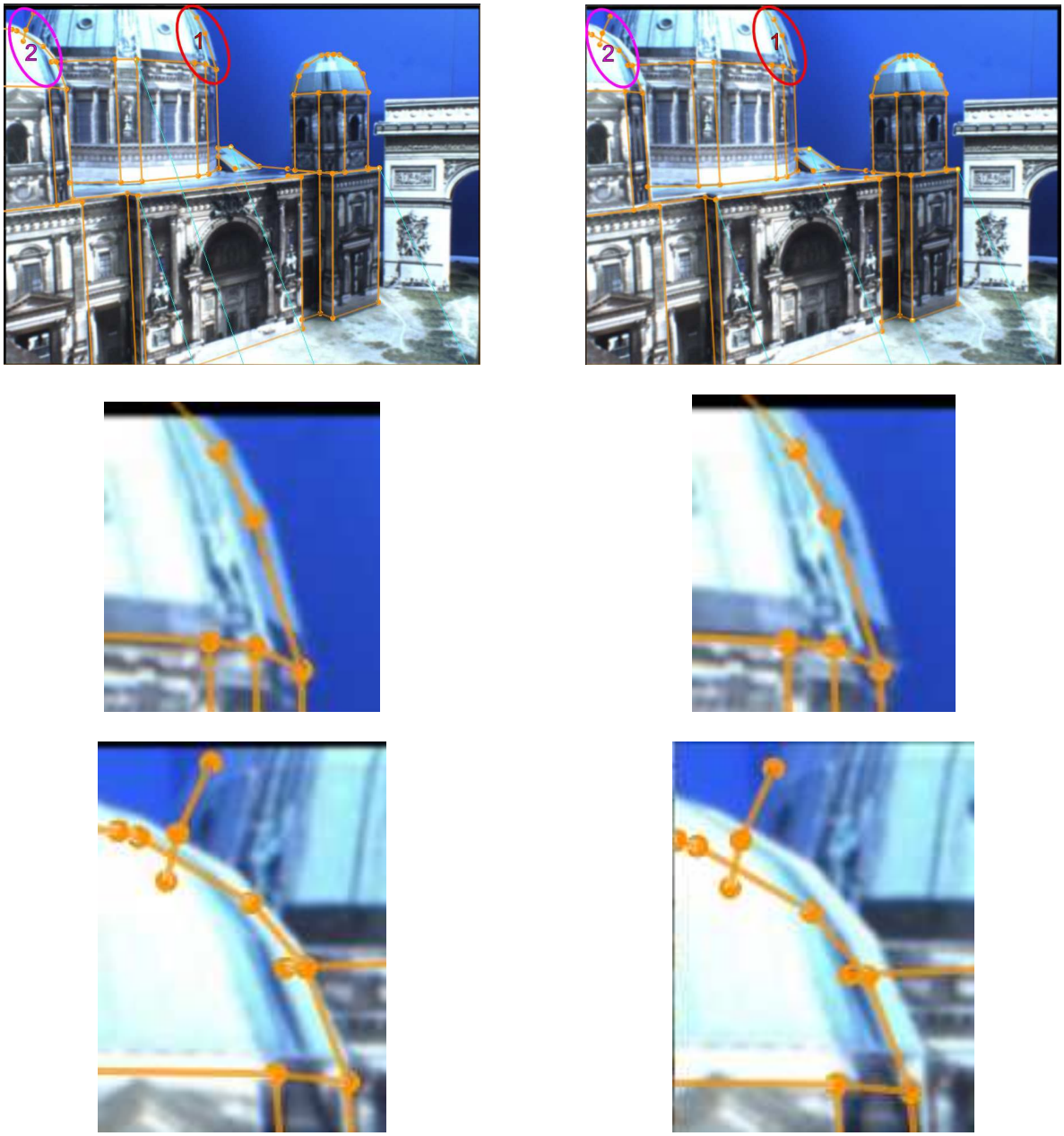

Figure 4.18 – Highlight and details of the differences between the augmentations shown in Figure 4.17. Comparison between the (left) augmentation based on the reference virtual camera pose and the (right) augmentation based on the refined virtual camera pose.

to confirm the hypothesis that lines of the model that have no visible correspondence in the real-world scene yield to incoherent augmentation when using the PBPR method. This is to be expected since the refinement method is based on *visible* lines. However, what is more of a problem is the seemingly very weak robustness of the PBPR method to this problem. The phenomenon seems to have impact on the resulting virtual camera pose even if only *one* line of the model does not find a match (or worst, a false positive match) in the real-world scene, as it has been observed with the Musikverein model. The method is supposedly relatively resilient to occlusion given a manual adjustment of the Tukey constant [69], but this resilience has not been observed with the Musikverein and BerlinerDom (*with* the domes' lines) models even when reducing the constant's value significantly[13].

In terms of *coherence*, even though the analysis is done qualitatively based on the visual aspect of the augmentation (how well it "fits" the real-world image), it must be noted that the virtual camera poses generated by the PBPR method do not show coherence levels that would meet the geo-engineering criteria presented in Section 2.1 (no visible offsets between the augmented structure and the overlaid model). This is true for both the Musikverein and Berliner Dom models. Indeed, the offsets highlighted in Figure 4.12 and 4.18 exceed few pixels.

4.3.4.2 Second data set: the LiDAR-based models

The next tests of these experiments involved the two models presented at Figure 4.9. However, no useful evaluation could be achieved due to several problems inherent to the virtual camera pose computation using the PBPR method. Indeed, either the augmentation based on the refined virtual camera pose was worse than the one based on the flawed virtual camera pose or no augmentation could be computed at all. Given the complexity of these models and considering the results obtained from the Berliner Dom model presented at the previous subsection, this outcome is not surprising. Indeed, these two models consists of a very high number of lines, many of which do not have *visible* correspondences in the real-world pictures. From these experiments, we can draw the conclusion that not only should the models be a representation of the reality as accurate as possible, but each of the model's lines should be visible in the image (or a more flexible approach should be devised).

[13] This is not to say that the method is not robust to occlusions. Additional tests would be required to check if an optimal value of the Tukey constant would solve the occlusion issues observed in the two previous 3D models.

4.4 Synthesis and conclusion

In this chapter, a panorama-based pose refinement method has been proposed as an alternative to the accuracy bottleneck caused by the panorama creation process. The proposed PBPR approach consists in applying the adaptative line tracking approach developed by Wuest *et al.* [69]. Because this approach associates visible lines of the model with the edges in the real-world image, if an augmentation is to be accurate, the 3D model used must be as representative of the reality as possible. The method was tested on two different data sets: one consisted of two simple 3D models composed of few lines and the other consisted of two real-world models built from terrestrial LiDAR point clouds. The experiments consisted of visually comparing the augmentation results using virtual camera pose computed respectively by a flawed method (i.e. inaccurate manual registration of the 3D model with the image), a reference method (i.e. manual registration of the 3D model with the image using the SphericalPnP method proposed by Poirier) and the PBPR method.

The two simple 3D models yielded similar augmentation results for the three camera pose computation methods. However, when slightly more complex models were used, discrepancies among the augmentation results could clearly be identified. Even the reference method showed augmentation incoherence. The fact that some model lines did not have visible line correspondences in the image was identified as the source of the coherence errors. This issue was even more noticeable when the PBPR method was applied to the real-world models; no coherent virtual camera pose could be computed. This means that, while generating the hidden line and wireframe version of the model, the visibility of all the lines of the model need to be checked in the real-world image. In the engineering or architecture contexts of this research project, such issue requires an operator to manually identify and remove the lines that are not visible in the physical world, task which is viewpoint-dependent. This would completely defeat the purpose of the PBPR semi-automatic approach.

These experiments also highlighted that even in a simplified context, achieving pixel accuracy on a virtual camera pose is not easy. Many factors can quickly degrade the registration between the 3D model and the image when relying on computer vision techniques (*e.g.* occlusions).

Ideally, the PBPR approach would yield augmentations more coherent than manual initialization through an automatic and pixel-precise match of the 3D model and image. However, in order to achieve such coherent augmentations, the method should be able to correctly differentiate between information in the model that can be used and infor-

mation that is irrelevant. Another approach would be to create a representation of the model which can directly be compared with the real-world image (*e.g.* by extracting only the contours of the model).

These experiments demonstrated that the PBPR approach, while providing results relatively close to a careful manual initialization in simple cases, does not provide virtual camera pose close enough to the real camera pose of the image to be compliant with the accuracy requirements of geo-engineering and architecture fields. Moreover, the PBPR method failed to scale for increases in the level-of-detail and complexity of the 3D models. However, it is possible that other primitives, instead of lines alone, could provide more coherent results and increased robustness.

Chapter 5

Conclusions and Future Work

This M.Sc. work has focused on tracking approaches compliant with accurate mobile AR GIS systems dedicated to architecture and geo-engineering applications. The main approach considered was the use of omnidirectional panorama images along with different computer vision techniques. Two different solutions were explored: one using online orientation tracking (OT) and the other using image-based pose refinement. Both approaches demonstrated strengths and weaknesses and their exploration has provided insights about the potential of using panorama images in high-accuracy AR contexts. This chapter discusses these matters and also proposes suggestions for future work.

5.1 Conclusions

5.1.1 Review of the project objectives and proposed research

The main objective of this research project was stated as such: to *define the accuracy requirements of an AR mobile GIS in a geo-engineering context and list the technical approach(es) that are most suited to achieve this accuracy level*. It has been broken down into three sub-objectives:

- Sub-objective 1: specifying the accuracy requirements
- Sub-objective 2: determining suitable tracking approaches for mobile AR GIS dedicated to geo-engineering

- Sub-objective 3: validating the selected potential tracking approaches.

The first sub-objective was met through a literature review. The accuracy requirements of a wide spectrum of tasks related to these fields were determined. These accuracy requirements varied as a function of the task at hand and the distance between the user and the location of the object being augmented. To simplify this metric, it was determined that, if the user's position remains constant, an accuracy of approximately 0.129° for most tasks conveyed at relatively close ranges (0-18 m) and 0.035° for tasks conveyed at longer ranges (+18 m) would be required to enable decision support for architects and engineers.

The second sub-objective was then met by conducting a review of the different available tracking solutions. Most of these solutions would neither fulfill the previously identified accuracy requirements nor be appropriate for the engineering and architecture context of this research. However, one of them seemed to present high potential to achieve the required level of accuracy: panorama-based tracking. Two approaches, relying on panorama-based tracking were then selected for investigation: Orientation Tracker (OT) and Panorama-Based Pose Refinement (PBPR).

Finally, the last sub-objective consisted in developing prototypes and propose tests in order to validate the selected approaches to assess their potential for mobile AR GIS solutions. The first approach to be validated was the Orientation Tracker system. This system was designed to provide the orientation of a live video frame by comparing the image contents with a corresponding panorama image, acting as a reference system. While OT was able to provide coherent orientation measurements, it failed to achieve the required level of accuracy previously determined. This first experiment with OT has enabled the identification of different accuracy limitations of a panorama-based tracking system (they will be described below). In order to address one of these limitation (considered as an accuracy bottleneck), a second experiment, PBPR, was proposed. It considered an alternative approach: tackling the accuracy problem of image-based augmentation by focusing on static augmentations instead of the online tracking component. This approach is worthwhile because high-accuracy augmentations, even if static, can still be very useful to any user as it allows decision support. Again, the approach achieved positive results for a very simple case: the virtual camera pose returned by the PBPR method was comparable to a virtual camera pose obtained through very careful manual initialization. However, the method simply failed when applied to more complex cases. Conclusions specific to each approach are detailed in the next two subsections.

5.1.1.1 Lessons learned from the OT experiment

While developing and testing Orientation Tracker, it was found that using omnidirectional panorama images as the reference of a vision-based 3 DOF tracking solution presents some problems:

- The panorama construction process itself is flawed: automatic panorama creation usually provides visually seamless results, but the proposed research underlined that the process also involves local distortions which make some portions of the image a more or less an accurate representation of reality by several pixels.
- There are no studies to confirm that the different image feature detectors used by the SURF algorithm, which are used in order to match the different images together, are able to reliably locate a *corresponding* pair of features with a sub-pixel precision;
- The different libraries implementing the SURF image descriptor offers varying levels of performance and accuracy. This means that the chosen implementation has an non-negligible effect on the overall system and that not every implementation might be "true" to the original SURF algorithm;

The first issue is a strong hurdle if panorama images are to be used in high accuracy fields. Indeed, the accuracy of the tracking system and, therefore, of the overall augmentation is limited by the accuracy of the representation of the reality; a system cannot be more accurate than the accuracy of the reference system it is based on.

OT experiment also highlighted a more general problem when assessing the accuracy of a tracking technology. System evaluation, while often considered a simple validation operation, is, in the context of accuracy assessment, far from being trivial. In this work, it had important implications. Because the system under study (OT) is required to *measure* the reality, accurate[1] measurements of the reality are required to validate the values provided by OT. However, each system that measures the reality is prone to error. This problem was highlighted in Sections 3.3.1 and 3.3.2. Initially, MEMS-type hardware was judged as appropriate to validate OT. However, after an extensive testing phase, it was concluded that the hardware was easily disturbed by magnetic interferences and miscalibrations. Using a pan-tilt camera (with a high-accuracy motor) then seemed a reasonable alternative, only to observe some problems regarding the hardware construction (Section 3.4.3).

[1] The measurements need to be at least as accurate as the evaluated system's required accuracy.

The ground-truth problem is usually encountered when a field is relatively young and that no benchmark for testing a given type of technology has been established. This is a problem because the comparison between systems developed by different groups becomes difficult without a given standard. Also, high accuracy validation requires high-end hardware, not always available to all research groups. The difficulties encountered during the OT experiment are definitely an argument in favor of the development of a standard benchmark.

Toward the end of the experiments of this project, such a benchmark was published by Gruber *et al.* [8]. Their work provides a complete set of models that can be easily built using cardboard paper or that can be used in their virtual form (CAD models). They also provides videos with corresponding ground-truth poses for every frame, computed with different methods (computer vision and mechanical arm).

5.1.1.2 Lessons learned from the PBPR experiment

One of the most important problem related to the use of panorama images for high accuracy AR, identified in the OT experiment, is the distortions introduced in the panorama during the creation process. This presents a bottleneck in terms of the achievable accuracy of panorama-based augmentations. The next experiment, PBPR, was an attempt to work around this problem by only processing the *individual* images composing the omnidirectional panorama. Starting from a manually determined and very approximate virtual camera pose provided by the user, the PBPR consisted of refining this virtual camera pose for *each and every* individual image so that they would each display the most coherent augmentation possible. The pose refinement was done using line primitives.

The conveyed evaluation of the PBPR method being qualitative instead of quantitative, it is not possible to state that a pixel accuracy augmentation is achievable through this method. However, the test conducted on a simple geometry model provided an augmentation very similar to one resulting of a carefully executed manual initialization.

Additional tests were conducted on more complex models. They demonstrated that the method did not scale very well. Indeed, the PBPR method seemed very sensible to edges that were present in the models while being either poorly or not visible at all in the scene (*e.g.* due to occlusions or poorly lit portions of the frame). The presence of these edges with no correspondence shifted the computed virtual camera pose significantly. This hypothesis was confirmed by comparing the resulting virtual

camera pose of the PBPR method applied on two slightly different versions of the same model (with and without edges with no visual correspondence in the scene). This problem was highlighted even more when applying the PBPR method on real use case (*i.e.* very complex) models; no coherent results were obtained.

This leads to the conclusion that lines alone may not be the best primitive for virtual camera pose refinement in very complex contexts. At first, line primitives were selected as it was the most straightforward choice considering the architecture and geo-engineering application fields. Indeed, 3D models are usually composed of polygons (which can be simplified to lines). Image descriptors could also be used to refine the augmentation, but the problem lies in the availability of textured 3D models. Moreover, matching the texture of a 3D model with corresponding real-world images may cause some problems: textures are sensitive to lighting conditions, season changes or any changes of the physical environment that might have occurred between the 3D model creation and the image acquisition. There are solutions for the online creation of 3D textured models (ProFORMA [79] and Vi3Dim[2]) but they can only be used on small scale objects or require that a checkerboard pattern is present under the object. These limitations prevent their use when large scale objects (*e.g.* building) need to be modeled. Another solution could consist in using the same set of images to create both the textures of the model and the panorama. In this specific case, no pose refinement method would be required as mapping textures from an image to a 3D model essentially corresponds to creating a registration between the model and the image respective spaces (initialization). It is to be noted, though, that the creation of such textured model would be much more time consuming for the user than providing an approximate virtual camera pose through manual initialization. Such use of the same data set for both the panorama and the textured model might prevent any large scale use of this alternative.

5.1.2 Research contributions

In this M.Sc. work, two main approaches, based on panorama images, were considered. Each one of these approaches required that some simplifications of the original problem was made: OT reduced the number of tracked DOF from 6 to 3 while the PBPR approach dropped the tracking component completely in order to focus on the accuracy of static augmentations. These compromises, although very important, do not reduce the relevance of this research's contributions. While these simplifications would have generated prototypes that would have been incompatible, in part, with the definition

[2] http://www.vi3dim.com

of AR, they allowed the problems, already very complex, to be brought down to a acceptable level of complexity for the length of a M. Sc. project. Also, as the central objective of this project was to evaluate the *potential* of approaches for high accuracy AR in architecture and geo-engineering contexts, conclusions drawn from simplified contexts can be generalized to their more complex counterparts.

The two main experiments described in this research have highlighted the fact that panorama images in conjunction with computer vision techniques show potential for future AR GIS applications but might not provide a suitable solution that is both simple and very accurate. Namely, this research has highlighted, through prototype development and testing, that the computer vision components induce errors at different levels of the process preventing the accuracy objective to be met. Also, the selected hardware, mainly the camera's resolution was one of the main limiting factor. Instead, it might be worth looking into sensor fusion and advanced algorithms (*e.g.* SLAM) that would pair hardware solutions with computer vision techniques to increase the overall accuracy of the system. It would also be worth considering higher quality hardware components; a greater camera resolution (in pair with more capable processing unit) should be on the top of the list.

The two experiments (OT and PBPR) suggest that a relatively good accuracy can be produced using only panorama images and computer vision techniques, even if the said accuracy does not fulfill the requirements of decision support tasks of architects and geo-engineers. The proposed approaches could be of use for applications that would have slightly lower accuracy requirements ($\geq 0.5°$).

Finally, this research has confirmed the overall difficulty in aiming for mobile AR that is able to support engineering-level decisions while using off-the-shelf and relatively low-quality hardware. Even in simplified use cases (*i.e.* 3 DOF), the many components involved in the virtual camera pose determination algorithms can quickly accumulate error. The importance of carefully choosing each of these components has been highlighted throughout the experiments. In answer to the main objective of this research, that is identifying the technical approach(es) that are most suited to achieve the accuracy level required in architecture and geo-engineering contexts, the experiments realized cannot direct towards a specific tracking solution recommendation. Someone interested in high accuracy mobile AR should first evaluate appropriate tracking methods in controlled environments and with ideal hardware specifications. In the case where no real-time, highly accurate and mobile tracking solution would be available, alternatives to a real-time AR system might be worth considering. This was a key aspect of the PBPR approach: if the accuracy is more crucial than real-time interaction, considering static augmentations could still enable highly reliable insights for professional users as long

as an automatic or semi-automatic method to accurately anchor a 3D model with an image is available.

5.2 Future Work

This study focused on the use of panorama images in the context of mobile AR GIS dedicated to architecture and geo-engineering applications. Two main experiments have been realized: OT and PBPR. These experiments explore possible avenues to achieve pixel-accurate augmentation of the real world. Based on these experiments, several conclusions were drawn about both the feasibility and the potential of each type of approach. These experiments also sparked some ideas that were left untouched due to time constraints and might prove to be worth investigating in the future. Some of these ideas are presented here.

5.2.1 Future work related to OT

The experiments with OT highlighted some potential problems that need to be tackled before being able to achieve highly accurate augmentations using an online, computer vision only orientation tracking system.

The first one is the point of interest detector accuracy. The current benchmark used to evaluate the point of interest detectors does not include a metric to assess explicitly the accuracy of the detector. Instead, the main metric is the *repeatability* (or ϵ-repeatability, see Section 3.4.2) which consists of calculating the proportion of points for which a correspondence is found within a distance of ± 1.5 pixel of the ground truth. The ϵ value involved in the experiments described in this thesis was too high to assess the performance of a particular point of interest detector technology in the context of the proposed application fields. Repeatability tests should be run with a smaller threshold ($\epsilon \leq 1.0$ pixel). A point of interest detector that would show poor performances to this test should be avoided.

Another important problem underlined by the proposed research is the flawed panorama construction process. In fact, this problem was important enough that it justified the second experiment of this study (PBPR). PBPR presented a specific way to work around this problem. However, a different solution could consist in capturing a panorama using a single capture shot instead of merging a few dozens individual images together.

Figure 5.1 – Example of hardware that can be used to capture a one-shot 180° panorama image: (left) fisheye lens with a 8–15 mm focal length and (right) a standard camera aimed at a hyperbolic mirror. (*source: (left): http://goo.gl/Dh7Fp (right): http://goo.gl/exeD7*)

Different hardware solutions could enable this process: using a fisheye lens or by taking a picture of the environment indirectly via a hyperbolic mirror (Figure 5.1). If such hardware are to be used in conjunction with any computer vision algorithm, it would be important to make sure that the captured frames can be undistorted accurately. Indeed, hyperbolic mirrors and fish-eye lenses present stronger distortions than standard camera lenses. For the images to be used for computer vision algorithms, it is necessary to have high confidence in the camera calibration process. This might prove to be more crucial in the case of the hyperbolic mirror since the mirror might present more imperfections than a high quality fish-eye lens. Moreover, the optical axes of both the camera and the hyperbolic mirror has to be carefully aligned. Finally, when capturing the panoramic image through the hyperbolic mirror, it is required to account for the distortion introduced by both the lens of the camera and the mirror and therefore require a different calibration technique (see the work of Mei and Rives [80]).

5.2.2 Future work related to the PBPR method

In Section 4.2, a pose refinement algorithm was proposed relying on image line extraction and matching with corresponding 3D model features. However, while being a good approach for line tracking for simple models composed only of visually observable edges, line tracking had trouble dealing with more complex cases. To get around this problem, image descriptor technology such as SURF or SIFT could be used. However, as stated in Section 5.1.1.2, textured 3D models are not the norm in the architecture and geo-engineering industry. Instead, it should be possible to use a rendering engine to create a high quality monochrome render of the 3D CAD model and to use SIFT or any image comparison features that works on *intensity gradient*. The problem would therefore amount to comparing two images.

The overall idea and workflow for this approach is very similar to the one proposed for the PBPR method in Section 4.1. The user first provides approximate model-image correspondences and a first virtual camera pose is computed. Then, this pose is used to produce a high-quality monochromatic render of the model (a simple example illustrated at Figure 5.2). These two images can now be compared with a high-level image descriptor algorithm (*e.g.* SIFT) that works on intensity gradient or with a contour extraction algorithm. Once a set of 2D correspondences is obtained, and because the position and orientation from which the render was produced are exactly known (*i.e.* the virtual camera pose), a new set of 3D model-image correspondences can be fed to the Levenberg-Marquardt algorithm in order to refine the pose. However, it must be noted that all the conclusions drawn from the OT experiment (Section 5.1.1.1) apply to this type of approach as it uses the same basic components (*i.e.* computer vision).

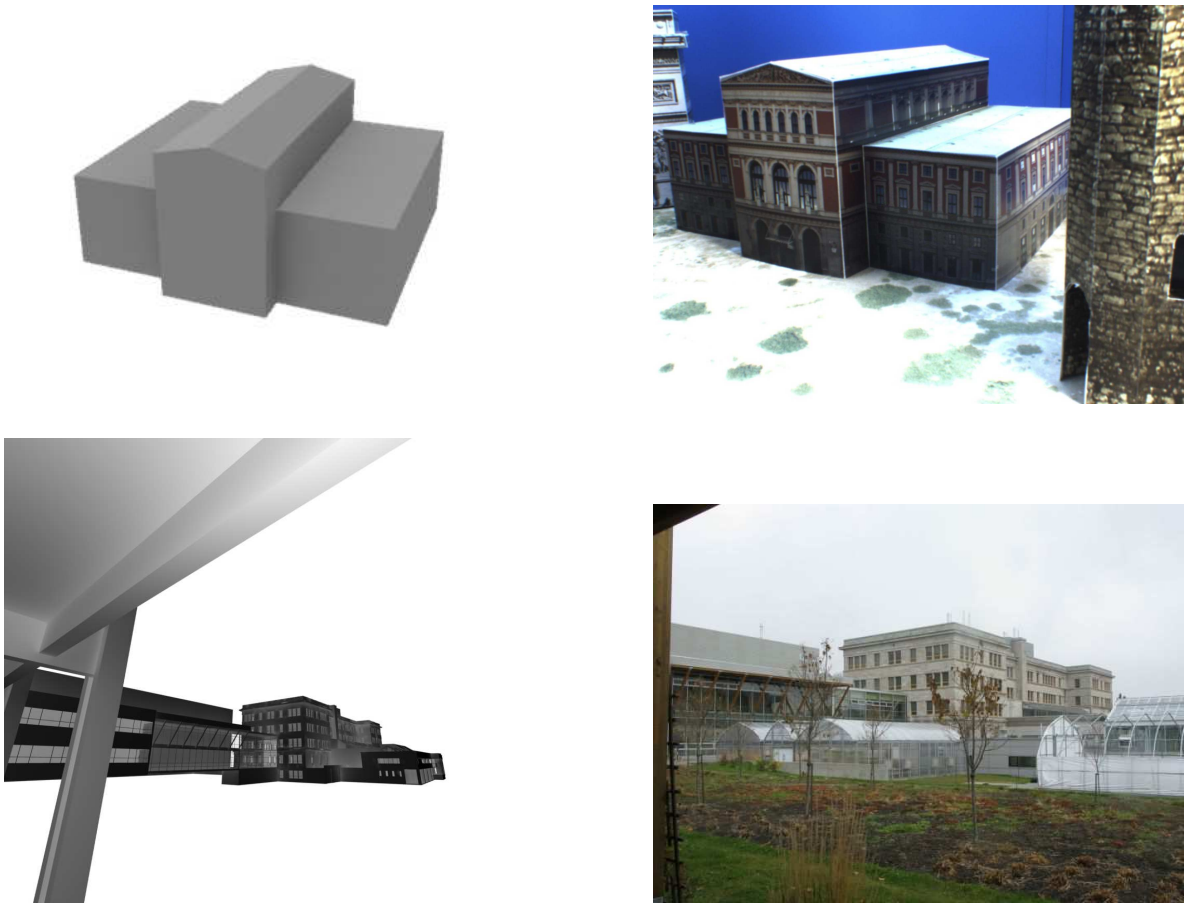

Figure 5.2 – Example of a 3D CAD model render. Produced from the approximate pose computed from the manual initialization realized on the image on the right.

Bibliography

[1] D. H. Shin, W. Jung, and P. S. Dunston, "Camera constraint on multi-range calibration of augmented reality systems for construction sites," *Virtual and Augmented Reality in Design and Construction*, vol. 13, no. Special Issue, pp. 521–535, 2008.

[2] G. Jee and M. Petovello, "What is GNSS repeater-based positioning and how is it different from using pseudolites?" *Inside GNSS*, no. July/August 2009, pp. 18–21, 2009.

[3] D. Schmalstieg and D. Wagner, "Experiences with handheld augmented reality," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, Japan, Nov. 2007, pp. 1–13.

[4] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *2nd IEEE and ACM International Workshop on Augmented Reality, 1999. (IWAR '99) Proceedings.* IEEE, 1999, pp. 85–94.

[5] E. W. Weisstein, "Cylindrical projection," *MathWorld–A Wolfram Web Resource*, [Online; accessed 28-July-2011]. Available: http://mathworld.wolfram.com/CylindricalProjection.html

[6] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of interest point detectors," *Int. J. Comput. Vision*, vol. 37, no. 2, pp. 151–172, 2000.

[7] C. Evans, "Notes on the OpenSURF library," University of Bristol, Tech. Rep. CSTR-09-001, Jan. 2009.

[8] L. Gruber, S. Gauglitz, J. Ventura, S. Zollmann, M. Huber, M. Schlegel, G. Klinker, D. Schmalstieg, and T. Höllerer, "The city of sights: Designing an augmented reality stage set," in *Proc. Nineth IEEE International Symposium on Mixed and Augmented Reality (ISMAR'10)*, Seoul, Korea, Oct. 2010.

[9] N. Cowan, "The magical number 4 in Short-Term memory: A reconsideration of mental storage capacity," *Behavioral and Brain Sciences*, vol. 24, no. 01, pp. 87–114, 2001.

[10] C. Rinner, M. Raubal, and B. Spigel, "User interface design for Location-Based decision services," in *13th International Conference on GeoInformatics*, 2005.

[11] M. Rohs, J. Schöning, M. Raubal, G. Essl, and A. Krüger, "Map navigation with mobile devices: virtual versus physical movement with and without visual context," in *Proceedings of the 9th international conference on Multimodal interfaces*, ser. ICMI '07. Nagoya, Aichi, Japan: ACM, 2007, pp. 146–153, ACM ID: 1322219.

[12] F. Zhou, H. B. Duh, and M. Billinghurst, "Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR," in *7th IEEE/ACM International Symposium on Mixed and Augmented Reality, 2008. ISMAR 2008.* IEEE, Sep. 2008, pp. 193–202.

[13] D. H. Shin and P. S. Dunston, "Evaluation of augmented reality in steel column inspection," *Automation in Construction*, vol. 18, no. 2, pp. 118–129, Mar. 2009.

[14] Wikipedia, "Agile software development — wikipedia, the free encyclopedia," 2011, [Online; accessed 9-August-2011]. Available: http://en.wikipedia.org/w/index.php?title=Agile_software_development&oldid=443605182

[15] "precision, n. (2.d. statistics)," *The Oxford English Dictionary*, Mar. 2010, [Online; accessed 31-October-2011]. Available: http://dictionary.oed.com/cgi/entry/50186553

[16] "accuracte, a. (3)," *The Oxford English Dictionary*, [Online; accessed 17-November-2010]. Available: http://dictionary.oed.com/cgi/entry/50001458

[17] S. Amos, *Newnes Dictionary of electronics*. Oxford Boston: Newnes, 1999.

[18] Wikipedia, "Camera resectioning — wikipedia, the free encyclopedia," 2011, [Online; accessed 31-October-2011]. Available: http://en.wikipedia.org/w/index.php?title=Camera_resectioning&oldid=455201872

[19] R. Azuma, "A survey of augmented reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355—385, Aug. 1997.

[20] W. Broll, I. Lindt, I. Herbst, J. Ohlenburg, A. K. Braun, and R. Wetzel, "Toward Next-Gen mobile AR games," *IEEE Computer Graphics and Applications*, vol. 28, no. 4, pp. 40–48, Aug. 2008.

[21] G. Klein and D. Murray, "Improving the agility of Keyframe-Based SLAM," in *Proc. 10th European Conference on Computer Vision (ECCV'08)*, Marseille, Oct. 2008, pp. 802–815.

[22] "trilateration, n." *The Oxford English Dictionary*, Nov. 2011, [Online; accessed 11-November-2011]. Available: http://www.oed.com/view/Entry/206064?redirectedFrom=trilateration

[23] Wikipedia, "Pseudorange — wikipedia, the free encyclopedia," 2011, [Online; accessed 11-November-2011]. Available: http://en.wikipedia.org/w/index.php?title=Pseudorange&oldid=445339239

[24] M. Arikawa, S. Konomi, and K. Ohnishi, "Navitime: Supporting pedestrian navigation in the real world," *Pervasive Computing, IEEE*, vol. 6, no. 3, pp. 21–29, 2007.

[25] R. Sabatini and G. Palmerini, "Differential global positioning system (DGPS) for flight testing," NATO Research and Technology Organisation, Tech. Rep. RTO-AG-160 AC/323(SCI-135)TP/189 Volume 21, Oct. 2008.

[26] G. Lachapelle, "Pedestrian navigation with high sensitivity GPS receivers and MEMS," *Personal and Ubiquitous Computing*, vol. 11, no. 6, pp. 481–488, 2007.

[27] D. S. Chiu and K. P. O'Keeke, "DGPS + UWB," *GPS World*, no. March 2009, 2009.

[28] V. Renaudin, B. Merminod, and M. Kasser, "Optimal data fusion for pedestrian navigation based on UWB and MEMS," in *PLANS 2008*, 2008.

[29] S. Ingram, D. Harmer, and M. Quinlan, "UltraWideBand indoor positioning systems and their use in emergencies," in *Position Location and Navigation Symposium, 2004. PLANS 2004*, 2004, pp. 706–715.

[30] A. Dempster, "QZSS's indoor messaging system - GNSS friend or foe?" *Inside GNSS*, no. January/February 2009, pp. 37–40, Jan. 2009.

[31] B. Forssell, "Indoor message system evaluated," *GPS World*, no. April 2009, Apr. 2009.

[32] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[33] "jitter, n. (2)," *The Oxford English Dictionary*, Apr. 2010, [Online; accessed 09-April-2012]. Available: http://oxforddictionaries.com/definition/jitter

[34] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, vol. 1, May 2006, pp. 430–443.

[35] D. G. Lowe, "Distinctive image features from Scale-Invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[36] H. Bay, A. Ess, T. Tuytelaars, and L. Vangool, "Speeded-Up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[37] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[38] J. Xiao, H. Cheng, H. Sawhney, C. Rao, M. Isnardi, and S. Corporation, "Bilateral filtering-based optical flow estimation with occlusion detection," in *In ECCV, volume I*, 2006, pp. 211–224.

[39] I. M. Rekleitis, "Steerable filters and cepstral analysis for optical flow calculation from a single blurred image," in *In Vision Interface*, 1996, pp. 159–166.

[40] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 376–382, Jun. 1991.

[41] 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*, vol. 26, no. 6, pp. 1052–1067, 2007.

[42] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, Nov. 2007.

[43] R. O. Castle, D. J. Gawley, G. Klein, and D. W. Murray, "Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras," in *Proc. International Conference on Robotics and Automation, Rome, Italy, April 10-14, 2007*, 2007, pp. 4102–4107.

[44] S. Riisgaard and M. Rufus Blas, "SLAM for dummies : A tutorial approach to simultaneous localization and mapping," [Online; accessed 25-February-2009]. Available: http://ocw.mit.edu/NR/rdonlyres/Aeronautics-and-Astronautics/16-412JSpring-2005/9D8D

[45] E. Eade and T. Drummond, "Scalable monocular SLAM," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*. IEEE Computer Society, 2006, pp. 469–476.

[46] "panorama, n. (1)," *The Oxford English Dictionary*, Jun. 2011, [Online; accessed 25-July-2011]. Available: http://www.oed.com/view/Entry/136923?redirectedFrom=panorama

[47] T. Langlotz, D. Wagner, A. Mulloni, and D. Schmalstieg, "Online creation of panoramic augmented reality annotations on mobile phones," *IEEE Pervasive Computing*, 2010.

[48] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)*, Orlando, Oct. 2009.

[49] Z. Zhu, G. Xu, E. Riseman, and A. Hanson, "Fast generation of dynamic and multi-resolution 360 deg; panorama from video sequences," in *Multimedia Computing and Systems, 1999. IEEE International Conference on*, vol. 1, Jul. 1999, pp. 400 –406 vol.1.

[50] E. W. Weisstein, "Cylindrical equidistant projection," *MathWorld–A Wolfram Web Resource*, [Online; accessed 28-July-2011]. Available: http://mathworld.wolfram.com/CylindricalEquidistantProjection.html

[51] J. Park and J. Park, "3DOF tracking accuracy improvement for outdoor augmented reality," in *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, Oct. 2010, pp. 263 –264.

[52] S. DiVerdi, J. Wither, and T. Hollerer, "Envisor: Online environment map construction for mixed reality," in *2008 IEEE Virtual Reality Conference*, Reno, NV, USA, Mar. 2008, pp. 19–26.

[53] S. Kim, C. Coffin, and T. Höllerer, "Relocalization using virtual keyframes for online environment map construction," in *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*. Kyoto, Japan: ACM, 2009, pp. 127–134.

[54] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg, "Real-time panoramic mapping and tracking on mobile phones," Waltham, MA, Mar. 2010, pp. 211–218.

[55] S. E. Chen, "Quicktime VR: an image-based approach to virtual environment navigation," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995, pp. 29–38.

[56] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Comput. Vision*, vol. 74, pp. 59–73, Aug. 2007.

[57] M. Brown and D. G. Lowe, "Recognising panoramas," in *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ser. ICCV '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 1218–.

[58] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[59] S. Belongie, "Rodrigues' rotation formula," *MathWorld–A Wolfram Web Resource*, [Online; accessed 22-November-2011]. Available: http://mathworld.wolfram.com/RodriguesRotationFormula.html

[60] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal Of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[61] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal Of Basic Engineering*, vol. 83, no. 1, pp. 95–108, 1961.

[62] D. Gossow, P. Decker, and D. Paulus, "An evaluation of open source SURF computer vision implementations," Active Vision Group, University of Koblenz-Landau, Koblenz, Germany, Tech. Rep., 2009.

[63] Mikolajczyk, Tuytelaars, Schmid, Zisserman, Matas, Schaffalitzky, Kadir, and Gool, "A comparison of affine region detectors," *Int. J. Comput. Vision*, vol. 65, no. 1-2, pp. 43–72, 2005.

[64] C. Harris and M. Stephens, "A combined corner and edge detection," in *Proceedings of The Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[65] F. Remondino, "Detectors and descriptors for photogrammetric applications," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 3, pp. 49–54, 2006.

[66] V. Bilodeau, "Panorama creation: Construction of accurate 360 degrees panoramas," Bentley Systems Inc, Quebec, Qc, Canada, Tech. Rep., Aug. 2010.

[67] S. Poirier, "Estimation de pose omnidirectionnelle dans un contexte de realité augmentée," Quebec, Qc, Canada, 2011.

[68] L. Vacchetti, V. Lepetit, and P. Fua, "Combining edge and texture information for Real-Time accurate 3D camera tracking," in *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 2004, pp. 48–57.

[69] H. Wuest, F. Vial, and D. Strieker, "Adaptive line tracking with multiple hypotheses for augmented reality," in *Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium on*, 2005, pp. 62–69.

[70] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly Journal of Applied Mathmatics*, vol. II, no. 2, pp. 164–168, 1944.

[71] E. W. Weisstein, "Tukey's biweight," *MathWorld–A Wolfram Web Resource*, [Online; accessed 25-November-2011]. Available: http://mathworld.wolfram.com/TukeysBiweight.html

[72] D. F. Dementhon and L. S. Davis, "Model-based object pose in 25 lines of code," *International Journal of Computer Vision*, vol. 15, pp. 123–141, 1995, 10.1007/BF01450852.

[73] M.-a. Ameller, B. Triggs, and L. Quan, "Camera pose revisited: New linear algorithms," in *In 14eme Congres Francophone de Reconnaissance des Formes et Intelligence Artificielle*, 2002, p. 2002.

[74] Wikipedia, "3D modeling — wikipedia, the free encyclopedia," 2012, [Online; accessed 10-January-2012]. Available: http://en.wikipedia.org/w/index.php?title=3D_modeling&oldid=469144987

[75] "Wire-frame, a. (2)," *The Oxford English Dictionary*, Dec. 2011, [Online; accessed 10-January-2012]. Available: http://www.oed.com/view/Entry/272645?redirectedFrom=wireframe

[76] Wikipedia, "Ray casting — wikipedia, the free encyclopedia," 2011, [Online; accessed 10-January-2012]. Available: http://en.wikipedia.org/w/index.php?title=Ray_casting&oldid=460841099

[77] "GéoÉduc3D : la géomatique au service des jeux vidéos et de l'apprentissage - accueil," [Online; accessed 29-November-2011]. Available: http://geoeduc3d.scg.ulaval.ca/

[78] "GEOIDE," [Online; accessed 29-November-2011]. Available: http://www.geoide.ulaval.ca/

[79] Q. Pan, G. Reitmayr, and T. Drummond, "ProFORMA: probabilistic feature-based on-line rapid model acquisition," in *Proc. 20th British Machine Vision Conference (BMVC)*, London, Sep. 2009.

[80] C. Mei and P. Rives, "Single view point omnidirectional camera calibration from planar grids," in *IEEE International Conference on Robotics and Automation*, Apr. 2007.

Appendix A

Pinhole Camera Model

The most commonly used camera model to represent the projection of a scene on an image plane is the *pinhole camera model* (also referred as the *perspective camera model*). Most of the information presented here is adapted (or directly ported) from [32].

Images are, by definition, mostly in two dimensions. That means that a point in the real world needs to be converted somehow to find its place on the image's projection plane. Between these two coordinate systems, there is the camera world that refers to 3D points from the point-of-view of the camera. This set of transformations includes the *intrinsic* and *extrinsic* camera parameters. The first allows to project a given point expressed in the camera's coordinate system onto the projection plane while the second allows to define the position and orientation of the camera according to the reference coordinate system (often referred as the world's coordinate system).

Intrinsic parameters

Trucco and Verri [32] defines the intrinsic parameters as follow:

> Intrinsic parameters can be defined as the set of parameters needed to characterize the optical, geometric, and digital characteristics of the viewing camera. For a pinhole camera, there are three sets of intrinsic parameters, specifying respectively:
>
> - the perspective projection, for which the only parameter is the focal length, f;

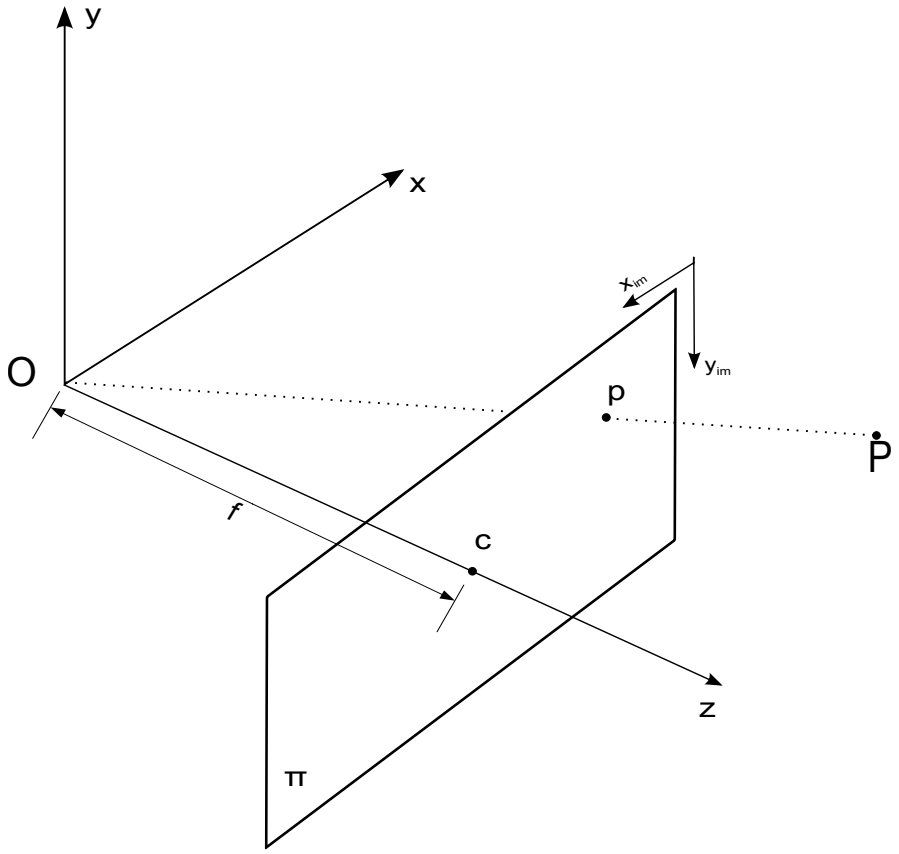

Figure A.1 – The pinhole or perspective camera model. The model consists of an image plane π and a 3D point O, representing the *camera center* or the *focus of projection*. The positive z axis that goes from O through the center c of the image plane is called the *optical axis*. A 3D point in the world P is projected on the image plane π at point p. x_{im} and y_{im} represents the coordinate system of the image.

- the transformation between the camera frame coordinates and pixel coordinates;
- the geometric distortion introduced by the optics.

The pinhole camera model is illustrated in Figure A.1. The model consists of an image plane π and a 3D point O, representing the *camera center* or the *focus of projection*. The positive z axis that goes from $O = [\ 0 \quad 0 \quad 0\ ]^t$ through the center $c = [\ c_x \quad c_y \quad f\ ]^t$ of the image plane is called the *optical axis*.

The first set of parameters (the focal length f[1]) allows to project a 3D point in the camera world $P = [\ X \quad Y \quad Z\ ]^t$ on the image plane π at point $p = [\ x \quad y \quad z\ ]^t$[2]. The image plane, perpendicular to the optical axis, is located at a distance of the focal length f of the camera. The following equations allow the projection of P on the image plane π, expressed in the camera coordinate system:

$$x = f\frac{X}{Z}$$

$$y = f\frac{Y}{Z} \tag{A.1}$$

The second set of parameters allows to convert the projected point $p = (x,y)$, expressed in the camera's coordinate system, in pixel coordinates (image's coordinates) (x_{im},y_{im}). This conversion also involves the actual, physical dimensions (in mm) of the pixels of the camera (s_x,s_y) and the center of the image plane $c = (c_x,c_y)$. Note that the sign change in Equation A.2 is due to the fact that the camera's coordinate system and the image reference frame have inverted x and y axis (Figure A.1).

$$x = -(x_{im} - c_x)s_x$$

$$y = -(y_{im} - c_y)s_y \tag{A.2}$$

The third and last set of parameters is related to the radial distortion introduced by the imperfections of the optics in the camera. The distortion becomes really visible near the edge of the image (distortion at (c_x,c_y) is null) – easily resulting in a shift of several pixels. Fortunately, these deformations can be modeled with parameters $k_1,\ldots,k_n$ where, in most standard calibrations, $n = 4$ is sufficient to undistort an image for it to be usable for processing.

Most of CV algorithms expect an undistorted image as an input. It means that the image should be remapped to compensate for its shifted center c and the radial distortion modeled with parameters $k_1,\ldots,k_n$. Most CV libraries (*e.g.* Intel's OpenCV) can determine these sets of parameters using an once-in-a-lifetime calibration process and undistort images.

[1]The focal length can be different in x and y direction resulting in having two focal length values f_x and f_y. We however make the assumption that these values are equals resulting in a single f value.

[2]As the z component of a point on the image plane is always f, the representation $p = [\ x \quad y\ ]^t$ is used instead of $p = [\ x \quad y \quad f\ ]^t$.

Extrinsic parameters

Extrinsic parameters represents the transformation from the world's coordinate system to the camera reference frame. This can simply be modeled using a rotation R and a translation T in 3D space. A point in the world can be brought in the camera's coordinate system using the following matrix transformation:

$$P_c = R(P_w - T) \tag{A.3}$$

Transformations summary

If a point $P = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^t_w$ in the world needs to be converted to a position in pixels in the image's reference frame ($\begin{bmatrix} x & y & z \end{bmatrix}^t_{im}$), all the transformations described above can be combined in a single matrix expression, where M_{int} and M_{ext} are respectively the matrix representing the intrinsic and extrinsic parameters of the camera, f is the focal length of the camera, (s_x,s_y) are the physical dimensions (in mm) of the pixels of the camera, R is a rotation matrix and T is a translation matrix:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{im} = M_{int} M_{ext} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_w \tag{A.4}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{im} = \begin{pmatrix} -f/s_x & 0 & c_x \\ 0 & -f/s_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R_{3x3} \vdots & -R^t T_{3x1} \end{pmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_w \tag{A.5}$$