UNIVERSIDADE FEDERAL DE PERNAMBUCO

INCREMENTAL SEMANTIC TRACKING ON MOBILE DEVICES

Rafael Alves Roberto

Ph.D. Thesis

RECIFE

2018

UNIVERSIDADE FEDERAL DE PERNAMBUCO

INCREMENTAL SEMANTIC TRACKING ON MOBILE DEVICES

Rafael Alves Roberto

A Ph.D. Thesis presented to the Informatics Center of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Philosophy Doctor in Computer Science.

Advisor: *Veronica Teichrieb*
Co-Advisors: *João Paulo Silva do Monte Lima and Hideaki Uchiyama*

RECIFE
2018

Rafael Alves Roberto
Incremental Semantic Tracking on Mobile Devices/ Rafael Alves Roberto. – RECIFE, 2018-
109 p. : il. () ; 30 cm.

Advisor Veronica Teichrieb

Ph.D. Thesis – Universidade Federal de Pernambuco, 2018.

1. Mobile Devices. 2. Depth Sensor. 3. SLAM. I. Veronica Teichrieb. II. Universidade Federal de Pernambuco. III. Centro de Informática. IV. Incremental Semantic Tracking on Mobile Devices

CDU

02:141:005.7

Rafael Alves Roberto

"**Incremental Semantic Tracking on Mobile Devices"**

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Aprovado em: 27/06/2018.

_____

Orientadora: Profa. Veronica Teichrieb

BANCA EXAMINADORA

_____

Prof. Silvio de Barros Melo
Centro de Informática / UFPE

_____

Prof. João Marcelo Xavier Natário Teixeira
Departamento de Eletrônica e Sistemas / UFPE

_____

Prof. Douglas Guimarães Macharet
Departamento de Ciência da Computação / UFMG

_____

Prof. Dieter Schmalstieg
Faculty of Computer Science and Biomedical Engineering /
Graz University of Technology

_____

Prof. Takafumi Taketomi,
Graduate School of Information Science /
Nara Institute of Science and Technology

Acknowledgments

O doutorado é caminho longo, tanto que quando este aqui começou não existia nenhuma piada com os números 7 e 1 e a gente não sabia que o placar da FIFA tinha barra de rolagem. Falo com tranquilidade que eu não teria conseguido chegar aqui sem o suporte de muita gente. Primeiramente, meus pais, irmão e irmã, que me acompanham numa jornada ainda mais longa que essa. Vocês moram de pantufas no meu coração! Também à minha esposa, por estar presente durante os sacrifícios e caos absoluto e eterno que é fazer um doutorado. Por me acompanhar até o outro lado do mundo (literalmente). Você é minha guerreirinha! Aos meus orientadores por me darem oportunidade de fazer o meu nome, por servirem de inspiração e por todas as discussões e contribuições. Me vejo obrigado a concordar que isso foi fundamental para a qualidade final deste trabalho. Só conselho top! Não posso esquecer os companheiros e companheiras do Voxar Labs por todos os momentos de discussão, descontração e mão na brasa que me fizeram crescer bastante. Vocês tem talento pra isso! Por fim, mas não menos importante, aos amigos e amigas que, apesar do ritmo ragatanga insano, a gente conseguiu manter um convívio social honestíssimo. Vocês me ajudaram a não ficar mais louco que o padre baloeiro.

I would like to say a few words in English to my friends overseas. First, I would like to thank my advisors in Japan and Austria for having me there and for all the discussions. Your contributions were very important to this Ph.D. I also would like to thank people at LIMU and ICG for teaching me so much. And last but not least, I want to leave my gratitude to the people at ICG and Kroisbach for the good moments the helped us to feel at home when we were far away from our comfort zone.

Abstract

Tracking is an important task that is used for several applications, such as navigation assistance and augmented reality. The improvement and popularization of mobile devices in recent years allowed these applications to be executed on such devices, which provides a mobility that is not possible on desktop computers. Although there is an improvement in both tracking techniques and mobile devices, there are still several challenges in this field. In this sense, the goal of this Ph.D. is to investigate methods to perform tracking on mobile devices considering the characteristics of such platform.

To achieve this goal, the first step was to conduct a systematic mapping on tracking for mobile devices in order to classify the area and identify the state of art methods as well as the gaps. This study collected 2,602 papers from three scientific databases using an open-source crawler, from which 444 were selected to be classified according to four properties: tracking type, degrees of freedom, tracking platform and research type. The results indicated a growing interest in this field. It showed that most works use the device's sensors for tracking in location-based applications and almost all of them calculate a 2D or 2D + θ pose. Beyond that, there is a clear preference for systems that calculate the pose locally on the device and only a few use a remote server to assist in this task. However, it was not found any work that extracts semantics on such devices.

The mapping was used to elaborate preliminary experimental scenarios aiming to create a practical knowledge on the specificities of developing tracking techniques for mobile devices. First, the Google Tango platform was evaluated to establish a ground base of the state-of-the-art trackers. It was observed that the precision in indoor spaces is suitable to provide a good user experience, including for augmented reality applications. Another experiment evaluated the use of parallelism, distributed approach and native implementation. This test showed that using parallel implementation was faster, but it required more memory than any other architecture whereas the distributed approach has the opposite results. On average, native development was the most efficient. Besides that, experiments were designed intending to test different tracking techniques that the systematic mapping indicated to be suitable for mobile devices. One is a face tracking technique using machine learning and local binary features. This algorithm was adapted to consider the characteristics of mobile devices and it runs in approximately eight milliseconds on such equipments. However, the size of the training file can be a limiting factor for the use of this type of method on mobile devices. The other one is a SLAM technique that was developed in desktop and evaluated in the challenging scenario of the ISMAR Tracking Competition. Additionally, STAM was ported to a Tango tablet device. The mobile version was evaluated in comparison with the desktop implementation and it proved to be slower than its desktop counterpart.

There were several lessons learned from the experiments. One of them was the importance of finding high-level semantic information from a scene, which can improve tracking and provide more realistic rendering. In this Ph.D., it was developed a technique that incrementally detects and tracks primitives using the generating process of point clouds of visual SLAM systems, called Geometric and Statistical Incremental Semantic Tracker (GS-IST). The main idea of this approach consists in taking on every keyframe the sparse point map created by a visual SLAM system to detect basic primitives using a batch-based method. These primitives are planes, spheres and cylinders. Then, it uses the parameters of the shapes found to fuse all the primitives belonging to an object into one in order to improve their representativeness and reliability. After that, the detected shapes are matched with those previously found. To do this, it is used the intersection of the 3D bounding box and the distance between the center of mass of these primitives. Later, it verifies if the current estimation is following

the detection history over time. If the current shape has a type that is different from the one that appears most of the time, it is updated to that class of primitive. Moreover, if a shape that was not detected on the current keyframe appeared previously, the system can recover that shape. Finally, the system computes the reliability of each estimated primitive to suppress incorrect detections in a noisy point cloud. This is done by analyzing the geometric relationship between the point cloud and the estimated shapes and performing a statistical evaluation to measure the randomness of the detection over time.

The experiment indicates that GS-IST was able to improve both precision and stability of existing methods. However, since it focuses on precision, it compromises the recall to ensure the detection and tracking of correct shapes. One benefit of having the shapes is to improve the representation of the data. In the experiments performed, it was observed that representing a shape using primitives requires almost nine times less memory to describe the scene than using the point cloud. Moreover, the evaluation also suggests that it is possible to segment more than 70% of the point cloud.

In order to evaluate how GS-IST would perform running on mobile devices, it was ported to the Android platform and evaluated using two distinct phones. The evaluation showed that the mobile version is 8.5 to 9.9 times slower in comparison with the desktop implementation depending on the smartphone. Moreover, it uses up to 30.5% of the CPU load, which allows this implementation to run on a separate thread of the main tracking technique. Additionally, the energy consumption was not a concern because GS-IST can run for more than 4 hours in the worst case. Finally, the memory usage was less than 8% of the total RAM memory of the test devices, which did not have an impact on the execution time.

Keywords: Semantic tracking, visual SLAM, mobile devices, tracking, Android.

Resumo

Rastreamento é uma atividade importante usada em várias aplicações, como auxílio à navegação e realidade aumentada. Atualmente, estas aplicações podem ser executadas em dispositivos móveis graças à popularização e à melhoria desses aparelhos, dando uma mobilidade que não é encontrada nos computadores. Apesar das melhorias tanto nos dispositivos como nas técnicas de rastreamento, ainda existem problemas em aberto nessa área. Dito isto, o objetivo desta pesquisa de doutorado é o de investigar métodos para realizar rastreamento em dispositivos móveis considerando as características desta plataforma.

Para atingir esse objetivo, primeiramente foi conduzido um mapeamento sistemático sobre rastreamento para dispositivos móveis com o objetivo de classificar a área e identificar as técnicas mais recentes e suas limitações. Este estudo coletou 2.602 artigos de três bases de dados científicas usando um buscador de código aberto, dos quais 444 foram selecionados para serem classificados de acordo com quatro critérios: tipo de rastreamento, graus de liberdade, plataforma de rastreamento e tipo de pesquisa. O resultado mostrou um interesse crescente na área. A maioria dos trabalhos usa os sensores do aparelho para fazer o rastreamento em aplicações que provem a localização do usuário e muitos deles calculam uma pose 2D ou 2D + θ. Além disso, foi observada uma preferência por sistemas que calculam a pose localmente no aparelho e poucos estudos usam um servidor remoto para auxiliar nessa tarefa. Entretanto, não foi encontrado nenhum trabalho que extrai semântica nesses aparelhos.

O mapeamento foi usado na criação de experimentos preliminares com o objetivo de adquirir um conhecimento prático sobre as especificidades de desenvolver técnicas de rastreamento para dispositivos móveis. Inicialmente, o Google Tango foi avaliado para encontrar um referencial de precisão para os rastreadores atuais. Foi observado que sua precisão em espaços fechados é adequada para prover uma boa experiência para o usuário, incluindo aplicações de realidade aumentada. Outro experimento avaliou o uso de paralelismo, execução distribuída e implementação nativa. Estes testes mostraram que usar paralelismo deixa a execução mais rápida, mas usa mais memória, enquanto a implementação distribuída tem o resultado oposto. Na média, a implementação nativa foi a mais eficiente. Além disso, foram criados experimentos para testar diferentes técnicas de rastreamento identificadas no mapeamento como adequadas para dispositivos móveis. Uma delas foi o rastreamento de faces usando aprendizagem de máquina e características binárias locais. Essa abordagem foi adaptada considerando as limitações dos dispositivos móveis e é executada em aproximadamente oito milissegundos nesses aparelhos. No entanto, o tamanho do arquivo de treinamento pode ser um fator limitante para o uso deste tipo de método em dispositivos móveis. O último experimento está relacionado com uma técnica de SLAM (sigla em inglês para Localização e Mapeamento Simultâneos) chamada STAM. Ela foi desenvolvida para desktop e avaliada na competição de rastreamento do ISMAR. Posteriormente, o STAM foi portado para um aparelho com suporte ao Google Tango. A versão móvel foi comparada com a implementação desktop e mostrou-se mais lenta.

Várias lições foram aprendidas a partir dos experimentos. Uma delas foi a importância de encontrar informações de semântica de alto-nível de uma cena, que podem ser usadas para melhorar o rastreamento ou criar renderizações mais realísticas. Neste doutorado foi desenvolvida uma técnica que detecta e rastreia primitivas geométricas de maneira incremental usando o processo de geração de nuvens de pontos dos sistemas de SLAM, chamada GS-IST (sigla em inglês para Rastreador Semântico Incremental Geométrico e Estatístico). Em cada quadro-chave, essa abordagem acessa o mapa criado pela técnica de SLAM e detecta as primitivas usando um método não-incremental. Essas primitivas são planos, esferas e cilindros.

Em seguida, ela usa os parâmetros das formas encontradas para combinar as primitivas que pertencem ao mesmo objeto de modo a aumentar sua representatividade e confiabilidade. Depois disso, as primitivas são relacionadas com aquelas encontradas em instantes anteriores. Para isso, é usada a intersecção do invólucro dos pontos e a distância do centro de massa dessas primitivas. Posteriormente, o sistema verifica se a estimativa atual da primitiva é a mesma da sua detecção através do tempo. Caso não seja, a técnica atualiza a forma geométrica para estar de acordo. Além disso, se uma primitiva que aparecia antes não foi detectada, ela pode ser recuperada. Por fim, se calcula a confiabilidade de cada forma geométrica encontrada para suprimir aquelas que são incorretas dada uma nuvem de pontos ruidosa. Isto é feito a partir da análise da relação geométrica entre a nuvem de pontos e as primitivas estimadas, além de uma avaliação estatística para medir a aleatoriedade da detecção através do tempo. A avaliação do sistema indicou que o GS-IST foi capaz de melhorar a precisão e a estabilidade dos métodos existentes. Porém, como o foco está na precisão, a técnica peca na revocação para garantir a detecção e rastreamento das primitivas corretas. Um benefício de ter formas geométricas parametrizáveis é que ela melhora a representação dos dados. Na avaliação foi observado que o uso de memória com esta representação é quase nove vezes menor do que representar a cena usando apenas a nuvem de pontos. Além disso, a avaliação mostrou que é possível segmentar mais de 70% desses pontos.

Para avaliar como o GS-IST se comportaria sendo executado em dispositivos móveis, ele foi portado para a plataforma Android e avaliado usando telefones distintos. A avaliação mostrou que a versão móvel é, dependendo do aparelho, de 8,5 a 9,9 vezes mais lenta quando comparada com a implementação em desktop. Mais do que isso, ele usa até 30,5% da capacidade da CPU, permitindo que esta implementação possa ser executada em uma thread separada da técnica de rastreamento. Além disso, o consumo de energia não foi uma preocupação, uma vez que o GS-IST pode ser executado por mais de 4 horas no pior cenário. Finalmente, o uso de memória RAM foi inferior a 8% do total disponível nos aparelhos testados, o que não apresentou nenhum impacto no tempo de execução.

Palavras-chave: Rastreamento semântico, SLAM visual, dispositivos móveis, rastreamento, Android.

List of Figures

List of Tables

Table of contents

1
Introduction

Several applications require tracking, which is the computation of an object placement relative to a real world element or location over a time period. For instance, some augmented reality software use the camera position related to a marker to display a virtual content registered with the pattern (Roberto et al. 2013). Another example is a GPS-based navigation system that calculates its location relative to the road in order to show the driver directions to a destination (Leshed et al. 2008), seen in Figure 1.1 (a). However, this is still a challenging task. Moreover, determining this placement can demand a lot of computational power and memory depending on the approach and the required information.

Figure 1.1: Example of mobile applications that need tracking to deliver experience to the users.

Mobile devices, such as phones and tablets, are becoming increasingly popular. Research shows that a median of 43% of the world's population owns a smartphone (Pew Research Center 2016). Moreover, these devices are continuously improving regarding processing power and memory space available (Halpern et al. 2016), which makes them powerful enough to perform complex tasks, such as tracking. In fact, the processing power on mobile devices is increasing rapidly enough to reduce the gap with desktop computers. While in 2009 the desktop CPU clock frequency was 5.7 times the mobile CPU clock, in 2015 this distance dropped to 2.1 times. This scenario favors the creation of numerous types of applications since such devices create several opportunities that are only possible when the user can be mobile. One example is to annotate relevant information precisely on the facade of buildings in an outdoor environment (Yovcheva et al. 2012), as illustrated in Figure 1.1 (b).

Besides all the improvement on the mobile devices itself, several tracking techniques that are capable to run on such devices were released in the last couple of years. There are examples in the academy and

in the industry. The most distinguished were the ARCore[1] and ARKit[2], by Google and Apple and shown in Figure 1.1 (c) and (d) respectively. However, there are still several gaps in the field that can be illustrated with a challenging scenario, such as creating a 3D model of an entire house in order to redecorate it. Tracking an environment that is so large demands manipulation of a large amount of data, which impacts both the processing power, especially because the architecture of mobile processors compromise on speed to be more efficient on energy consumption and on temperature. Although memory is not so limited on these devices nowadays, using algorithms such as bundle adjustment can be an issue in a scenario like this. Additionally, the walls must be aligned in order to correctly recreate all the rooms. And in each room there are several objects that also need to be recognized and modeled, such as tables, sofas and wardrobe.

This particular scenario presents several research opportunities and one is extracting and tracking high-level semantic information. Several types of semantic can be collected, from the geometric primitives of the objects to the model of a piece of furniture and this process is referred to as semantic modeling. Shape parameters of geometric primitives and the relationship between them are valuable knowledge to be estimated especially in human-made environments such as a house. These semantics are useful for replacing redundant point clouds with more efficient data structures or aligning the walls based on their parameters. This data can also be gathered in different ways: from the input image, the scene map or a combination of both.

1.1 Objectives

In this sense, the objective of this Ph.D. thesis is to **investigate methods to perform tracking on mobile devices considering the characteristics of such platform**.

In order to achieve this goal, four specific objectives should be met. The first one is to **perform a strong literature review of tracking for mobile devices to find relevant studies and gaps**. One effective way to analyze the related works of a research area that has so many studies is through a systematic mapping, which is a research method to review, classify and provide an overview of a wide range of papers on a particular topic. *Chapter 2* describes how the systematic mapping was performed in this work and presents its major findings.

The second specific objective of this Ph.D. is to determine experimental scenarios aiming to **acquire a practical knowledge on the specificities of developing tracking techniques for mobile devices**, which are based on the results of the systematic mapping. The goal is to evaluate different tracking approaches for mobile device in order to assess a state-of-the-art tracking method and find an efficient architecture for a computer vision system on mobile devices. Additionally, test different tracking techniques to decide which ones are suitable for such devices and which ones are not, as explained in *Chapter 3*.

The lessons learned from both the systematic mapping and the experiments were the foundation to **create a semantic tracking technique for sparse visual SLAM**, which is the third specific objective of this Ph.D. This approach incrementally detects and tracks primitive shapes using geometric and statistical analyses. *Chapter 4* details this method and presents the evaluation results.

And last but not least, the fourth specific objective of this study is to **port the semantic tracking approach to mobile devices and evaluate how it performs running on such platforms**. *Chapter 5* shows the different criteria used to evaluate this technique and discusses the implication of the results.

[1] `https://developers.google.com/ar/`
[2] `https://developer.apple.com/arkit/`

2
Systematic Mapping

During the past years, researchers have proposed different techniques to perform tracking on mobile devices. A preliminary search for related works revealed a significant amount of papers in this area. Therefore, it becomes important to summarize the current state of the art and provide an overview of the trends in this specialized field. In order to address this issue, it was performed a systematic mapping of the literature in this area. The main goal of this mapping is to analyze, classify and map existing papers about tracking for mobile devices, providing a primary study and an inclusive overview of this topic. The result of the systematic mapping was published in (Roberto et al. 2016c), and an update of this study is detailed in sequence.

Systematic mapping is a method to review, classify and structure papers related to a specific research field (Petersen et al. 2008). It is frequently used in medical research and lately has been applied to software engineering. Unlike systematic reviews, the goal of this research method is not to perform a deep analysis of works in order to identify the best practices of a field, which usually includes a quality evaluation. The aim of a systematic mapping is to provide an overview of a wide range of papers. This broader analysis enables to observe more studies, which allows more general conclusions (Petersen et al. 2008). Nevertheless, both methods use a well-defined methodology, which reduces bias (Keele 2007). Moreover, systematic mapping papers have an educational value to provide valuable information for students and young researchers, being a useful first step for Ph.D. candidates (Kitchenham et al. 2010).

To the best of the author's knowledge, there is currently no study that synthesizes or systematically analyzes, classifies and maps existing papers about tracking for mobile devices. However, some surveys were found on the field or one of its specific subareas. For instance, (Liu et al. 2007) evaluated wireless indoor localization techniques and (Lane et al. 2010) listed tracking algorithms for mobile phones that use only their sensors, as well as related applications. There are also surveys regarding mobile augmented reality, in which tracking is an important step. Examples are (Olsson et al. 2011) that studied the overall acceptance and user experience of mobile augmented reality consumer applications, (Huang et al. 2013b) that presented the technologies and methods to perform augmented reality on mobile devices and introduces some applications, and (Grubert et al. 2011) that conducted a survey about augmented reality browsers and performed a quantitative and qualitative analysis regarding the usability aspects of these tools.

As a definition, tracking for mobile devices means that an off-the-shelf cell phone or tablet extracts information from the environment and then processes it locally or remotely in order to compute the device's pose related to the world, which will be used by an application or a service on the device itself.

2.1 Methods

The systematic mapping was conducted based on the process proposed by (Petersen et al. 2008) and illustrated in Figure 2.1 in which a list of research questions is proposed, which guides the search strategy, the definition of inclusion and exclusion criteria for relevant studies and the classification schema of all the selected studies. The process steps performed in this study are described in the following subsections.

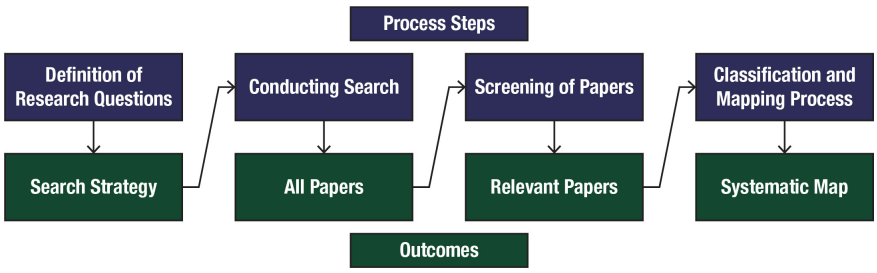


Figure 2.1: Systematic mapping process. The research question guides the definition of the search strategy, which is used to collect the works. Some criteria are defined to select the relevant studies that are classified in order to provide the systematic mapping.

2.1.1 Research Questions

The goal of this systematic mapping is to provide an overview of the current research on the topic of tracking for mobile devices. The overall objective was defined in the following four research questions:

RQ1 How has the frequency of research on tracking for mobile devices changed recently?

RQ2 What are the most common approaches of tracking for mobile devices?

RQ3 In which platforms has tracking for mobile devices been executed?

RQ4 In which forums has research on tracking for mobile devices been published?

The first question aims to use the number of publications to investigate trends of the field in the past few years. The second and third questions explore the approaches and platforms researched in the field. The objective of the fourth question is to identify where tracking for mobile devices research can be found, which could be targets for the publication of future studies.

2.1.2 Scientific Databases and Search Strategy

Three online academic search engines were used to find the relevant papers:

- ACM Digital Library;
- IEEE Xplore Digital Library;
- ScienceDirect.

In order to perform an automatic search on the selected libraries, the search string consisted of two parts. The former regards the tracking domain and the later covers the device used. Thus, the search string was the following:

(“tracking” OR “registration” OR “localization”)
AND
(“phone” OR “tablet” OR “handheld” OR “smartphone”)

Tracking is the key term of the first segment and the other ones are its most used synonyms. Other terms were not used because a preliminary analysis showed that the majority of the papers found would not be selected for classification. An example is “positioning”, which appears mostly in studies in which the device’s pose is used only by an external agent and not on the device itself, such as the phone’s position that is

employed by the carrier to determine in which GSM antenna it will connect to. Moreover, the analysis revealed that the relevant papers were already found using the chosen terms.

Regarding the second segment, it was chosen to search for each device instead of using the terms “mobile” or “mobile device”. The reason is that these keywords returned too many papers and a preliminary analysis revealed that the vast majority of them use a broader concept of mobile devices than the desired in this mapping. For instance, some works use the term “mobile objects” for objects with sensors embedded that are tracked by computers. There are also several references to mobile device as a large object that is used for tracking-related activities, such as airplane radar or medical scanner, which was shrunk to become mobile. Therefore, using the types of mobile device as search terms showed to be more efficient.

An automatic search was performed in the aforementioned databases using an open-source paper crawler[1] software and applying the search in the title, abstract and keywords. The crawler was developed during this Ph.D. and aims to automate the process of retrieving papers. Hence, the crawler accesses the digital libraries, performs the search using the search strings, collects the papers, eliminates duplicate versions and creates a worksheet containing all the works with their title, year, source, abstract and web address.

2.1.3 Screening of Papers

After collecting the papers, the crawler automatically remove duplicate works. Whenever a work had multiple publications, only the most complete version was selected and the other ones were removed as duplicates. Later, relevant papers were manually selected using the following inclusion and exclusion criteria.

- Inclusion criteria:
  - Papers about tracking techniques implemented on mobile devices;
  - Papers about mobile applications that use existing tracking techniques, even if they do not explain how tracking was implemented.
- Exclusion criteria:
  - Papers published before 2009, which is one year after the release of phone models that allowed 3rd party development;
  - Papers not written in English language;
  - Papers published on non-peer reviewed vehicles, such as books and magazines;
  - Papers not related to tracking techniques on mobile devices;
  - Papers about tracking techniques that were implemented only on desktop platform and that have no indication of how they can be developed for mobile devices.

2.1.4 Classification

Following, all included papers were classified according to four properties in order to answer the research questions. They are detailed next.

2.1.4.1 Tracking Type

Each paper was classified regarding its tracking type. The classification was adapted from (Zhou et al. 2008), which is shortly explained as follows and illustrated in Figure 2.2.

[1] Available at `https://goo.gl/7t8kG8`

- **Sensor-Based Tracking:** techniques that calculate device's pose relative to real world using exclusively non-vision based sensors. This approach can be divided in two categories: *single sensor*, which uses only one sensor for tracking, and *sensor fusion*, which uses different sensors to perform the same task;
- **Vision-Based Tracking:** techniques that use images captured by the device cameras to calculate pose relative to the real world. This approach can also be divided in two categories: *marker-based* and *natural feature-based*. The former method calculates device's pose from artificial markers placed in the scene and the latter performs the same task using natural characteristics from the environment, such as points and edges. The natural feature-based approach was also split into two subcategories: *static model* and *dynamic model*. The first one uses prior knowledge of the scene that does not change during tracking to compute device's pose and in the second one the tracker can use an initial model if it is available or build it entirely from scratch and this environment information is updated during computation of the device's pose;
- **Hybrid Tracking:** techniques that combine sensor-based and vision-based methods to calculate device's pose;
- **Several:** papers that present techniques from several categories, such as surveys.

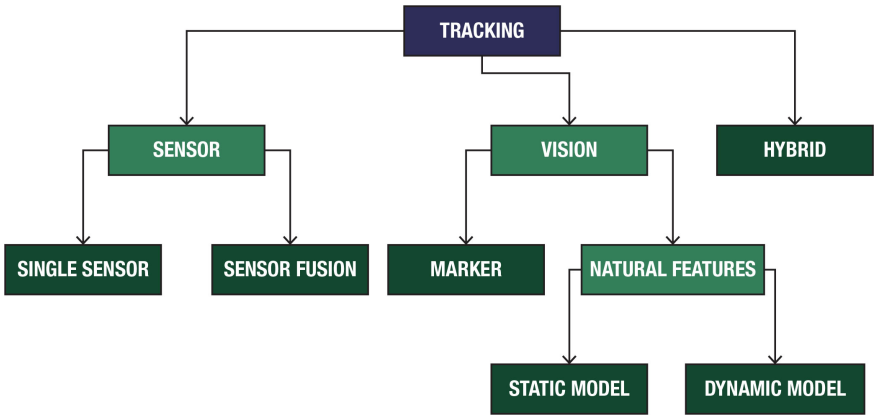


Figure 2.2: Tracking type classification diagram.

2.1.4.2 Degrees of Freedom

This property details the degree of freedom required to compute the information desired. This classification was based on (Normand et al. 2012). One modification was the addition of the 3D degree of freedom, which was not mentioned in the original work. Thus, the complete degree of freedom classification used in this work is detailed in the following list.

- **0D:** techniques that detect a pattern and display an information about it without any relationship with its position and orientation;
- **2D:** techniques that provide information about the position, being indoor, outdoor or in the screen. It can also be called "2D Location";
- **2D + θ:** techniques that extend the position information with orientation, providing the location with direction. It can also be named "2D Location + Orientation";

- **3D:** techniques that compute the device's rotation in all three axis;
- **6D:** techniques that calculate the device's pose with rotation and translation. Systems that also compute scale were considered 6D as well;
- **Several:** papers that present techniques from several categories.

2.1.4.3 Tracking Platform

Two tracking platforms were considered to classify papers regarding this property, as detailed below.

- **Local Tracking:** techniques that compute all the required information at the mobile device;
- **Distributed Tracking:** techniques in which part or all the information is calculated on a server and the result is transmitted to the device and used to display the content;
- **Several:** papers that present techniques from both categories.

2.1.4.4 Research Type

The research type feature concerns the research approach used in the papers. This classification was adapted from (Wieringa et al. 2005) and is summarized in the list below.

- **Evaluation Research:** papers that present implementation and extensive evaluation of existing techniques in order to determine their benefits and drawbacks;
- **Opinion Papers:** publications in which the author expresses a personal opinion whether a certain topic is good or bad without relying on related work;
- **Philosophical Papers:** papers that present new ways of looking at existing things, such as structuring the field in form of a new taxonomy;
- **Proposal of Solution:** works that propose solutions for problems, which can be based on novel or existing techniques;
- **Survey Papers:** papers that summarize and organize a research field based on other publications;
- **Technique Research:** publications in which the authors propose and implement a novel technique.

2.1.5 Threats to Validity

It is important to consider threats to validity in order to judge the systematic mapping strengths and limitations. The main issues are related to incomplete sets of relevant papers and researcher bias with regards to inclusion/exclusion criteria and classification.

Limitations with search string, scientific databases and search strategy can result in an incomplete set of relevant papers. As a way to mitigate that risk, three strategies were used. In order to validate the search string, the terms were discussed with five other experienced researchers in the field of tracking. The scientific databases that publish works from the most important conferences and journals in the area were selected. As for the search strategy, a different approach was used to maximize the number of papers found. Instead of using the complete search string, twelve different searches were performed using a two-by-two combination of

every term in both parts of the search string. Using this strategy, it was possible to retrieve almost 34 times more papers than when the complete string was employed.

The Ph.D. candidate conducted the analysis to include/exclude and classify a paper. Since this may lead to a researcher bias, 11.74% of the studies were randomly selected before the subjective part of the screening phase to compose a set of control papers, and one of the co-advisors, which is a experienced researcher in the field of tracking, analyzed them. The results were compared using the Cohen's Kappa coefficient, which measures the agreement between the two classifications taking into account how much agreement would be expected to be present by chance (Cohen 1960). The coefficient lies between -1.0 and 1.0 in which 1.0 denotes perfect agreement, 0.0 indicates that any agreement is due to chance and negative values present agreement less than chance. Cohen's Kappa was used to measure the reliability regarding inclusion and exclusion of papers and the classification of the included papers in common according to the classification schema. There is no consensus on what are good levels of agreement. Nevertheless, a common scale (Altman 1990) indicates that there is no agreement for negative values, poor agreement between 0.00 and 0.20, fair agreement between 0.21 and 0.40, moderate agreement between 0.41 and 0.60, good agreement between 0.61 and 0.80 and very good agreement for values higher than 0.80. At first, the classification ratio was in the range of good agreement. The main reason for that was the fact that the first classification schema was leading to dubious interpretations. For instance, natural feature tracking was divided into model-based and model-less approaches, in which it was not clear if information used could be considered a model or not. The classification schema was refined to the one previously presented, which uses a more straightforward classification, and all papers were then reclassified. Thus, the included/excluded papers Cohen's Kappa coefficient was 0.8062 $\pm$ 0.0495 and the Cohen's Kappa classification was 0.8345 $\pm$ 0.0303.

2.2 Results

The search was made on March 3rd, 2016 and resulted in 2,602 papers found. As can be seen in Figure 2.3, 611 papers were removed for being duplicated and 1,991 studies were available for the subjective steps of screening. Only 444 works remained for trends analysis and classification.

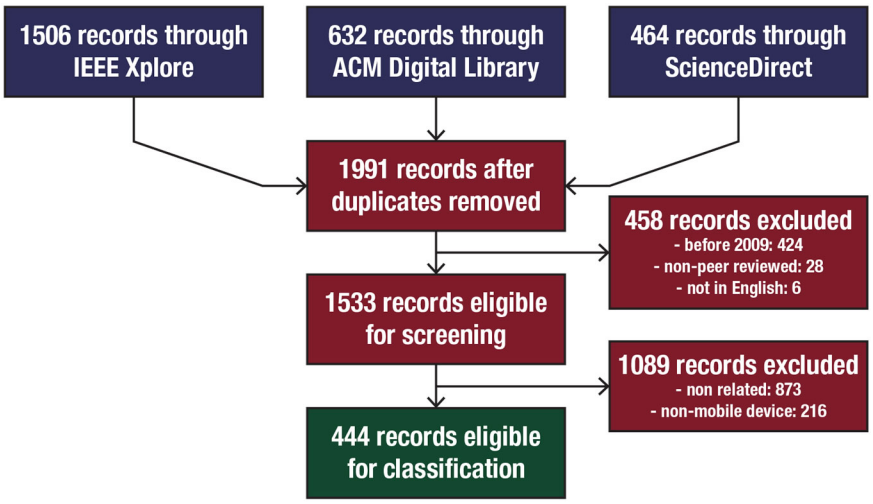


Figure 2.3: Selection process shows the number of papers included and excluded and the reasons for exclusions.

Figure 2.4 shows the annual trend of papers. It is possible to see that the number of studies is growing between 2009 and 2014. Even with a reduction in the number of publications in 2015, there is an indication of increasing interest in tracking for mobile devices in recent years.

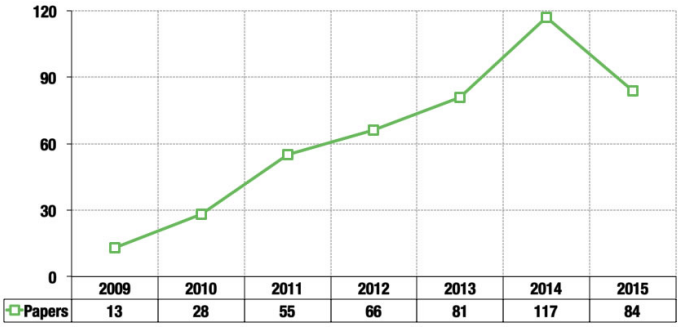


Figure 2.4: Publications over time. Annual trend of included papers.

The 444 papers were published in 260 forums. As seen in Table 2.1, almost a quarter of all works came from the top 12 venues. ISMAR is the flagship event in the field with 30 studies. Preferable targets for such papers are conferences and symposiums in which 196 papers were published. They were followed by 52 journal works and 12 workshop studies.

Each paper was classified according to the scheme presented in the previous section. The full list of works can be accessed through an open-source web application (Figueiredo et al. 2015). Using this system, it is possible to filter the papers according to the year when the works were published and the classification criteria, as well as search for terms and words in the abstract. Moreover, collaborators can send new entries of studies about tracking for mobile devices, which will be revised and then added to the online data set. The web application can be accessed at `http://cin.ufpe.br/~rar3/tracking_sm/`.

Table 2.1: List of the most popular publication forums.

Forum	Acronym	Type of Forum	Number of Papers	Percentage of the Total
International Symposium on Mixed and Augmented Reality	ISMAR	Symposium	30	6.76%
International Conference on Indoor Positioning and Indoor Navigation	IPIN	Conference	19	4.28%
Conference on Embedded Networked Sensor Systems	SenSys	Conference	7	1.58%
International Conference on Mobile Computing and Networking	MobiCom	Conference	6	1.35%
Pervasive and Mobile Computing	-	Journal	6	1.35%
IEEE Transactions on Mobile Computing	-	Journal	5	1.13%
IEEE Virtual Reality Conference	IEEE-VR	Conference	5	1.13%
International Conference on Computer Vision	ICCV	Conference	5	1.13%
Conference on Multimedia and Expo	ICME	Conference	5	1.13%
International Conference on Pervasive and Ubiquitous Computing	UbiComp	Conference	5	1.13%
Int. Conference on Pervasive Computing and Communications	PerCom	Conference	5	1.13%
Symposium on 3D User Interfaces	3DUI	Symposium	5	1.13%
Other 248 Forums	-	-	341	76.80%

2.3 Mapping

From the classification of the studies it is possible to establish a mapping that aims to provide an overview of tracking for mobile devices and can help to identify potential research gaps. This map gives the distribution of works for each classification criteria, their annual trends and the relation between them.

2.3.1 Classification Distribution

It is possible to see in Figure 2.5 that most of the works, such as (Ando et al. 2014), rely on a combination of the devices' sensors to calculate pose, and that marker-based tracking, which is used for example in (Oui et al. 2011), is the least used method for the same task. Moreover, it can be noted that vision-based methods like (Wagner et al. 2010a) are present in three out of ten papers.

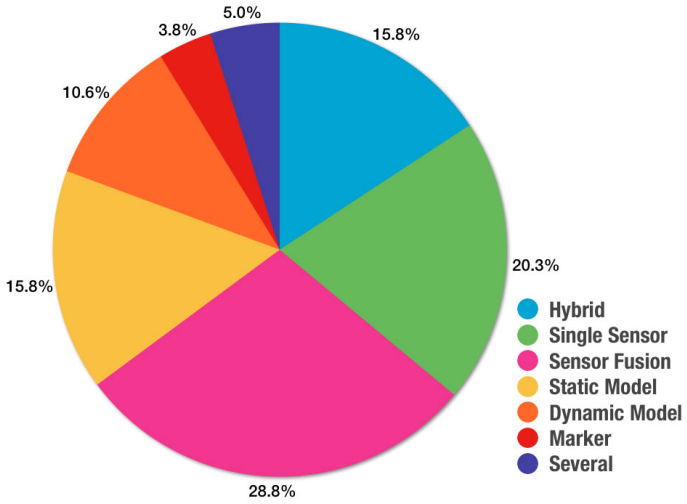


Figure 2.5: Tracking type distribution over the database.

Table 2.2 lists the sensors used on each paper in which the tracking type is hybrid or based on sensors as well as the number of studies that uses them. Several works fuse different sensors and Table 2.2 also lists the 13 most common combinations.

Regarding the degree of freedom found in the works, several of them calculate a 6D pose (Hagbi et al. 2011), as shown in Figure 2.6. However, in 58.5% of the studies, a 2D position is computed. In some papers, a 2D position on the screen is found (Song et al. 2011), but the majority discovers this 2D position on the environment (Hu et al. 2010). In the latter case, more works also find the orientation θ (Shin et al. 2012a), and the least common papers are the ones aiming 0D systems (Teraura et al. 2012).

In the majority of the works all the processing needed to calculate a pose is done at the device (Engelke et al. 2013). Only 14.0% use a remote server to assist in this task (Ventura et al. 2012) or completely perform pose calculation (Ha et al. 2011), as illustrated in Figure 2.7.

Table 2.3 lists relevant studies for each classification, those with more citations per year.

As for research type, approximately two thirds of the papers propose a new technique to perform tracking (Shanklin et al. 2011) and 28.6% use an existing method to develop a mobile solution that requires tracking (Polo et al. 2014), as can be seen in Figure 2.8.

2.3.2 Classification Trends

The annual trend per tracking type shows that until 2015 the number of papers about sensor-based systems is increasing. Moreover, the number of works that use a single sensor in 2015 is almost 5 times higher

Table 2.2: List of sensors and their most used combinations.

Sensor	Number of Papers	Combination of Sensors	Number of Papers
Accelerometer	137	Wi-Fi	40
Magnetometer	98	GPS	36
Wi-Fi	97	Accelerometer, Gyroscope and Magnetometer	26
GPS	96	Accelerometer and Gyroscope	22
Gyroscope	90	Accelerometer	11
Cellular Network (GSM, CDMA)	32	Accelerometer, Gyroscope, Magnetometer and Wi-Fi	11
Acoustic	19	Accelerometer and Magnetometer	9
Barometer	10	GPS and Wi-Fi	8
Bluetooth	10	Cellular Network	8
Depth	6	Cellular Network and GPS	7
Illuminance	3	Accelerometer, GPS and Magnetometer	7
Thermal	1	Accelerometer and GPS	6
Radio	1	Other 50 combinations	97

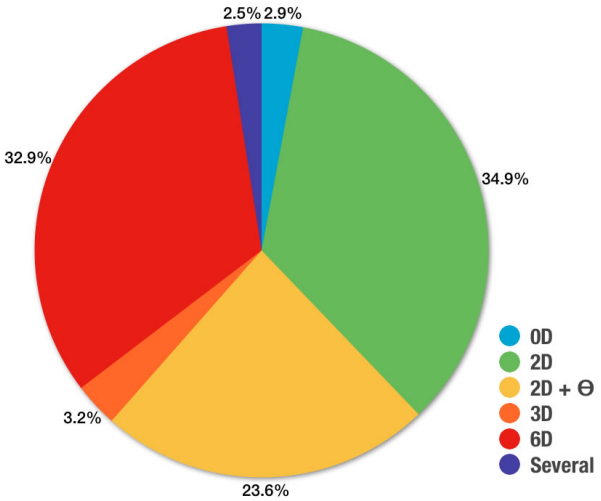


Figure 2.6: Degrees of freedom distribution over the database.

than in 2009 and 13 times larger for sensor fusion techniques, as can be seen in Figure 2.9. From Figure 2.9 (top) it is also possible to conclude that the other tracking types have an overall growing tendency. From 2009 to 2015 natural feature solutions went from 3 works to 8 with static model studies and 3 to 8 with dynamic model papers. On the same period, hybrid solutions went from 0 to 20 studies and it was the only tracking type that had more publications in 2015 than in 2014. The growth of marker studies occurred in the last three years.

Regarding the annual trend per degree of freedom, it is possible to see in Figure 2.10 an increasing number of publications about 0D, 2D, 2D + θ and 6D trackers. Respectively, they went from 0 to 5, 4 to 27, 1 to 22 and 7 to 27 between 2009 and 2015. The image also shows that the community did not demonstrate the same interest in systems with a 3D approach.

Most of the works use a local approach to calculate the device's pose and this fact is reflected in the annual trends per tracking platform, as shown in Figure 2.11.

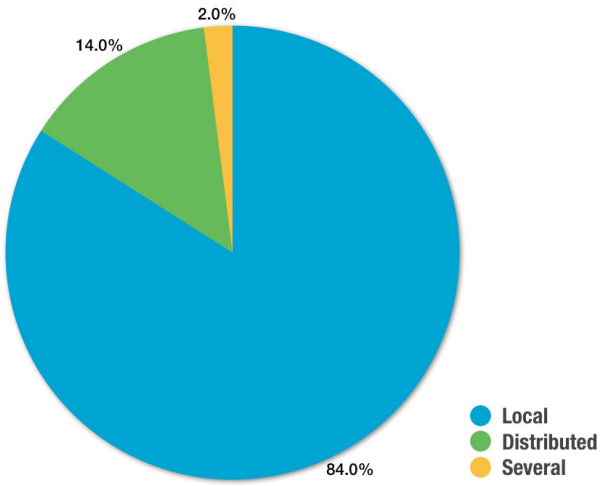


Figure 2.7: Tracking platform distribution over the database.

Table 2.3: Relevant papers for each classification.

Tracking Type	**Relevant Studies**
Hybrid	(Kurz et al. 2011) and (Ventura et al. 2012)
Single Sensor	(Shin et al. 2010) and (Gozick et al. 2011)
Sensor Fusion	(Chon et al. 2012) and (Zhang et al. 2013)
Static Model	(Wagner et al. 2010a) and (Hu et al. 2013)
Dynamic Model	(Klein et al. 2009) and (Wagner et al. 2010b)
Marker	(Oui et al. 2011) and (Gherghina et al. 2013)
Degree of Freedom	**Relevant Studies**
0D	(Rai et al. 2012) and (Xu et al. 2014)
2D	(Shin et al. 2010) and (Lv 2013)
2D + θ	(Schall et al. 2010) and (Shin et al. 2012b)
3D	(Li et al. 2013) and (Elloumi et al. 2013)
6D	(Takacs et al. 2010) and (Tanskanen et al. 2013)
Tracking Platform	**Relevant Studies**
Local	(Arth et al. 2009) and (Schöps et al. 2014)
Distributed	(Chen et al. 2009) and (Ventura et al. 2014)

2.3.3 Classification Relationship

The relationship between the classifications can provide a powerful and quick overview of tendencies on tracking for mobile devices. A bubble chart was used because it offers a more visual result than tables. Figure 2.12 presents a bubble plot in two dimensions in which the leftmost represents the tracking type by tracking platform and the rightmost displays the tracking type by degree of freedom. It should be noted in the first dimension that the ratio of publications of local systems is at least four times larger than the ratio of distributed approaches.

The same balance cannot be seen in the second dimension, in which the majority of the sensor works are location-based solutions, such as (Michael et al. 2013) that compute a 2D pose and (Jung et al. 2011) for 2D + θ papers. Only three publications present a system that computes a 6D pose using only a combination of the device's sensors. One example is (Robinson et al. 2012), in which the authors append a pico projector to a mobile device in order to make projective drawings on the wall. The approximated position is computed in a calibration step using the sensors, in which the user has to move the device according to a projected guide. All 112 2D works that use sensors to compute the pose are location-based systems, as well as all three marker

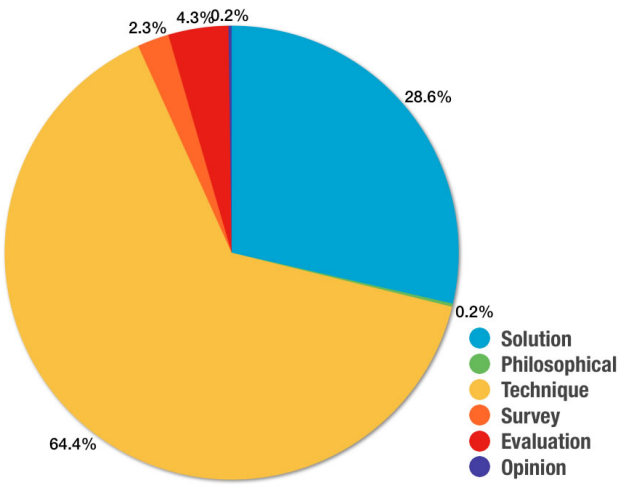


Figure 2.8: Research type distribution over the database.

papers, such as (Raj et al. 2013), seven of the hybrid works, like (Santos et al. 2013), and four of the static model studies, such as (Nguyen et al. 2010). All the other 29 works compute a 2D position on the screen, as in (An et al. 2011).

Single sensor and sensor fusion systems are the only two tracking types in which a 6D pose is not the most common information required. For all other tracking types at least 58.8% of the papers are about a system that calculates a full rotation and translation pose, as exemplified in (Issartel et al. 2014; Herling et al. 2012; Pirchheim et al. 2013; Kurz et al. 2012). Additionally, the dynamic model and sensor fusion techniques are the only tracking type that has at least one paper for every degree of freedom.

The bubble chart in Figure 2.13 presents the same dimensions of Figure 2.12. The difference is that it combines the vision-based and sensor-based techniques. Additionally, it also combines both location-based solutions. These combinations make more evident that most of the vision-based techniques calculate a 6D pose and that the majority of the sensor-based approaches are location-based services. Regarding the first dimension, it is possible to see that the ratio between the number of local and distributed solutions for each tracking type stays almost the same.

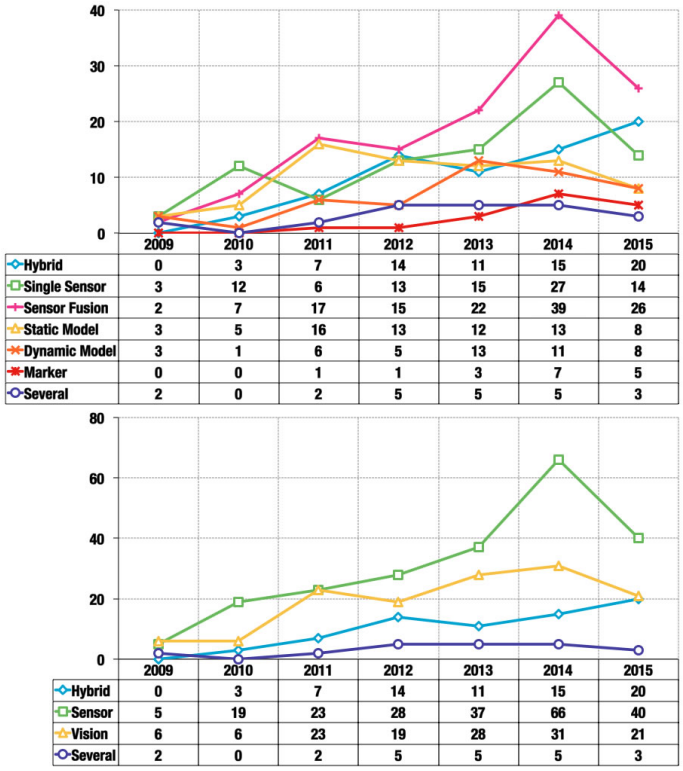
The relationship of degree of freedom by tracking platform is shown in Figure 2.14. The chart shows that for every degree of freedom category more than 80% of the publications are local. Moreover, all works that compute a 0D detection are local, such as (Ullah et al. 2012).

2.4 Discussion

Figure 2.4 shows that, even though the amount of publications in 2015 was smaller than in 2014, overall the number of papers about tracking on mobile devices is increasing over the years. There were almost 6.5 more works in 2015 than in 2009 and it is due to the improvement (Halpern et al. 2016) and popularization (Pew Research Center 2016) of such devices in recent years.

It is possible to see in Figure 2.9 (top) that there was an increase of more than two and a half times in the number of publications for all tracking types between 2009 and 2015. There is also a growth of vision-based works, as shown in Figure 2.9 (bottom). These data indicate that this type of tracking becomes possible with the improvement of the computational power of devices, especially for natural feature tracking.

Figure 2.9 also shows that sensor fusion tracking had the biggest growth in the analyzed period. It is also possible to note in Figure 2.5 that the majority of works use this type of tracking. Moreover, 49.1%



	2009	2010	2011	2012	2013	2014	2015
Hybrid	0	3	7	14	11	15	20
Single Sensor	3	12	6	13	15	27	14
Sensor Fusion	2	7	17	15	22	39	26
Static Model	3	5	16	13	12	13	8
Dynamic Model	3	1	6	5	13	11	8
Marker	0	0	1	1	3	7	5
Several	2	0	2	5	5	5	3

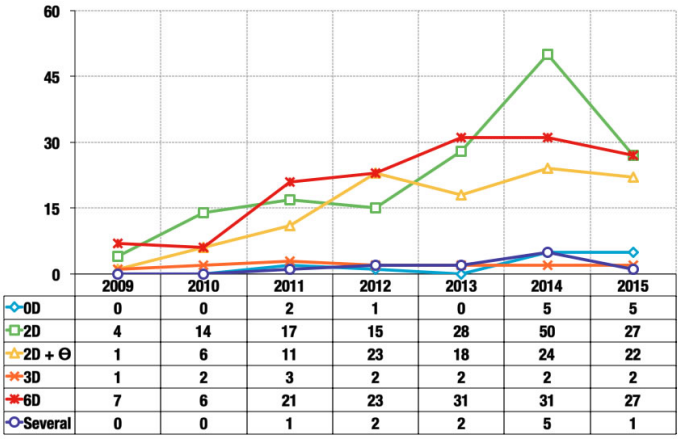
	2009	2010	2011	2012	2013	2014	2015
Hybrid	0	3	7	14	11	15	20
Sensor	5	19	23	28	37	66	40
Vision	6	6	23	19	28	31	21
Several	2	0	2	5	5	5	3

Figure 2.9: Annual trend per tracking type. Trends of all tracking types (top); yearly evolution of tracking types combining all vision-based and sensor-based techniques (bottom).

of all studies do not rely on the camera to perform tracking. This is probably because it is the most suitable approach to compute a pose for location-based solutions, which use 2D and 2D + θ information, and this type of solution is one of the most common type of application for mobile devices. This relationship is emphasized in Figure 2.12 and Figure 2.13. Nevertheless, it should be observed that only three works use only sensors to compute a full 6D pose because of their technical limitation, such as noise and error accumulation.

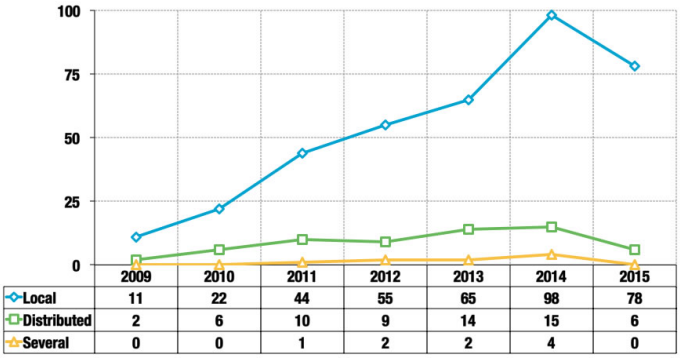
The analysis revealed that 41.3% of all sensor papers use data from only one sensor to compute the device's pose. The other 58.7% perform tracking using a combination of different sensors. This fusion of sensors is important because it allows using the data from one sensor to overcome the weakness of another one. Moreover, all studies that use a single sensor are 2D or 2D + θ, as can be seen in Figure 2.12. The nine papers that use sensors to compute a 3D or 6D pose require a combination of them in order to perform tracking. As seen in Table 2.2, the two most common sensors, accelerometer and magnetometer, are popular because they can be used in combination with other sensors in both indoor and outdoor situations since they do not require any external infrastructure, such as access points. The two sensors that follow them are related to providing the device's position. Wi-fi is widely used to compute indoor position. Although noisy, GPS is a great way to determine outdoor localization.

Figure 2.13 shows a clear trend that relates sensor techniques to location-based systems and vision-based approaches to solutions that require a 6D pose. Moreover, it is possible to see in Figure 2.5 that static model tracking is the favorite among natural feature-based approaches. However, there is a significant amount of systems that use a dynamic model technique. One reason is that some works use learning algorithms to calculate a pose, such as (Lambrecht et al. 2013). These techniques demand a massive processing power in the offline training phase that can be performed previously in a computer but does not use much processing for tracking, which makes them more suitable for mobile devices. One advantage of these techniques is that the trained model is usually refined using the tracking results.



	2009	2010	2011	2012	2013	2014	2015
0D	0	0	2	1	0	5	5
2D	4	14	17	15	28	50	27
2D + Θ	1	6	11	23	18	24	22
3D	1	2	3	2	2	2	2
6D	7	6	21	23	31	31	27
Several	0	0	1	2	2	5	1

Figure 2.10: Annual trend per degree of freedom.



	2009	2010	2011	2012	2013	2014	2015
Local	11	22	44	55	65	98	78
Distributed	2	6	10	9	14	15	6
Several	0	0	1	2	2	4	0

Figure 2.11: Annual trend per tracking platform.

There was a huge increase in the number of publications of 2D + θ works, which had a growth of 22 times between 2009 and 2015, as shown in Figure 2.10. Papers with 2D and 6D systems grew almost 7 and 4 times in the same period, respectively, which is also a considerable value. All the other categories present a consistently small number of papers throughout the years. Moreover, Figure 2.6 shows that 2D techniques are the most common ones. The main reason for this is that there is a high demand for location-based applications and the amount of publications reflect this. 6D systems are also very popular because it is the most traditional information required for tracking, especially for augmented reality systems. The fact that the majority of papers of every tracking type except single sensor and sensor fusion calculate a 6D pose reflects the importance of computing the rotation and translation of the device relative to the real world, as observed in Figure 2.12. This approach is interesting because it combines the benefits of both vision and sensors to perform a more accurate and robust tracking. It is also possible to see in Figure 2.6 that 0D and 3D approaches are the least common. This is due to the fact that there is a small number of applications that require a 0D or 3D pose.

Even with the devices' limitations, Figure 2.7 shows that the majority of works execute all the steps to compute the pose at the device. One reason is the lack of a good communication infrastructure to transfer the data to a remote server. However, it can be noted in Figure 2.11 that more than half of the distributed works were published in the last three years. This can be an indication that there is a recent improvement in the network infrastructure and researchers are exploring the use of a remote computer, which provides more resources than the mobile device, such as processing power, memory and storage space. Another reason is the possibility of using sensors and cameras that are not available in the device (Bai et al. 2013).

Figure 2.12 indicates that there is no relationship between the tracking type and the execution platform since the proportion of local and distributed works varies little per tracking type. However, the same proportion

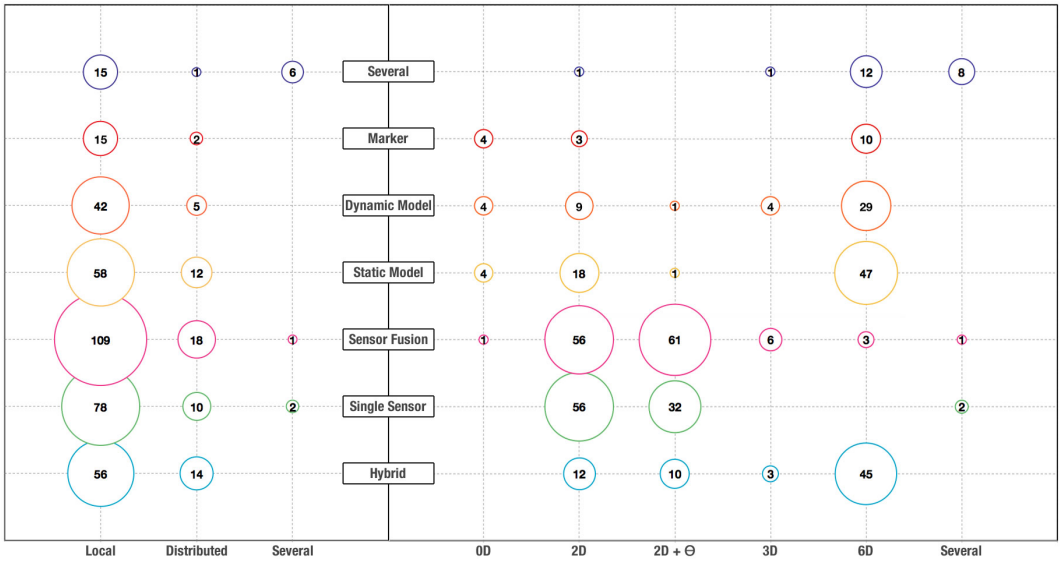


Figure 2.12: Two dimensional bubble chart: left side presents the tracking type by tracking platform and the right side presents the tracking type by degree of freedom.

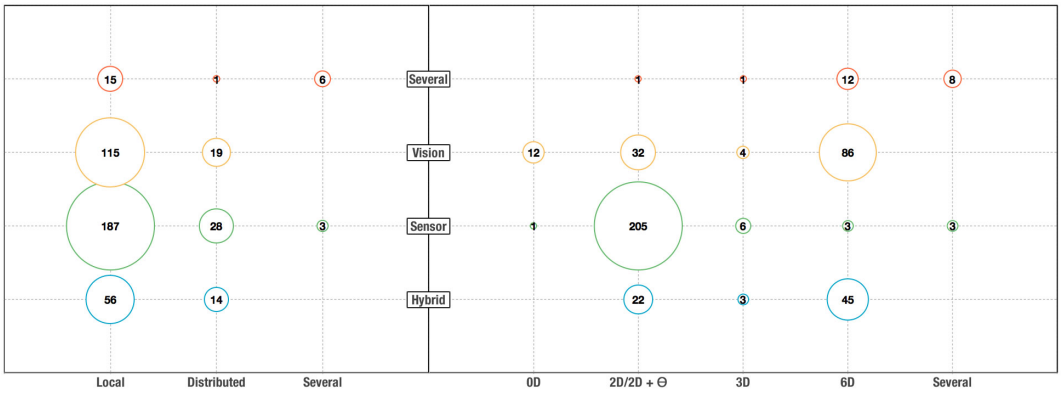


Figure 2.13: Two dimensional bubble chart: left side presents the combined tracking type by tracking platform and the right side presents the combined tracking type by degree of freedom with location service systems combined.

is not seen when relating tracking platform with degree of freedom. It is possible to see in Figure 2.14 that a few more than 80% of 6D papers are local while none 0D and only one 3D study is distributed.

As seen in Figure 2.8, the majority of the papers propose a new technique and there are also several works that use an existing method to create a solution for an open problem. These two research types represent 93% of all studies classified. This is an indication that the demand for systems that use tracking is high. More than that, it is a clear suggestion that the field of tracking for mobile devices still has a lot of open problems to tackle.

2.4.1 Implications for Future Studies

This mapping study not only offers useful information for researchers who are interested in the existing works regarding tracking for mobile devices but also identifies gaps in this research topic.

Most of the works calculate the pose using devices' sensors or computer vision algorithms. However, there is a tendency to combine both approaches to provide a more robust tracking. It can be noticed in Figure 2.9 that no hybrid work was published in 2009 and in 2015 the 20 papers that use this type of tracking represent 23.8% of the studies in this category. It is the highest percentage in the evaluated period. One reason

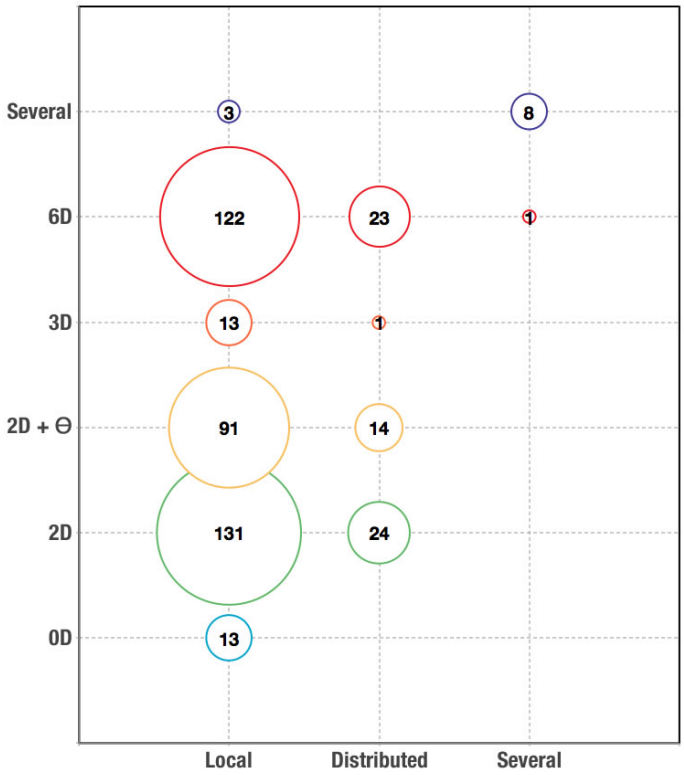


Figure 2.14: One dimensional bubble chart of degree of freedom by tracking platform.

is the improvement and miniaturization of more complex sensors and cameras. Following this tendency, Apple and Google launched their augmented reality platform at the end of 2017, namely ARKit (Apple Inc. 2017) and ARCore (Google Inc. 2017), respectively. The tracking algorithm on both SDK relies on hybrid techniques. They use the camera image to extract and track natural features and to perform loop closure, while the inertial sensors are responsible to recursively update the device position.

Another sensor that achieved this level of maturity regarding miniaturization and integration into mobile devices is the depth camera. Google's Tango is another augmented reality platform that combines different types of sensor to perform tracking (Google Inc. 2014). However, different from ARKit and ARCore that only use sensors available in most mobile devices, Tango also incorporates a depth and a fisheye camera. These two cameras play an important role in the development of accurate and robust tracking techniques. As a result, other kinds of devices, such as DAQRI Smart Helmet (DAQRI 2016) and Microsoft Hololens (Microsoft 2016), are embedding depth sensors to improve tracking.

This mapping found a few studies focusing on the use of machine learning approaches to compute the pose. But this is a prominent research area because such algorithms learn what are the best features to be used for tracking (Tan et al. 2014). Moreover, as mentioned before, learning techniques transfer most of the computational effort to an offline training phase while the tracking itself demands few processing resources, which makes them suitable for mobile devices. Machine learning is a mature area and its use for tracking is increasing rapidly. Although, there are still several open problems in the area.

Recent improvements in communication networks enable the increasing number of works that use distributed approaches, as shown in this study. In the future, this infrastructure will probably be more reliable and faster (The Next Generation Mobile Networks Ltd. 2015), which creates new opportunities to perform tracking on remote servers, using the mobile device only to capture the input information and display the output results. Moreover, this connectivity is a basic requirement for creating sensitive environments using smart objects. Each one of these connected sensors can be aware of its location and may be used to share this

information with a mobile device and perform fully distributed tracking. For instance, smart objects spread over an indoor place can be used to provide or improve the indoor localization of a person using a smartphone connected with them.

It is also important to be aware of the improvements of the hardware capabilities that will be available on mobile devices in near future. New tracking techniques can be proposed or existing ones adapted taking into consideration the use of multiple cores of the device's processor and graphics processing unit (GPU). Beyond that, it is possible that several mobile devices will have chips dedicated exclusively to execute embedded computer vision algorithms, such as Qualcomm's Hexagon digital signal processor (DSP) (Qualcomm Technologies, Inc. 2015). These dedicated chips will allow tracking to be performed faster while consuming less energy.

A research topic that did not appear in the mapping was the inclusion of any kind of semantic in tracking systems, even though it is an important research problem in computer vision. One reason could be that extracting any type of knowledge from the scene demands more resources than a mobile device can handle in a practical time at that point. However, recent tracking techniques for augmented reality, such as ARCore and ARKit are able to identify planes to anchor the augmented content, which can be the floor, a table or a wall. With this simple knowledge of the environment, it is possible to improve rendering, such as adding more natural shadows.

3
Preliminary Experiments

The systematic mapping provided valuable theoretical knowledge regarding tracking for mobile devices. These lessons learned were used to elaborate a set of preliminary experimental scenarios aiming to obtain a practical know-how about the specificities of developing tracking techniques for mobile devices. This chapter describes these experiments and discusses their results and findings. The first one evaluates the Google Tango platform to establish a reference of the state-of-the-art trackers. Additionally, another experiment tests the use of parallelism, distributed approach and native implementation in order to find an efficient architecture to execute computer vision algorithms on mobile devices. After that, it is shown the experiments to test different tracking techniques that the systematic mapping indicated to be suitable for mobile devices. One is a face tracking technique using machine learning and local binary features. The other one is a SLAM technique that was developed in desktop and ported to a Tango tablet device.

3.1 Evaluation of Tango Platform

Tango (Google Inc. 2014) is a platform that combines computer vision techniques with state-of-the-art sensors, allowing to perform 6DoF tracking on mobile devices. Even with the release of ARCore and ARKit, Tango is still arguably the best tracker available for them. Mainly because it uses cameras that are not available on most of the handheld devices, such as depth and fisheye ones. While the fisheye camera widens the field of view, which increases the number of characteristics, the depth camera provides a good approximation of the scene structure and scale. These additional data help achieving a more precise and stable tracking.

Therefore, it is relevant to evaluate its accuracy. As far as the author knows, so far no study evaluates the Tango platform. This experiment is important because it provides expertise on how to assess tracking systems and also to find scenarios and situations in which they work well and fail. This section aims to evaluate the precision of both motion tracking and depth sensing technologies of the Tango platform. The evaluation method proposed and the results were discussed in (Roberto et al. 2016a), and details are provided in the next subsections.

3.1.1 Tango Platform

As mentioned, the devices that support the Tango platform have some features that provide new ways to navigate in different environments. They use technologies such as motion tracking and depth perception. This subsection explains how these technologies work on the Tango platform as well as describes the characteristics of the sensors available.

Motion tracking means that a Tango device can track its own movement and orientation through 3D space. Tango implements motion tracking using visual-inertial odometry to estimate where a device is relative to where it started. Standard visual odometry uses camera images to determine a change in position by looking at the relative position of different features in those images. Visual-inertial supplements visual odometry with

inertial motion sensors capable of tracking a device's rotation and acceleration. This allows a Tango device to estimate both its orientation and movement within a 3D space with even greater accuracy. Unlike GPS, motion tracking using visual-inertial odometry works indoors. In addition to the gyroscope and accelerometers, Tango uses a wide-angle motion tracking camera to add visual information, which helps to estimate rotation and linear acceleration more accurately. To perform motion tracking, the Tango APIs provide the position and orientation of the user's device in full six degrees of freedom. The data is returned with two main parts: a vector in meters for translation and a quaternion for rotation.

Depth Perception gives an application the ability to understand the distance to objects in the real world. Tango tablet implements depth perception with time-of-flight (ToF) technology, which requires the use of an infrared projector and an infrared sensor. Tango tablet depth sensor is designed to work best indoors at moderate distances (0.5 to 4 meters). This configuration gives good depth at a distance while balancing power requirements for infrared illumination and depth processing. It may not be ideal for close-range object scanning or gesture detection.

The Tango APIs provide a function to get depth data in the form of a point cloud. This format gives (x, y, z) coordinates for as many points in the scene as are possible to calculate. Each dimension is a floating point value recording the position of each point in meters in the coordinate frame of the depth-sensing camera.

3.1.2 Evaluation Methodology

Evaluating tracking systems is a challenging task (Tamura et al. 2009). Many efforts have been made in the past years to provide metrics and standards to analyze the aspects related to this problem (Petit et al. 2011; Roy et al. 2015; Hodaň et al. 2016). The main reason is the difficulty to find the ground truth to compare with the obtained results. There are several benchmark datasets available aiming computer vision tracking systems evaluation, each one having many purposes and providing several types of input data. For instance, (Lieberknecht et al. 2009) presents an image dataset to evaluate planar model-based techniques, (Shibata et al. 2010) describes a benchmark to measure the quality of 3D model-based algorithms and (Sturm et al. 2012) provides RGB and depth information aiming SLAM systems. All of them also provide the expected results, which are used to assert the precision of the algorithm. However, it is hard to use these datasets on mobile devices. One reason is the difficulty to extract information from different sensors since they are noisy and the precision varies among devices. Therefore, it was proposed a different methodology to evaluate the motion tracking and depth sensing functionalities in the Tango Platform. The selected device was the Yellowstone tablet, which provides all the Tango functionalities.

3.1.2.1 Motion Tracking

The evaluation method is based on moving the Yellowstone tablet between two known positions in the real world. Then, comparing the distance between these positions estimated using the device with the ground truth value. This way, it is possible to evaluate the error the system accumulates during motion tracking from a starting point to an ending position. It was used a graph paper with the precision of one millimeter to ensure the experiment accuracy. The paper was glued to a table so it does not move during the tests. A needle was attached to the base of the Yellowstone tablet in order to have the exact position of the device over the paper. Figure 3.1 shows the setup. It was designed two different experiments based on this setup, one to evaluate a small augmented reality workspace and another one for large environments.

For the first one, the idea is to evaluate how the Yellowstone tablet works on a small workspace, which for this study is a table with an area up to one square meter. Therefore, the device is positioned with the needle on the origin of the graph paper and their axes are aligned. The tablet will be moved freely and placed in any

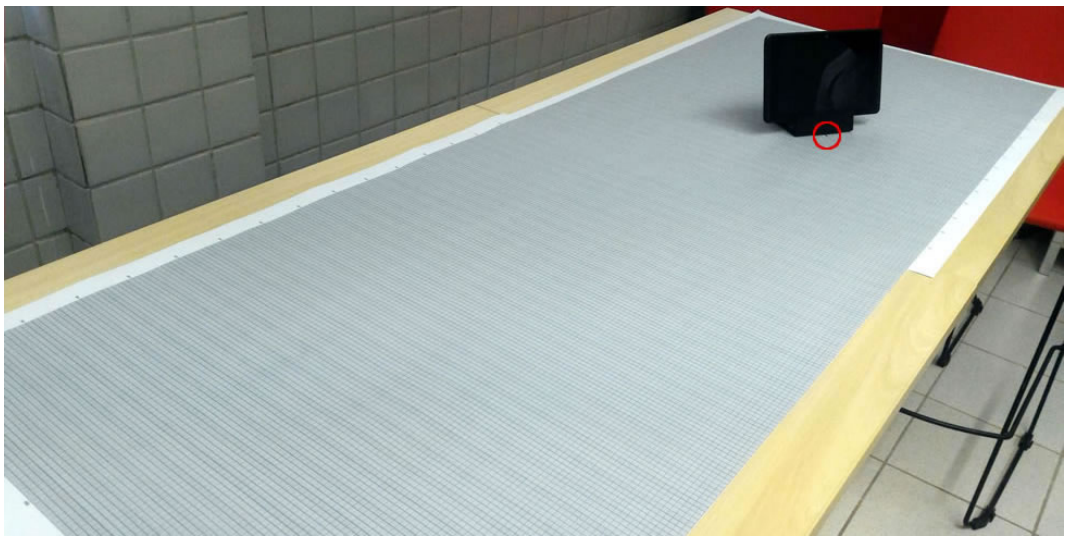


Figure 3.1: Evaluation setup consists of a graph paper with precision of one millimeter and measuring 1.5 x 0.55 meters. Red circle highlights the needle used to get the exact position on the paper.

other position on the graph paper. The error is the difference of the Euclidean distances between the origin and the final position computed on the Tango device and measured with the graph paper.

Regarding the large environment, the goal is to measure how the Tango device behaves when performing motion tracking on places such as a regular office and outdoors. The office is a closed room that has an approximate area of 50 square meters and artificial illumination. As for the outdoor experiment, it was placed in two different courtyards measuring around 100 square meters each. It was also located in the corridor of a building that is open to the outside. All the outdoor measurements were collected using natural illumination during daylight.

It is not possible to have a graph paper that is large enough to cover the entire office or the courtyards. Thus, for this experiment, the device is also positioned with the needle on the origin of the graph paper, their axes are aligned and it is moved freely in these environments. The difference from the previous experiment is that the tablet is returned to the same position where it started. The error is also the Euclidean distance between the position computed on the Tango device after finishing the movement and the initial position.

3.1.2.2 Depth Sensing

Regarding the depth sensor, it was evaluated the accuracy of the 3D points positions obtained from it. The process consists of calculating the Euclidean distance between 3D points reconstructed from the color camera and corresponding ones from the depth camera. The Tango API provides the registration between color and depth cameras. The intrinsic parameters used are the ones from the manufacturer calibration. The 3D points of the color camera are the inner corners of a detected chessboard pattern whose pose is estimated using the Direct Linear Transformation (DLT) method and refined by minimization of reprojection error (Hartley et al. 2003).

Since the depth image generated by the Tango device has a lower resolution when compared to the color image, it has to be upsampled when obtaining the corresponding depth measure of a chessboard corner. Both nearest-neighbor and bilateral interpolation (Tomasi et al. 1998) were evaluated for performing this task.

3.1.3 Results

In order to evaluate the motion tracking capability of the Tango platform, it was used a sample application available on the project GitHub[1] that uses both Tango Area Learning and Motion Capture features and is called “C++ Augmented Reality Example”. This application uses the fisheye camera and the device gyroscope and accelerometer to compute its pose relative to its initial position.

3.1.3.1 Motion Tracking

For the experiment on the small workspace, the Yellowstone tablet was moved freely to any other position over the graph paper. To have statistical power, the sample size for this experiment was calculated aiming 95% confidence within 1 centimeter precision (Jain 1991). Therefore, these measurements were repeated 67 times to ensure that.

Figure 3.2 shows the error dispersion, in which the smallest was 0.009 meters and the largest was 0.181 meters. On average, the error was 0.067 $\pm$ 0.040 meters.

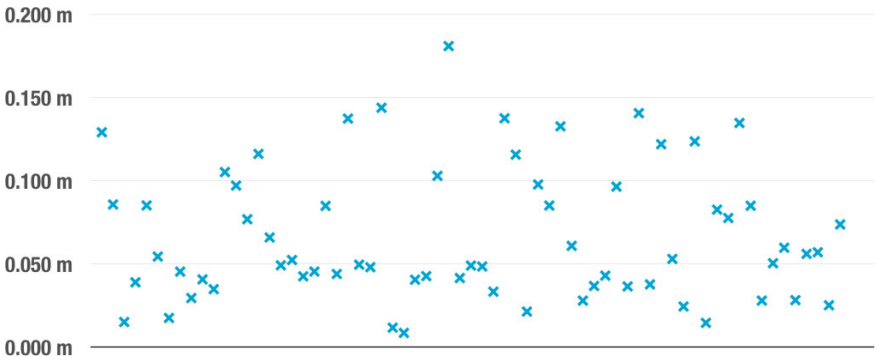


Figure 3.2: Error dispersion for the small workspace experiment.

Figure 3.3 shows the device’s position distribution over the graph paper during the experiment.

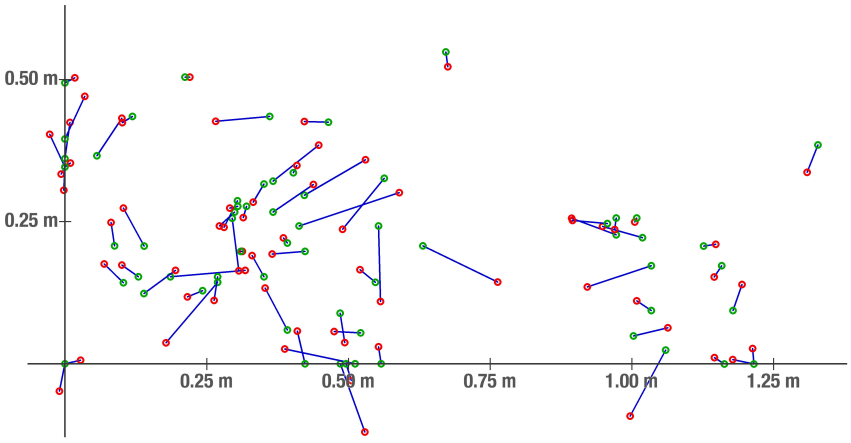


Figure 3.3: Distribution of the device positions on the graph paper (green) and their correspondent positions calculated by the Yellowstone tablet (red).

Regarding the evaluation of the motion tracking on large environments, the error is the Euclidean distance between the initial and the final position calculated by the device after moving it freely in this environment and returning to the same location. After a few measurements, it was noted a large difference

[1] https://github.com/googlesamples/tango-examples-c

among the errors from the indoor and outdoor environments. Therefore, it was decided to perform two different evaluations, one for each situation. The number of samples to ensure statistical power emphasize this decision. For the indoor experiments, it was necessary to have 31 samples to have 95% confidence within 2 centimeters precision, which is the double of the small workspace because the covered area was much larger. On the other hand, it was not possible to have such confidence in the outdoor experience. The reason is that the error variation is so high that it would be necessary to have more than 5000 samples to have 95% of confidence within 2 centimeters precision.

Figure 3.4 shows the error dispersion in the large indoor scenario. The smallest one was 0.049 meters while the largest was 0.261 meters. On average, the error of the 45 samples measured was 0.142 $\pm$ 0.057 meters. Also, the average distance walked with the Yellowstone tablet was 23.608 $\pm$ 7.892 meters.

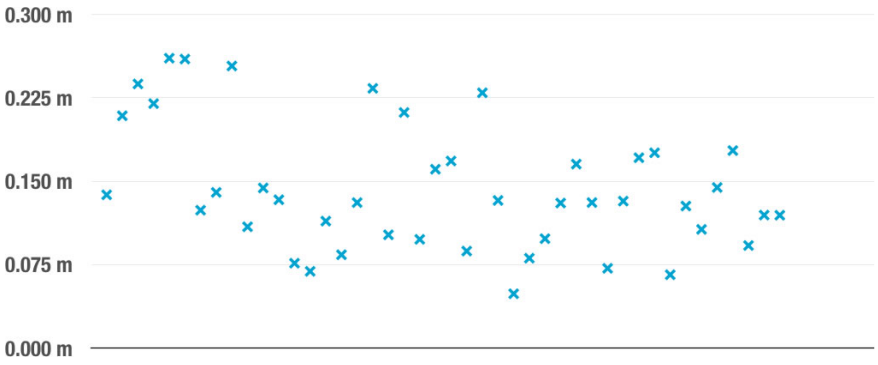


Figure 3.4: Error dispersion for the large indoor environment experiment.

Figure 3.5 illustrates one of the paths walked with the Yellowstone tablet and the difference between the initial and final positions.

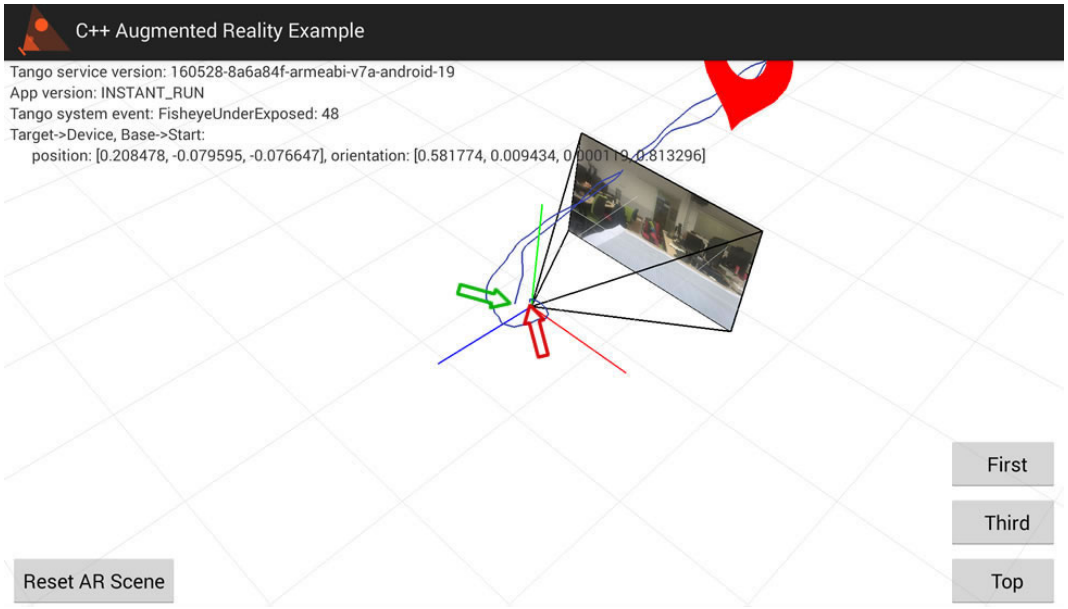


Figure 3.5: Screenshot of one of the paths computed using the Yellowstone tablet. The green arrow points to the initial place and the red one to the final position calculated after a free walk. The error is the average Euclidean distance between them.

Regarding large outdoor environments, it was performed 21 repetitions. However, this amount was not enough to have statistical power. Although, the average error of 0.905 $\pm$ 0.753 meters indicates that precision of the Yellowstone tablet is much smaller when it is dealing with natural illumination and wide spaces.

3.1.3.2 Depth Sensing

In the first experiments, the chessboard pattern was printed on a paper using black ink. However, the dark squares did not reflect infrared light in a way that would allow robustly estimating the depth of the chessboard corners. Due to this, it was used a mix of cyan, magenta and yellow ink in order to have dark squares that are correctly scanned by the depth camera. This aspect is illustrated in Figure 3.6. The chessboard was printed on A4 paper with a square side of 28 millimeters.

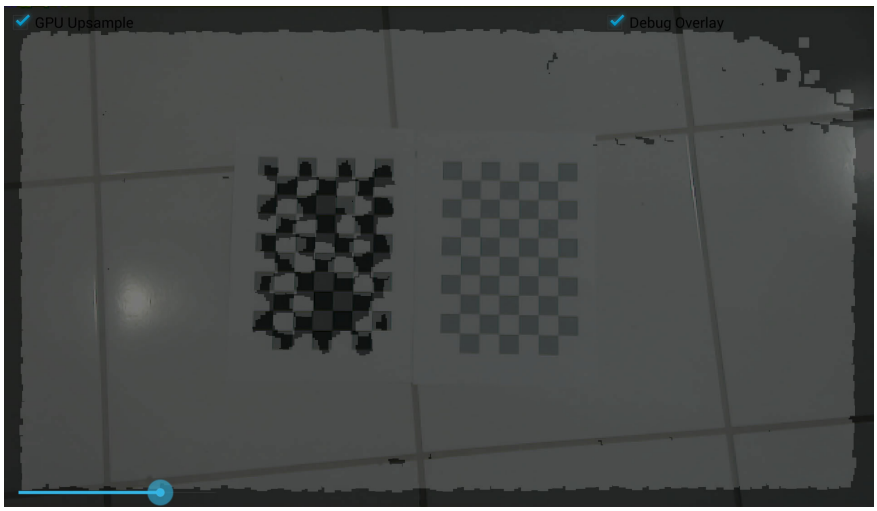


Figure 3.6: Screenshot of the depth estimation of two chessboards printed on a paper. On the right, the one printed with a mix of cyan, magenta and yellow inks. On the left, the same pattern printed with a black ink. Note that the sensor is not able to estimate depth on the black squares of the left paper.

Figure 3.7 shows the mean depth estimation error considering different distances between the device and the chessboard pattern and different depth interpolation strategies. In order to obtain error values accurate within 0.1 millimeters at 95% confidence, 150 samples were collected for each configuration. The average execution times of the nearest-neighbor and bilateral depth interpolation procedures for each point were 0.696 $\pm$ 0.127 and 1.881 $\pm$ 0.312 milliseconds, respectively.

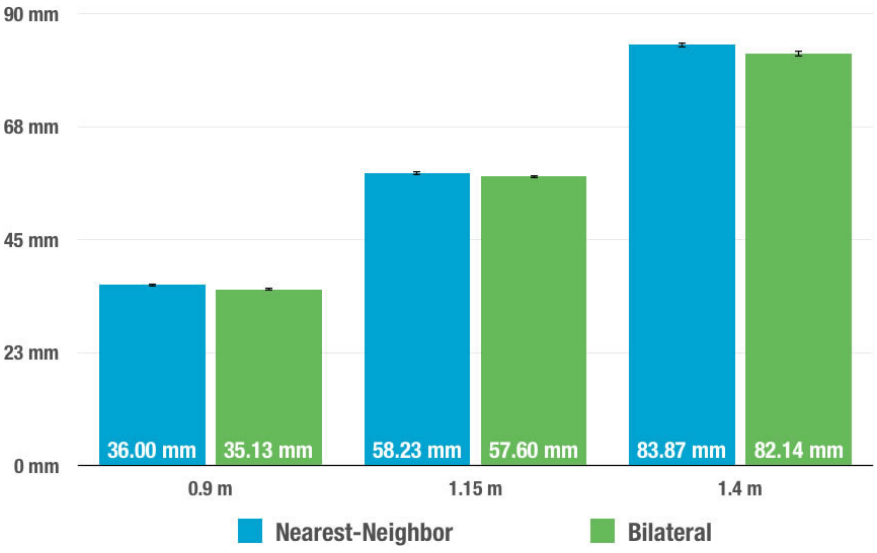


Figure 3.7: Mean depth estimation error with respect to distance between Tango device and chessboard pattern using different depth interpolation methods.

3.1.4 Discussion

The results showed that the motion tracking of the Yellowstone tablet is 2.3 times more precise on a small workspace than on large indoor environments. However, the presented errors can have an impact on the user experience. An average error of 6 centimeters is often noticed in a small workspace.

On the other hand, even having a bigger error when dealing with large environments, the precision of the motion tracking on indoor spaces is suitable to provide a good user experience for several kinds of augmented reality applications. However, for scenarios in which it is necessary to have accuracy, this error can harm the user experience. Figure 3.8 (left) shows an example where Yellowstone tablet measure tool is calculating the width of a 0.7 meter door. After moving the device for a few steps away from the door and back, the ruler is placed in a different position, as seen on the right side.

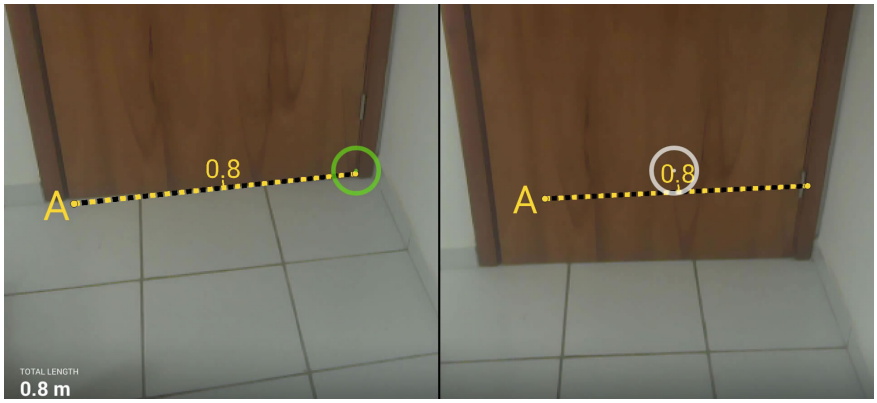


Figure 3.8: Door width estimation using the Yellowstone tablet. Left side shows the initial measurement and the right side shows the ruler position after moving the device. Door actual width is 0.7 meters.

During the experiment, it was noticed that the algorithm that Tango uses for motion tracking seems to mistrust the sensors it uses regarding their precision. There is an indication that it has a much stronger confidence in the information provided by the fisheye camera. For instance, sometimes the device was left standing still over the table and when some object moves in front of the camera, the motion tracking algorithm calculates that the tablet was moving in the opposite direction.

The tests on large outdoor environments could not provide results with statistical power because there was a significant variation on the error measured on every sample. However, this disparity suggests that the Yellowstone tablet has some issues to deal with outdoor illumination and wide spaces. It is emphasized by the fact that no result of the outdoor measurements presented a smaller error than any indoor sample. Moreover, in some cases, the error was greater than 1.0 meter and in the worst case it reached more than 3.0 meters.

Regarding depth sensing, the mean errors presented by Yellowstone tablet were similar to the ones obtained with a desktop depth sensor. The results also suggest that the depth estimation error increases linearly with respect to the distance between the device and the object. Bilateral depth interpolation provided an average precision improvement of 1.82% (1.08 millimeters) with respect to the nearest-neighbor approach. However, it was more than 2.5 times (1.18 milliseconds) slower on average for each point.

3.2 Efficient Tracking on Mobile Devices

Computer vision tracking is a task that demands many computational resources. It may be necessary a lot of processing and memory to extract and describe natural characteristics from a scene or to correctly match them with the features from a model. As the tracking system runs for an extended period of time, the

number of features may grow, requiring more memory space, both ROM and RAM.

Since mobile devices have limited resources, it is necessary to investigate the best approach to minimize these constraints. This section aims to explore different ways to implement a computer vision tracking system in order to find the best trade-off between processing and memory. The results found in this experiment were published in (Lima et al. 2015), and details are given in the next subsection.

3.2.1 Android Architectures for Computer Vision Tracking

After analyzing some possibilities to implement an Android system to perform computer vision tracking, two of them appeared to have the potential to combine high performance with low memory consumption. The first one is a Multi-Thread Partial Native implementation, which benefits from using the different cores of the device processor as well as the performance gain of having native code for some tasks, which is known to be faster than Java implementations. The second one is a Client/Server approach that uses the processing power of a remote server to perform the most demanding processing tasks. During their development, another method aroused, which leaves only the necessary functions on the Java side while transfer most of the computation to the native part of the code and is called Full Native implementation.

A simple static model tracking technique was designed in order to provide a quick prototype of these approaches. The first step is to build the model. In order to do so, ORB features (Rublee et al. 2011) are extracted from a loaded image that will work as a planar template. After that, a z-coordinate is given to the features, so each one now has a corresponding 3D point. Then, it is computed their ORB descriptors. The planar static model that will be tracked is represented by the data structure that stores the descriptors and the 3D points of the features extracted from the image.

After creating the static model, it is possible to start tracking this template. First, ORB features are extracted from the frame captured by the device's camera and their descriptors are computed. Then, they are matched with the model's descriptors using a nearest neighbor search approach that finds the Hamming distance between descriptors to determine the best correspondence candidates between these groups of features. However, it is common to have several wrong matches and it is necessary to filter them to improve accuracy. One effective way to do that is by comparing the distance between the closest and the second closest neighbors. This strategy works because correct matches have the nearest neighbor much closer than the second closest one, which is incorrect (Lowe 2004). After filtering the good matches, the camera rotation and translation are estimated using EPnP (Lepetit et al. 2009). RANSAC (Fischler et al. 1981) is also used to reduce the influence of outliers, which are spurious matches that could also remain after the filtering process. This process is repeated until the application finishes.

3.2.1.1 Multi-Thread Partial Native Implementation

This implementation parallelizes part of the tracking steps described above, which are: feature extraction and description, matching with model keypoints, and filtering for good matches.

After building the static model, the tracker enters the main loop. This implementation uses a port of OpenCV (Bradski 2000) to this mobile platform named OpenCV4Android. The library provides the data structures and computer vision functions to perform the tracking task. OpenCV4Android is an interface that provides access to almost every OpenCV function in Java. Additionally, the original C++ version is also available to be used on native developments.

This implementation uses both versions. First, the current frame is captured using a camera interface provided by OpenCV and a Mat structure stores it. The Native part of the system receives this image, which is divided into equal parts according to the number of cores available in the device. Each core will process

one part of the frame in parallel. Intel TBB library (Intel Corporation 2016) has a version that is compatible with Android and was used on this version of the tracker. Then, in parallel, each core extracts features from one part of the image, matches them with the model's keypoints and filters to select the good matches. After every core finishes their tasks, the good matches are combined in only one structure and it is used to compute the device rotation and translation. The augmented reality content is drawn over the original image, which is returned to the Java part of the code so the rendering structure provided by OpenCV can display the modified frame on the screen. Figure 3.9 illustrates this implementation.

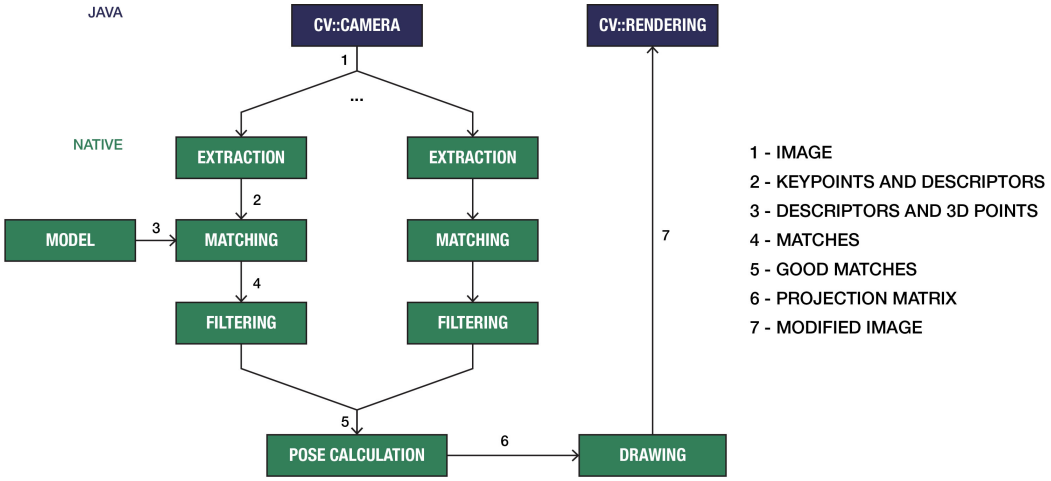


Figure 3.9: Multi-Thread Partial Native flow diagram.

3.2.1.2 Client/Server Implementation

This architecture aims to transfer part of the tracking processing to a remote server, which has more computational power to perform this task quicker than on the mobile device. Several tasks can be assigned to be executed at the server. On one hand, it is possible to implement all the tracking pipeline remotely, but it will demand an excellent network infrastructure to transfer the frames in real-time on both directions. On the other hand, the server can execute only selected tasks, which will decrease the network requirements. However, the server processing and memory resources will not be fully exploited.

For this implementation, every tracking step that depends on the image will be processed on the device in order to decrease the dependency of a good network infrastructure. Therefore, as seen in Figure 3.10, the current frame is captured using a camera interface provided by OpenCV and ORB is used to extract features and compute their descriptors. These information are encapsulated and sent to the server through a wireless local network of 54 Mbps, which is 35% faster than the fastest 4G network available (OpenSignal, Inc 2016). The server receives this data, which is hundreds of time smaller than the full image, and matches it with the model sent to the server during the initialization. Then, it filters these matches to select the good ones and computes the object rotation and translation. These two vectors are sent back to the device that draws the augmented content over the captured frame and renders it on the screen.

3.2.1.3 Full Native Implementation

It has become clear during the development of these approaches that capturing and rendering a frame using OpenCV functions is not efficient. The alternative found was to use camera structures from Android to capture a frame buffer and render it with OpenGL (Khronos Group 1997).

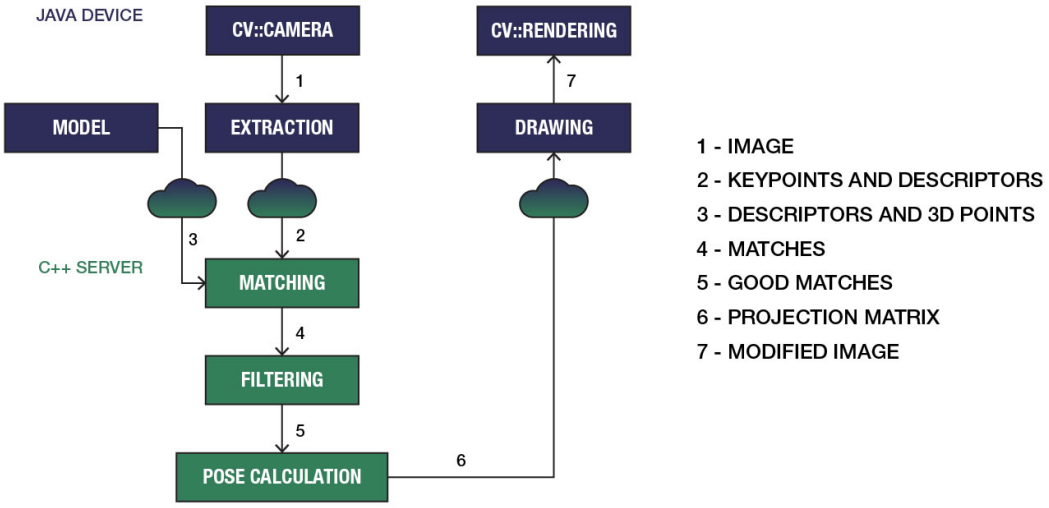


Figure 3.10: Client/Server flow diagram.

The problem with this approach is that Android camera and rendering structures work only with YUV color space while the OpenCV library uses the RGB color space (Kuehni 2003). In other words, each frame buffer received from the Android camera preview is in the NV21 format, which is the standard picture format on Android camera preview. This way, it was necessary to convert the color space of the image buffer on each frame, and a better approach to do so was to implement a parallel loop rather than using a sequential one. Additionally, it was necessary to implement methods that use OpenCV functions to render on YUV images like it was an RGB frame.

This way, every part of the tracker that uses OpenCV is implemented with native code. Therefore, the frame is captured, stored in a byte array and then sent using Java Native Interface (JNI). The YUV frame buffer is converted to RGB to be tracked sequentially. The augmented content is drawn and the image is converted to a byte array so it could be sent to the Java layer in order to be rendered, as shown in Figure 3.11.

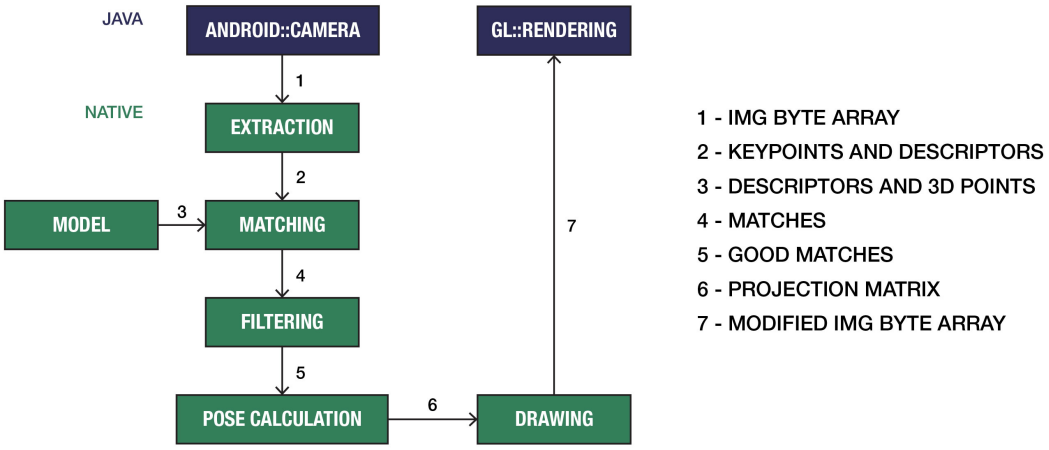


Figure 3.11: Full Native flow diagram.

Another benefit of this architecture is that it eliminates the necessity of having OpenCV Manager. This software has to be downloaded from Google Play Store and it provides all the dependencies necessary to execute applications that use the Java version of OpenCV4Android. When programming only with the C++ part, Android compiles all the dependencies and embeds them in the final application package.

3.2.2 Architectures Evaluation

The solutions were compared regarding performance and memory used, both RAM and ROM. Additionally, it was also developed a solution in which all the tracking is performed in Java. This solution was set as a base for comparison with the other implementations. For all of them, the evaluation method consisted in tracking the static model in the real world to project virtual lines over the template board, as seen in Figure 3.12.

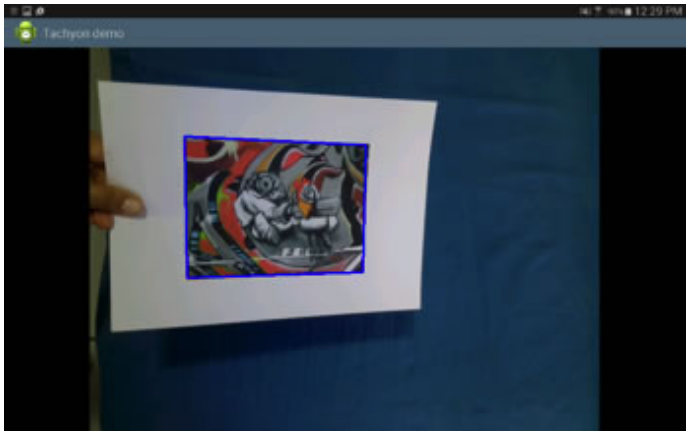


Figure 3.12: Screen capture of the system tracking the template. The blue square is the virtual content that is placed on top of the model.

The device used for evaluation was a Samsung Galaxy Note 10.1, which has a 2.3GHz quad-core processor, 3GB and 32GB of RAM and ROM memory, respectively. Figure 3.13 shows the results of running the tracker with each implementation.

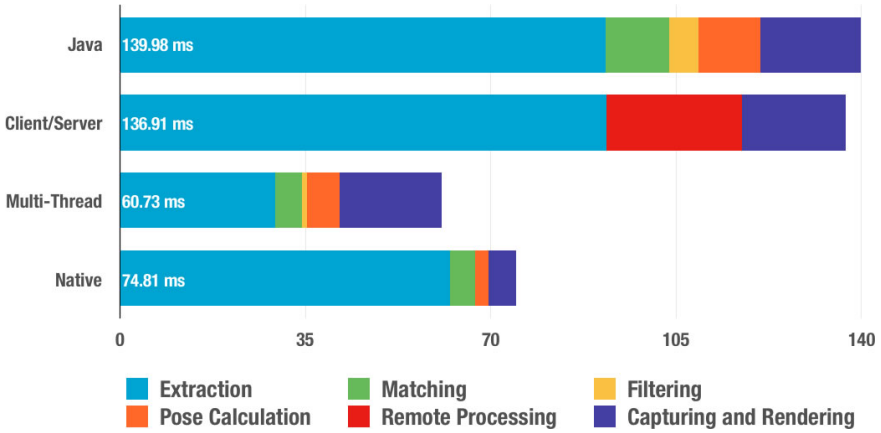


Figure 3.13: Execution time in milliseconds on every tracking stage for each implemented architecture. For Client/Server approach, feature matching, matching filtering and pose calculation also includes the time to transfer the data to the server and back to the device.

On average, Java implementation presented the worst results, being more than two times slower than the Multi-Thread version. It is clear that the bottleneck for every implementation is the feature extraction step. In the Full Native version, this stage consumes 83.3% of the total processing time. As expected, parallelizing this task provides a good speedup, more than three times when compared to a Java implementation.

In the other tracking phases, there is a significant speedup on using native code when compared with Java. However, parallelizing these tasks decreases their performance when comparing to sequential native implementation. The reason is that there is an overhead to divide and manage the threads. On the server, these tasks combined are executed in almost 2 milliseconds, but the time required to transfer the data to the server and back to the device eliminated all this gain. It is important to mention that the latency of local networks

is usually around ten milliseconds, which has a significant impact when dealing with real-time systems.

As expected, the Full Native implementation is able to capture and render the frame using Android functions much faster than the other approaches, which use OpenCV functions to do the same task.

Regarding memory consumption, it is possible to see on Figure 3.14 that the Client/Server implementation is the most efficient. On the other hand, Multi-Thread Partial Native is the one that requires more RAM to execute. The reason is that the first one deals with less information because part of the steps is processed on the server. However, the second one manipulates all the data on the device and it is also necessary to create new structures to manage the different threads, which impacts on memory consumption.

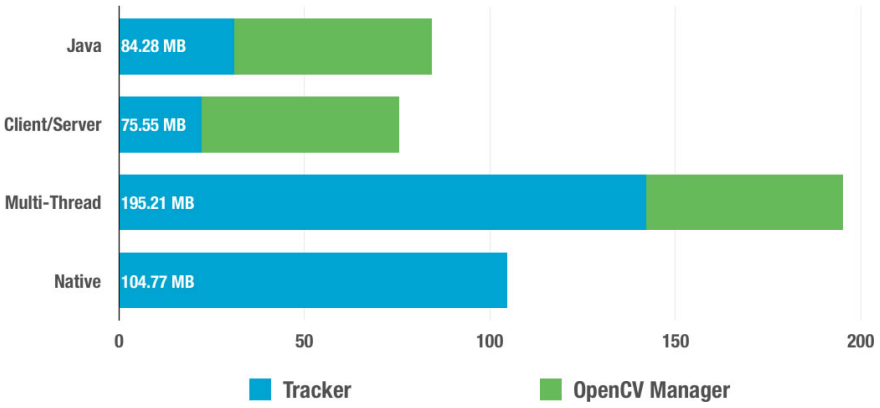


Figure 3.14: RAM memory consumption in MB during the execution of each architecture implementation.

The Full Native implementation also has massive memory consumption when compared with the Client/Server approach. It is due the fact that this version includes all the OpenCV dependencies because it is fully implemented using native code. On the other hand, the Client/Server version requires OpenCV Manager, which consumes twice more memory than the application itself.

As for the space required to store each application, both Client/Server and Multi-Thread Partial Native versions are much smaller than the Full Native implementation. One more time, the reason for that is that they do not embed the OpenCV dependencies because they access them from OpenCV Manager. Figure 3.15 shows that Full Native application is 7.55 times larger than Client/Server, but has less than half of its total size when OpenCV Manager is also taken into account.

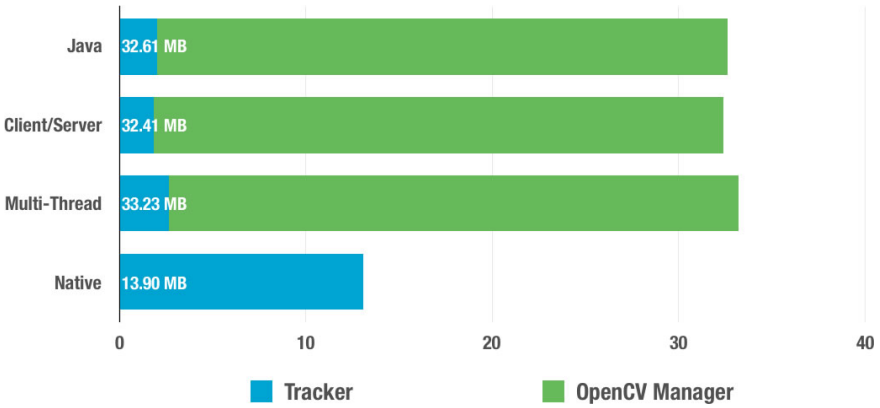


Figure 3.15: ROM memory in MB required to storage each architecture implementation.

3.2.3 Discussion

The tests showed that the selection of the best architecture depends on different factors. For instance, when using a mobile device that has a lot of memory and multiple cores, the best approach would be the Multi-Thread Partial Native. However, in case a good network infrastructure is available for a device with few resources, the Client/Server implementation would be a good choice.

It is safe to say that, on average, the Full Native architecture is the most efficient between the evaluated alternatives. It is approximately 23% slower than Multi-Thread Partial Native, but uses about half of the memory. On the other hand, requires around 39% more RAM than Client/Server running more than 45% faster. Other advantage is that it does not require the installation of any additional application. Additionally, depending on the algorithm that will be developed, it is possible to identify parts that could be paralleled to make the system faster.

Perhaps the ideal architecture would be the mixing of all of them. A Full Native approach that supports multi-thread and is able to transfer tasks to a remote server whenever this alternative is suitable. Therefore, the system would constantly and automatically evaluate the resources available on the device as well as the quality of the network to decide which one has the best trade-off for a certain task. There are works that use this principle to select the fastest tracking technique according to the available resources that will achieve the required level of accuracy (Wagner et al. 2009; Teixeira 2013). Nevertheless, no works were found that evaluate the resources of a mobile device.

3.3 Machine Learning Tracking on Mobile Devices

Because of the characteristics of machine learning approaches, which concentrate most of the computational effort in the offline training phase while the online tracker demands fewer resources, this type of technique can be very useful for tracking. For instance, it is possible to update the camera pose using a very small amount of information, but highly discriminative, which was inferred previously (Jurie et al. 2002).

One application in which machine learning is making an impact is to track a body or its parts, such as face and hand. Face tracking, in particular, has several purposes on mobile devices. For example, it can be used to identify the face of the owner of the phone using the frontal camera to unlock the device or not (Hadid et al. 2007). Another example is to provide a makeup tutorial using augmented reality (Almeida et al. 2015). In this sense, it is important to identify not only the position of the face on the screen but also the location of relevant points in the eyes, nose, mouth and chin, called landmarks.

This section describes the research process to develop a real-time face tracking technique for mobile devices that, given an input image, is capable of identifying face landmarks.

3.3.1 Research Methodology

There are several face tracking techniques. Among them is the Constrained Local Model (CLM) (Morency 2012) that performs a local search to find the position of the landmarks. Another one is the work described in (Ramanan 2012), which presents an idea similar to the CLM technique, but has better results. Although these techniques provide good precision, they are not optimized for mobile devices. Therefore, it was necessary to perform a literature review in order to determine if there was a more suitable approach.

The snowball sampling method was chosen (Given 2008), in which a good reference study is used as a seed to find other relevant works. In this sense, both the papers that are referred in the base study and the ones that cite them are gathered for further evaluation. For this study, the reference paper used was (Ramanan

2012).

The interest was in finding newer studies that improve the base technique. Therefore, the snowball sampling was applied in only one direction: collecting papers that cite the base approach. In April 2015, a total of 253 studies were selected. Because several papers were about either applications that used face detection algorithms or body tracking, only 22 works about full face tracking techniques remained. Next, those filtered papers were evaluated according to different metrics regarding aspects that would have influence on the performance and precision, as listed below:

- Performance aspects:
  - FPS;
  - RAM memory;
  - ROM memory;
  - Energy consumption.
- Precision aspects:
  - Datasets used;
  - Tracking precision;
  - Number of landmarks;
  - Faces in complex environments (In-the-wild).

Table 3.1 summarizes main aspects extracted from the most relevant papers evaluated. Finally, these features were analyzed and the Local Binary Features (LBF) method described in (Ren et al. 2014) was selected. It was the only study that presented an approach that also has a mobile device implementation with real-time results.

Table 3.1: Evaluation of main aspects of the most promising works. Cells with asterisk mean that there is not a clear value for the feature.

Study	FPS	Precision	Number of Landmarks	In-the-wild
Base Work (Ramanan 2012)	25 (desktop)	90%	68	Yes
LBF (Ren et al. 2014)	300 (mobile) 3000 (desktop)	*	29 - 164	Yes
Real-Time Face Detection in CUDA (Cheng et al. 2014)	45 (desktop)	*	68	Yes
One ms Face Alignment (Kazemi et al. 2014)	*	0.04 px	194	Yes
Face Estimation Under Occlusion (Burgos-Artizzu et al. 2013)	12 (desktop)	n/a	*	Yes

3.3.2 Local Binary Features Technique

The selected work presents a regression approach for face alignment that uses a locality principle to independently learn a set of highly discriminative local binary features for each facial landmark. The obtained local binary features are used to jointly learn a linear regression for the final output.

The selected approach predicts facial shape S in a cascaded manner. Beginning with an initial shape S_0, S is progressively refined by estimating a shape increment ΔS stage-by-stage:

$$S_t = S_{t\text{-}1} + \Delta S_t. \tag{3.1}$$

In a generic form, a shape increment ΔS_t at stage t is regressed as

$$\Delta S_t = W_t \phi_t(I, S_{t\text{-}1}) \tag{3.2}$$

where I is the input image, $S_{t\text{-}1}$ is the shape from the previous stage, ϕ_t is a feature mapping function, and W_t is a linear regression matrix.

3.3.2.1 Learning The Feature Mapping Function ϕ_t

The feature mapping function is composed of a set of local feature mapping functions that are learned individually and then combined to compose $\phi_t = [\phi_{t_1}, \phi_{t_2}, \ldots, \phi_{t_L}]$. A standard regression random forest is used to learn each local mapping function for each ϕ_{t_1}. The split nodes in the trees are trained using the pixel-difference feature and the one that gives rise to maximum variance reduction is selected, as shown in Figure 3.16.

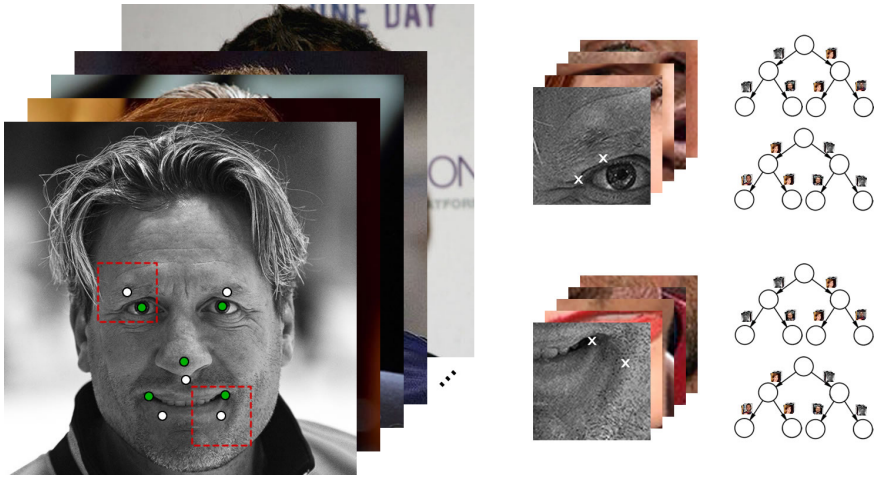


Figure 3.16: For every image on the training dataset, each local feature is learned individually from the landmarks (white circles). The intensity difference between two random pixels (white crosses) is used as decision function to split the training images. Each node has a distinct pair of features.

Only pixel features in a local region around the landmark are sampled. In the training phase, the optimal region size is estimated in each stage. The optimal radius region should depend on the distribution of the ground truth landmark of each training image around the respective landmark of the initial shape. Therefore, it decreases as the landmarks converge to the ground truth, as seen in Figure 3.17. As a consequence, it is necessary to extract fewer features on later stages than on earlier ones for maintaining the ideal distribution.

After creating the random forest, all training images traverse the trees until they reach one leaf node for each tree. The output of the random forest is the combination of the outputs stored in these leaf nodes. Supposing the total number of leaf nodes is D, the ϕ_t will be a DxL matrix in which a 1 value means the image reaches the corresponding leaf node and 0 otherwise, as illustrated in Figure 3.18. Therefore, ϕ_t is a highly sparse matrix called global feature mapping function and all ϕ_{t_L} are the local binary features.

3.3.2.2 Learning The Global Linear Regression Matrix W_t

After learning the global feature mapping function, the global linear projection W_t is learned by minimizing the distance of every landmark on the initial shape to the correspondent landmark on the ground

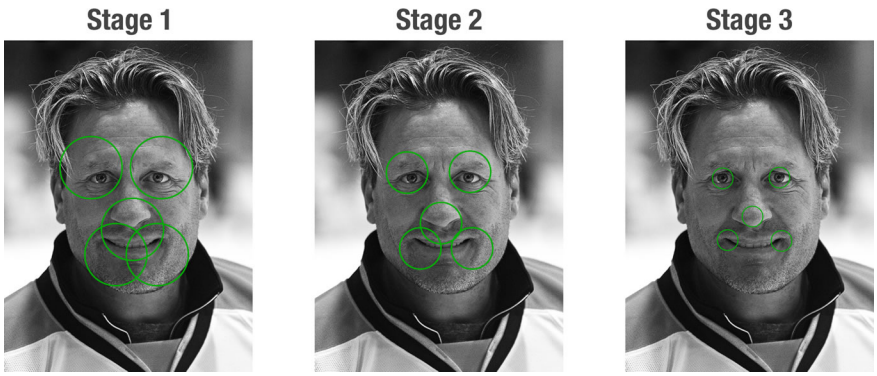


Figure 3.17: Local region around the landmark on every stage.

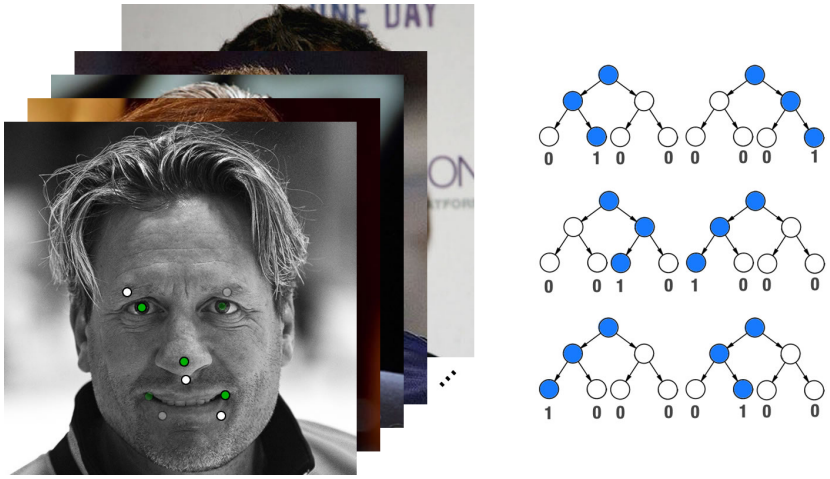


Figure 3.18: Traverse of three different landmarks of one of the training images on the generated forest. The sequence of zeros and ones of every landmark is the local binary feature.

truth.

This technique that uses a combination of local learners to discover global functions is called “transfer learning”, which the authors claim that significantly improves performance for two reasons: the locally learned output by random forest is noisy because the number of training samples in a leaf node may be insufficient and the global regression can effectively enforce a global shape constraint and reduce local errors caused by occlusion and ambiguous local appearance.

3.3.2.3 Tracking a New Face

The training phase provides the global feature mapping function and the global linear projection, which are loaded by the tracker. First, a regular face detector is used to determine the position of the face in the image. This position is used to choose a location to S_0 that is at least near the correct position of S. After that, the image traverses the trees until it reaches one leaf node for each tree, which will have 1 while all other leaf nodes will be 0. The global linear regressor is applied to the image’s local binary feature to determine the ΔS_0 increment and compute S_1, which is used as input to repeat the process until S is found. Figure 3.19 illustrates this process.

Figure 3.19: Cascade shape regressor, in which the landmark position is incremented every stage.

3.3.3 Implementation on Android

There is an open source C++ implementation of the technique available, which was not developed by the original paper's authors[2]. Figure 3.20 shows the results of this implementation.

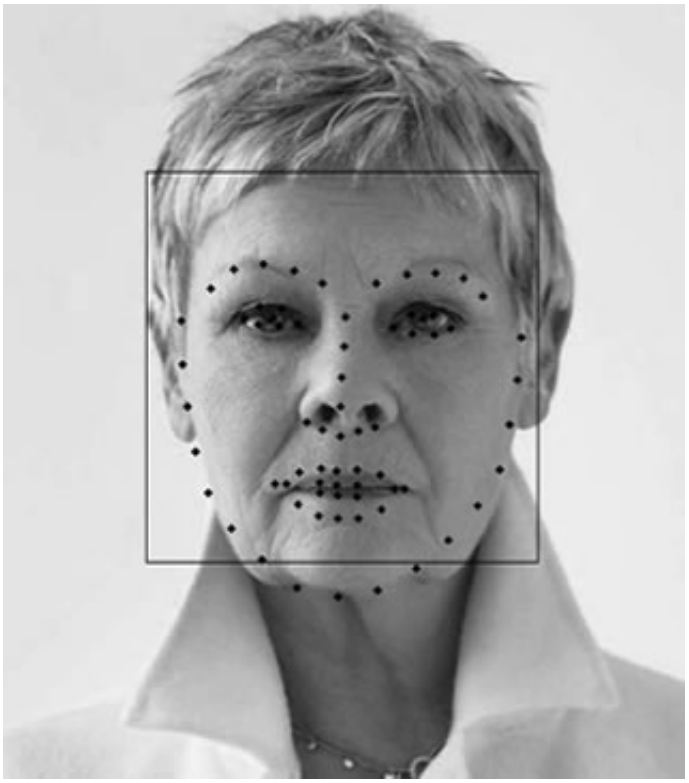


Figure 3.20: Result from C++ implementation of the LBF technique.

One advantage of having a C++ implementation is that it is easier to port to the efficient Android architecture described in the last section than other programming languages since it is not necessary to rewrite all the code. Additionally, OpenCV library is the only dependence of this implementation. Another benefit is the fact that it is possible to port only the tracking part of the system to Android and maintain the training phase in desktop without having incompatibilities regarding the training file.

This desktop platform implementation of LBF has some parts that do not compile or execute on the Android platform, such as the OpenCV functions for rendering images on a window. Because of that, it was created identifiers for both platforms, Windows and Android, in order to make the same code of the LBF system

[2] `https://github.com/yulequan/face-alignment-in-3000fps`

work properly on each of them. Hence, a Visual Studio project was created inside the JNI folder, in order to have an application for desktop environment that runs the same code of the Android environment. As a benefit, it is possible to execute, test and debug the C++ code using the Visual Studio IDE, once the Android developer environment does not provide tools for debugging the C++ code at this moment.

Additionally, it was necessary to compile some of the libraries used in the project itself. For example, in the case of the OpenCV library, only the modules utilized by the application were compiled, discarding unused modules, and this brought gains in the size of the system.

It was possible to test several parameters in desktop to determine which ones were the most suitable for Android. Four parameters were modified:

- Number of landmarks: L = 31 or L = 68;
- Number of trees per landmarks: $N \in \mathbb{Z} \mid 4 \leq N \leq 10$;
- Depth of each tree: $D \in \mathbb{Z} \mid 4 \leq D \leq 7$;
- Number of stages: $T \in \mathbb{Z} \mid 4 \leq T \leq 7$;
- Features per stage: $(F_1, F_T) \in \{(80, 200), (160, 400), (240, 600), (320, 800), (400, 1000)\}$.

The number of features on intermediate stages varies evenly from the minimal value to the maximum depending on the number of stages. Regarding the number of landmarks, 68 is the most common configuration used by most face tracking datasets. This is the only parameter that has influence in both training file size and execution time. Therefore, it was selected the smallest subset of landmarks that is necessary to identify the parts of a face. Figure 3.21 shows the standard landmark configuration with 68 points and the 31 landmarks selected. The number of stages, trees and their depth impact only the size of the training file, which grows as these parameters increase. Therefore, the code was modified to train and test the 1,120 combinations of these parameters automatically and evaluate them according to the average error of the estimated position of every landmark with respect to the ground truth, the size of the training file and execution time.

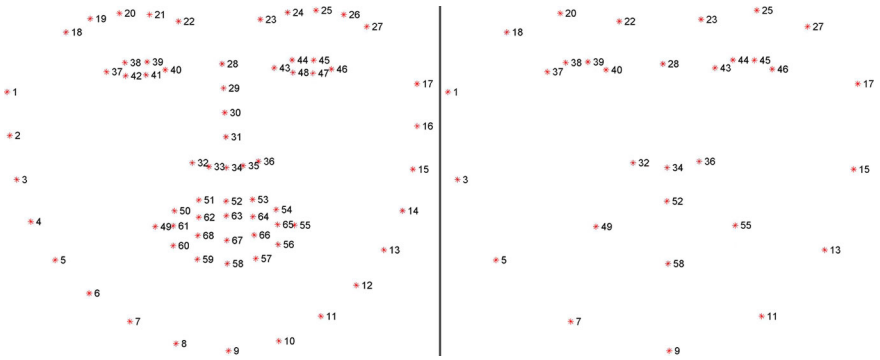


Figure 3.21: Standard landmark configuration with 68 points (left) and the 31 landmarks selected (right).

The set of parameters with the best trade-off between precision, execution time and training file size was P = {L = 31, N = 5, D = 4, D = 4 and F = (400, 1000)}. On desktop, this set of parameters presented a precision of 7.095 pixels, a training file of 1.60 MB and a mean execution time of 2.16 ms (462.96 FPS).

The training file generated with this set of parameters was used as input for the Android version of the LBF tracker. The evaluation was performed using an LG G3 device. The face was tracked using the frontal camera with 1920x1080 resolution. The binary application package file generated had 9.86 MB and used 45.4

MB of the RAM memory while executing. The execution time was 8.06 $\pm$ 3.57 ms (124.07 FPS). Regarding precision, since the Android version uses the same code as the desktop version, it is natural to say that they share the same error. Indeed, visually both implementations present similar results.

3.3.3.1 Improvements

The port of LBF to the Android platform brought some challenges regarding efficient implementation on mobile devices. One example is the data structure that should be used. The desktop implementation used double precision for floating point numbers. It has an impact on training file size. Changing these numbers from double to float decreased the size of the training file and did not have any implications in the tracking precision.

Another challenge is to know the resources provided by the Android platform and the libraries used. For instance, the first Android version was taking about 125 ms on average to track a single frame (8 FPS). The bottleneck was the OpenCV face detector, which takes more than 100 milliseconds to find the face position on each frame. The alternative was to use the Android native face detector, which provides the list of every face in a frame by the time it is available. This feature is accelerated in hardware and has no impact on the frame capture rate, which remains at 30 FPS with or without the face detector.

These are examples of situations and problems that can occur as the tracking system becomes more complex. The solution remains as lessons learned for future implementations.

Regarding improvements on the LBF itself, the original technique was designed for desktop and supports only images in which the face is upright. On the other hand, a mobile device can be held in any orientation while tracking the users' face. Thus, the first mobile version fails to track faces if the device is rotated. In order to solve this limitation, the first approach was to use the devices' sensor for obtaining its orientation. Then, the image is rotated to make the face upright. Finally, tracking is performed using the corrected image. However, the rotation of a full HD image is very slow. Rather than rotating the entire image, a more efficient approach is to rotate only the 31 landmarks of the initial guess and put them in the same orientation of the device, as shown in Figure 3.22.

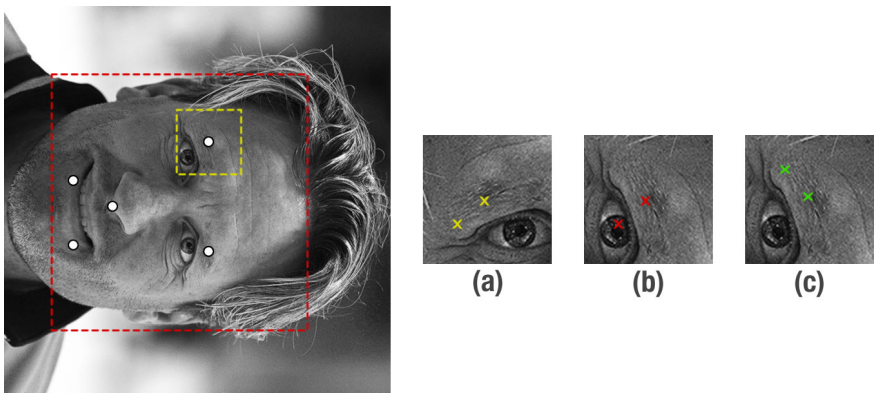


Figure 3.22: Initial guess chosen as if the image is in the upright position (a). If it is used on the rotated image, it will lead to an incorrect result (b). Therefore, if the initial guess is rotated using the device's orientation, it will be on the correct position (c).

However, the patch around each landmark that is used to transverse the tree is also rotated. Thus, the two pixels used in the decision node are also rotated in order to access the correct position in the image. This modification allows LBF to track in every orientation of the device and achieve the same result as in the upright position, as can be seen in Figure 3.23.

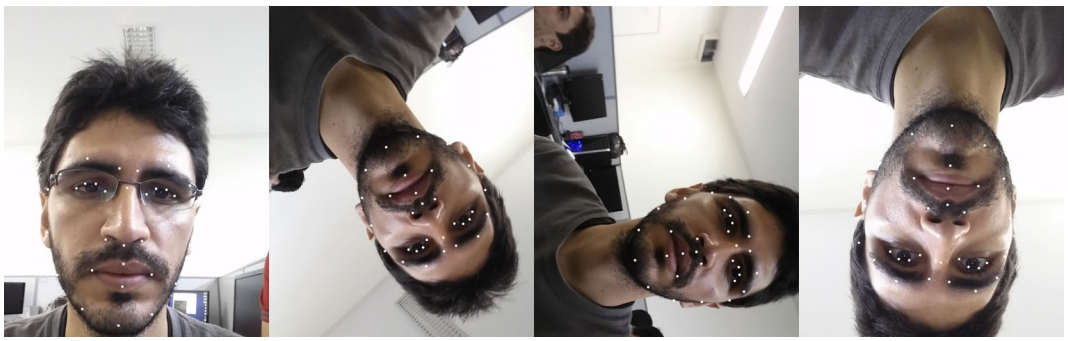


Figure 3.23: Tracking results with different device orientations[3].

3.3.4 Discussion

The use of machine learning to perform tracking seems to be very promising. The main concern is regarding the space necessary to store the training file because it can grow as the objects or scene to be tracked get more complex. For instance, in order to track only 31 landmarks in a face, LBF needs to load a 1.60 MB training file and to track a planar static model similar to the one used on the last experiment, the training file would have more than 25 MB (Simões 2016). This size will be even bigger when dealing with more complex environments, such as a small table with 3D objects and this is critical on mobile devices.

This fact is reflected on the systematic mapping presented in the previous chapter in which the machine learning approaches are used to track only parts of the body, such as face and hands. However, it is possible to make traditional methods more robust when combining them with learning techniques. One example is (Vineet et al. 2015), which uses a random forest classifier in order to evaluate the features extracted from an image to perform a semantic evaluation.

3.4 Simple SLAM System on Mobile Devices

Simultaneous Localization And Mapping, popular known as SLAM, is often related as an algorithm, but in fact it is the problem of building a map while localizing the device within that map at the same time (Leonard et al. 1991). It is often referred as a chicken and egg problem because a good map is required to localize the device while a precise location is needed to build the map. Moreover, these two problems cannot be solved independently of each other and consume a lot of computational resources.

However, SLAM algorithms have been recently deployed on mobile devices since they are continuously improving regarding processing power and memory. This fact makes them powerful enough to perform such complex tasks. This scenario favors the creation of numerous types of applications since these kind of devices create several opportunities that are only possible when the user can be mobile.

This section describes the initial stages regarding the port of an initial SLAM technique to a mobile device platform using Google's Project Tango tablet (Google Inc. 2014). The main motivation to use this device is to benefit from the technology embedded on it, especially the depth sensor. The results found on this implementation were also published in (Araujo et al. 2016).

3.4.1 SLAM Systems Works on Mobile Devices

SLAM systems are traditionally used for robot navigation (Sim et al. 2005). Recently, their use has become more and more popular in augmented reality applications (Klein et al. 2007). Since then, several

[3] A video with this result is also available at `https://goo.gl/kGRj4K`

works were developed using mobile devices so they can benefit from the mobility provided by such tools. One example is (Klein et al. 2009), which describes an initial prototype of a keyframe-based SLAM system running on an iPhone 3G. They have adapted PTAM (Klein et al. 2007) to generate small maps in order to mitigate the impact of the device's limitations regarding processing power, memory and storage capabilities.

Recent studies have proposed different types of solutions to overcome those limitations. For instance, (Pirchheim et al. 2011) uses the plane-induced homography between keyframes to implement an efficient mapping algorithm that decreases processing time. Li and Mourikis (Li et al. 2012) focus on visual-inertial odometry to create a real-time navigation system. Martin et al. (Martin et al. 2014) optimize SLAM on a mobile device by decoupling localization and mapping steps. Each one runs in a different thread and their results are combined at the end of the process.

On the other hand, mobile devices have some features that can help SLAM systems and are not available on a desktop platform, such as the device's sensors. Usually, they combine data measured by sensors with information extracted from the camera (Kao et al. 2013; Schöps et al. 2017). There are also works that use geolocation from the GPS combined with a remote server to reduce the amount of data processed on the device (Ventura et al. 2014). They create a 6DoF map locally on the mobile device. The global localization method, which runs on a server, processes the globally-registered map and returns a refined global registration correction to the mobile client.

Another way to efficiently run SLAM systems on mobile devices is using a panorama map (Pirchheim et al. 2013). They are registered on a 3D map in order to be able to maintain tracking during rotational camera motions, which is also a limitation for SLAM systems in general. Therefore, they can handle cameras with pure rotational motion while creating larger and denser maps. There are also works that integrate camera information with device's sensors to create the panorama map and continuously update it (Ventura et al. 2012). Because of that, the system can be used in large outdoor spaces.

3.4.2 Simple Tracking and Mapping

The proposed algorithm, called Simple Tracking and Mapping (STAM), is a monocular SLAM technique. In general, such kind of SLAM uses a standard camera that does not provide odometry, which gives the device position. Thus, it has to be performed visually. For STAM, visual odometry is done by tracking features using optical flow and feature descriptors. This information is used to triangulate new points and increment the map.

In the initialization phase, it is extracted the 2D coordinates of the corners of a chessboard pattern, and they are associated with their corresponding 3D points, which are known based on the size of each square on the pattern. Such keypoints are named *square features*. Additionally, it is extracted SURF (Bay et al. 2006) features from the entire initial frame, except the area inside the chessboard. Since this pattern is repetitive, this action minimizes the extraction of non-discriminative keypoints. These extracted features have no corresponding 3D points and are named *triangle features*. Then, it is computed the SURF descriptors for both square and triangle keypoints in the first frame. After that, the projection matrix of the first camera is calculated using only the square points since they are the most reliable points available. Finally, the first frame is stored as a keyframe and the square features utilized for the pose computation are kept in a track set, which consists of the collection of features in the map currently being used for tracking.

In the tracking phase, the points in the track set are tracked in the current frame using Kanade-Lucas optical flow (Lucas et al. 1981). Successfully tracked square features from the track set are then used to compute the projection matrix for the current frame based on the obtained 2D-3D correspondences. The track set is updated leaving in this set only the features successfully tracked so far. Then, it is performed a baseline check between the current frame and the last keyframe by verifying if the distance between camera optical

centers is higher than a threshold.

Whenever there is enough baseline, the mapping procedure is executed, in which SURF descriptors from keypoints extracted in the current frame are matched against the triangle features of the previous keyframe. The matching features are then triangulated, which means they now have a 3D point associated with them and are labeled as *square features*. They are also added to the map and the current track set. A feature matching between a square feature and a feature in the current frame is considered a re-detection. Square features that are re-detected are also added to the track set.

This process, illustrated in Figure 3.24, is repeated while there are new frames to be processed. A sparse bundle adjustment procedure that optimizes the camera trajectory is also performed after a pre-established number of new keyframes are added to the keyframe list.

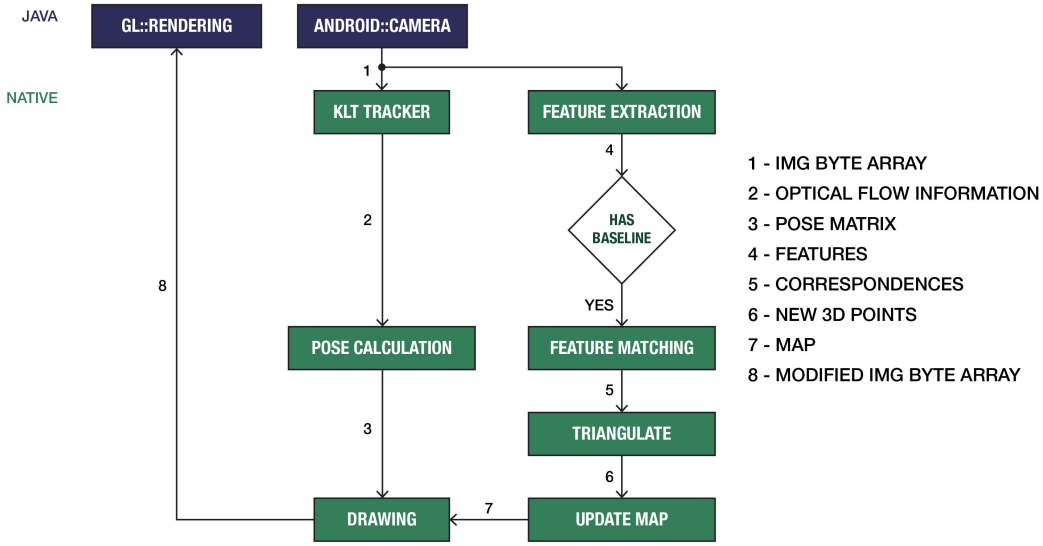


Figure 3.24: STAM flow diagram.

3.4.3 STAM Evaluation

As mentioned before, evaluating tracking techniques is a challenging task and many efforts have been made to provide metrics to analyze such systems. Since 2008, the International Symposium on Mixed and Augmented Reality (ISMAR) promotes the ISMAR Tracking Competition, a contest aiming to challenge state-of-the-art trackers through real-world problems. All the scenarios prepared for the competition try to replicate real problems for tracking systems, such as lighting conditions, task specificities, user constraints, levels of texture information available, objects relative size, camera resolution and others.

In 2015, ISMAR introduced a different competition style in which a SLAM system should be incrementally built in order to be able to compete on different levels. There were two categories: off-site and on-site competition (International Symposium on Mixed and Augmented Reality 2015). The former evaluates the system using images and participants would submit their results online. In the later, the system should be utilized in a real scenario that simulates a small office and participants should identify certain elements based on given 3D coordinates in an unknown area.

3.4.3.1 Off-Site Competition

This category was divided into three levels, which evaluated the processes of camera pose tracking step by step:

- **Level 1 - Point Matching:** finding the locations of known feature points in an image;
- **Level 2 - Point Tracking:** tracking known feature points in successive frames;
- **Level 3 - Mapping:** tracking and mapping new feature points in successive frames.

The first goal of *Level 1* was to find the 2D positions of the reference points, which were given by 2D image patches extracted from the same image it was used to find the 2D point, as seen in Figure 3.25. Every patch had a 3D point associated with it. Thus, the second objective was to calculate the projective transformation matrix from the correspondences between the 2D positions of the reference points in the image and their 3D coordinates.

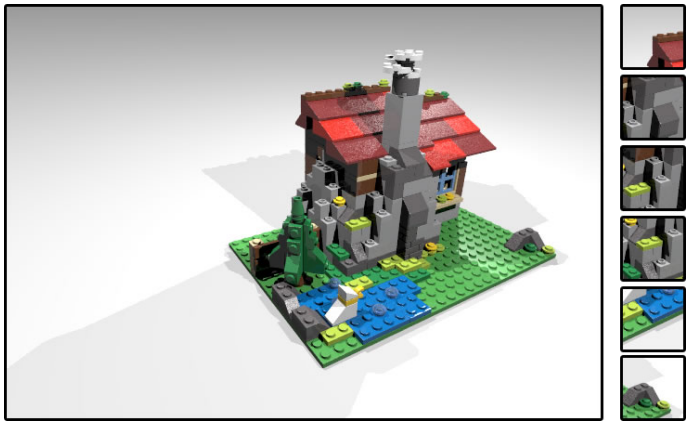


Figure 3.25: Frame from which the patches were extracted. Six samples of these patches are on the right. The 2D position of each patch should be found in the same image.

On *Level 2*, the goal was to find the 2D positions of the reference points that were also given by image patches in the initial image, then track them through the sequence of frames. Finally, calculate the projective transformation matrix from correspondences between the 2D positions of the reference points in the image sequence and their 3D coordinates. As seen in Figure 3.26, all the images of the sequence contain the same scene part that can be found in the initial frame.

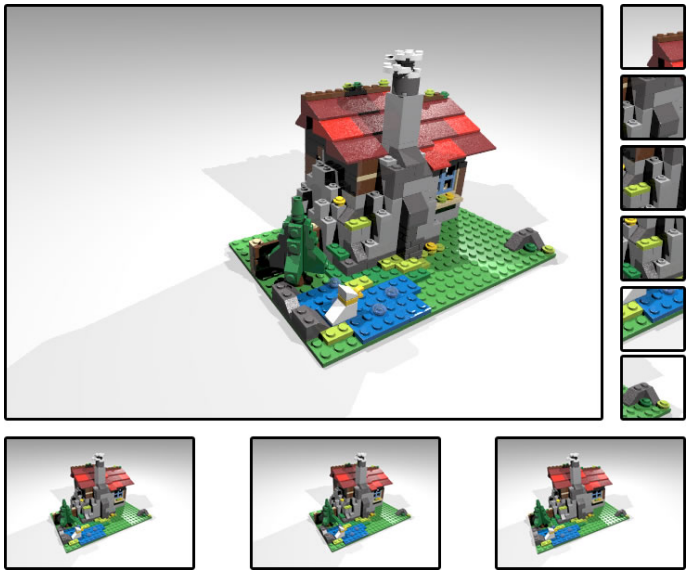


Figure 3.26: Six samples of patches (top right) were extracted from the first frame (top left). These patches should be tracked along the image sequence (bottom row).

The goal on *Level 3* was to find the projective transformation matrix from the correspondences between the 2D positions only in the last image of the sequence and their 3D coordinates. This task was similar to the previous level, except that none of the patches found in the first frame appear in the last image, as shown in Figure 3.27. Therefore, for this level, only tracking the given points was not enough and it was necessary to map new features.

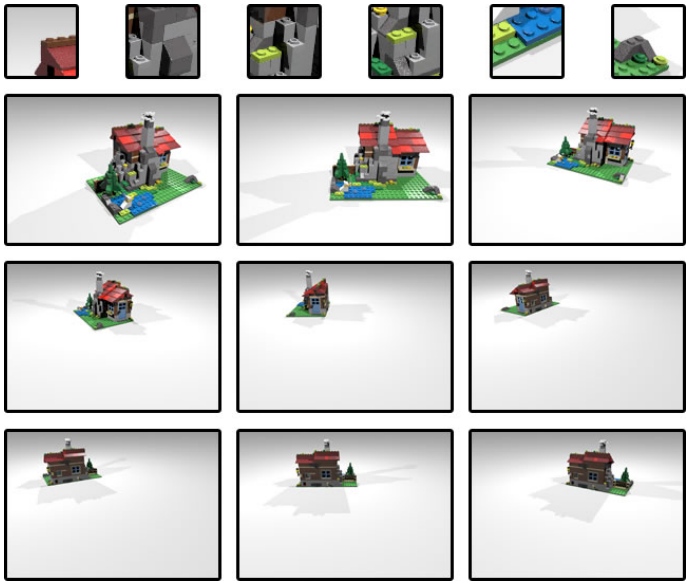


Figure 3.27: Six samples of patches (top row) were extracted from the first frame and nine sample images from the sequence to be tracked. Note that the patches are disappearing along the image sequence.

3.4.3.2 On-Site Competition

In this category, the contestant should use the system to guide him in an environment that simulates a regular office in order to mark the contest points in one of the posters on the walls. The task is similar to *Level 3* of off-site category, and it was performed in two sessions in which the contest points changed between them. The difference is that no initial points were given and the tracking area was much larger, an 8 by 8 meters room, as seen in Figure 3.28.

There was a chessboard marker printed on A0 paper on the starting area in which the size of each square was 10 cm, as shown in Figure 3.29. It was used to calibrate the coordinate system. Each rectangle on Figure 3.28 represents a table that had different types of objects that should be used by the tracking system. Each table had objects that presented different challenges for tracking systems, such as small and reflective objects or notebooks showing a video.

As noted in Figure 3.29, there were some posters with black and white squares on the wall where the contest points were located. Therefore, the system should display the virtual point correct position so the contestant could mark it using a pen. Additionally, the user should perform a specific path. From the start area, the contestant should enter the first hallway, assign the first and second points in session one and the first one on session two. Then, the contestant should enter the second hallway to mark the third or second point depending if it was the first or the second session. Later, the contestant should go back to the first hallway to collect the third point for session 2. Finally, the contestant should enter the ending area to assign the final points.

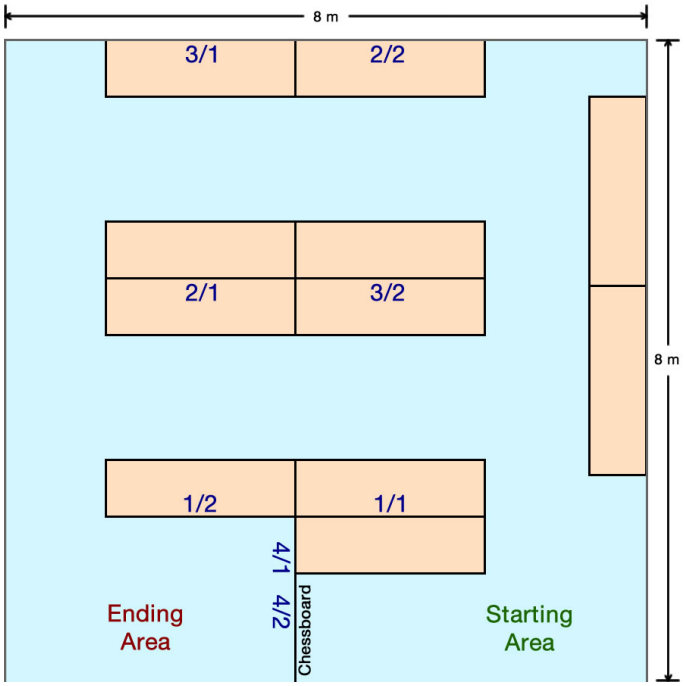


Figure 3.28: Schematic of the on-site tracking area. Each rectangle represents a table with different objects. The numbers X/Y indicate the poster used to assign the four competition points in which X represents the point order and Y the competition session.

3.4.3.3 Results

The first version of STAM was developed in C++ using data structures and basic functions from the OpenCV library. The competition rules stated that the tracker system must run using any device that only one person can carry and still be able to mark the points on the posters. Thus, the device used was a Microsoft Surface Pro 3 with an Intel Core i5 processor having 2 1.9GHz cores, 4GB of RAM memory and an Intel 4400 graphics card. The rear camera provided the video for tracking with 640x480 pixels resolution.

The STAM system was used to compete on Level 3 of the off-site category along with five other teams. The system was also evaluated in the on-site competition, in which there were two other teams. Regarding the off-site category, the organizers released the results anonymously and identified only the winner team. Since STAM was the winning system in that category, it is possible to attest its performance on Level 3.

Table 3.2 shows the result of Level 3, which is the average distance of all 3D points that were mapped on the last frame when projected using the matrix computed by the system to the projection of the same 3D points using the ground truth matrix. It is possible to see that only three teams were able to track the complete scene on every scenario. On average, STAM was more than four times more accurate than the second place, which indicates that the algorithm used to optimize camera trajectory was able to minimize the reprojection error during the scenes.

The results for the on-site category are shown in Table 3.3. Note in Figure 3.29 that there were several repeated objects on the table. Some of them were small, others were reflective and a few would change during the time. Additionally, the spaces between tables were almost textureless. All these facts make very hard for a system to map correctly such environment. Therefore, it is possible to see that only one team and in one session was able to map and track the whole area. As for STAM, it was able to identify only the closest point from the starting place. That is due to the fact that the system accumulates too much error during tracking in such complex environment.

Figure 3.29: Images from the tracking area. The chessboard used to calibrate the tracking system is seen on the top left image. The other images show the tables with the trackable objects and the posters to mark the 3D points.

Table 3.2: Mean reprojection error in millimeter of all points on the last frame of off-site category Level 3[4].

Team ID	Scenario 1	Scenario 2	Scenario 3	Average
301	552.833	2245.485	-	-
303	253.760	1250.542	402.034	635.445
304	217.836	-	-	-
305	1673.195	-	-	-
STAM	104.209	44.990	49.320	66.173
309	195.672	448.788	226.012	290.157

3.4.4 Implementation on Tango Device

As mentioned in Section 3.1, Tango is a platform that combines computer vision techniques with state-of-the-art sensors, allowing to perform tracking on mobile devices. For this implementation, it was also selected the Yellowstone tablet.

3.4.4.1 Android Programming

Since STAM was developed in C++, it can be ported to the efficient Android architecture in a process similar to the one used to port LBF to mobile devices. That includes the identifiers for both platforms, Windows and Android, so the same code of the STAM system works properly on each of them.

Additionally, it was necessary to compile some of the libraries used in the project itself. For example, the OpenCV library was recompiled to use only the modules necessary to the application and also to add extra modules required by the STAM system that are not available on the standard OpenCV version, such as xfeatures2d. Another example was the library for performing bundle adjustment. In desktop, it was cvsba (Lourakis et al. 2009), which does not have an Android version. Thus, it was replaced by the Ceres Solver library (*Ceres Solver*), which is an open source C++ library that supports Android and was also recompiled for the mobile platform.

[4] A video with this result is also available at `https://goo.gl/3mdkcL`

Table 3.3: Mean reprojection error in millimeters of on-site category.

	Session 1				Session 2			
Team ID	**1/1**	**2/1**	**3/1**	**4/1**	**1/2**	**2/2**	**3/2**	**4/2**
A	55	328	239	-	197	291	-	-
STAM	42	-	-	-	231	-	-	-
C	54	291	53	-	227	271	245	262

3.4.4.2 Results

It was performed a preliminary evaluation of the Tango version of STAM using the scenario shown in Figure 3.30. It aims to compare performance and precision between the desktop and mobile implementations. Although the Android version shares the same code of the desktop development, the optimization with bundle adjustment uses a different library. Therefore, it is necessary to evaluate if this modification has any impact on the precision.

Figure 3.30: Evaluation environment with the calibration chessboard in the middle and objects with different types of texture around it.

As for execution time, it was measured the time to perform the three main steps of the STAM technique, which are calibration, frame by frame tracking using optical flow and optimization with bundle adjustment. Figure 3.31 compares the results between the desktop[5] and the Tango implementation.

The performance difference between platforms on the calibration step is the smallest one. It is because this procedure only executes OpenCV functions and stores only few data. The performance difference increases for the tracking step due to two main reasons. First, as the number of points on the map increases, the Android version felt more the lack of processing and memory resources than the desktop version. Then, as tracking processing occurs, the Tango device starts to heat up fast, making the execution even slower.

The biggest difference between platforms is on the optimization process. It is because STAM uses different bundle adjustment libraries on both environments. Moreover, the cvsba library only optimizes the camera trajectory while the Ceres Solver library optimizes both camera path and point cloud. Therefore, the optimization step on Android deals with much more variables than the desktop implementation.

Regarding tracking precision of STAM system on the mobile device, the reprojection error on both implementations was compared. It was used a chessboard printed on a graph paper to calibrate the coordinate system. Then, the system starts to track a 3D point located on the outer corner of one of the squares, shown in

[5]Intel Core i7 with 3.60GHz and 8GB of RAM

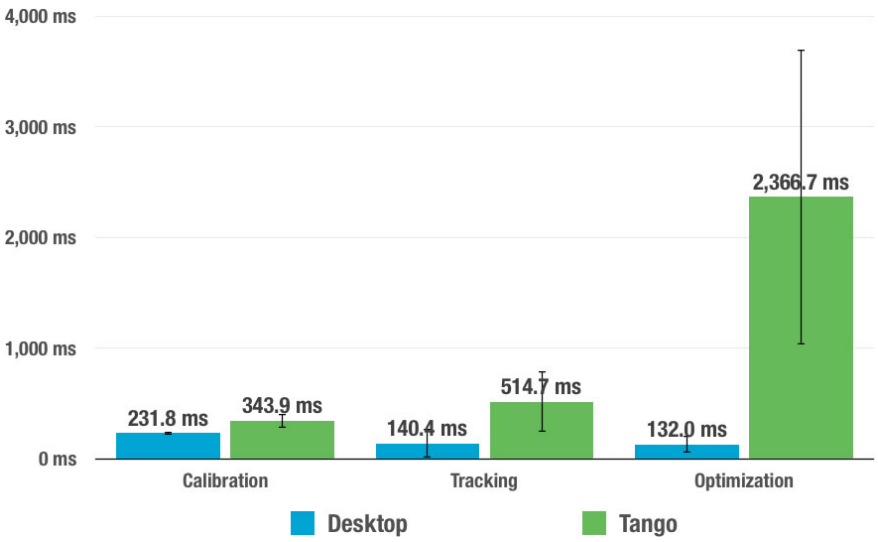


Figure 3.31: Execution time in milliseconds to run the main steps of STAM.

Figure 3.32. After one minute of tracking in which the device was moving freely, the user marked the position of the point at the paper. The same procedure was performed with the desktop version. The average distance of every assigned position to its correct location is the tracking precision. The error of STAM running on Tango was 10.50 millimeters ($\pm$ 3.91) while the average error on desktop was 19.75 millimeters ($\pm$ 6.98). This was expected because the Android version has a more robust optimization.

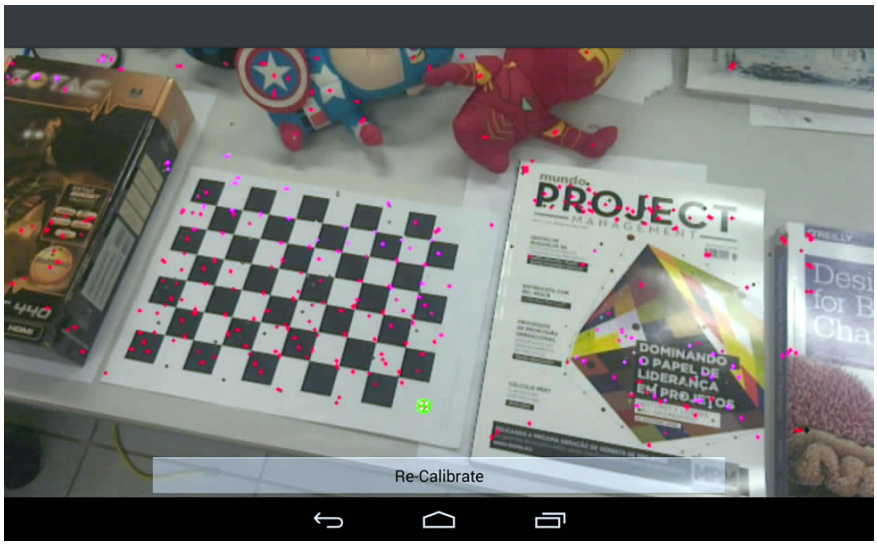


Figure 3.32: Tracking procedure running on Tango device. Small points are the keypoints extracted from the mapping while the large green dot shows the tracking point, which is a known point in the real world based on the chessboard template[6].

[6]A video with this result is also available at `https://goo.gl/ME9JHJ`

4
Incremental Semantic Tracking

From the preliminary experiments, it was acknowledged the importance of finding high-level semantic information from a point cloud. This is a challenging task and it can be used in various applications. Also known as called semantic modeling, it is useful to compactly represent the scene structure and efficiently understand the scene context. And such knowledge results in several improvements when applied to tracking techniques. It was mentioned in this thesis that two of the most used augmented reality libraries incorporate simple semantics into their tracking techniques. Google's ARCore and Apple's ARKit estimate planes to place the augmented content, which can be the floor, a table or vertical planes such as walls. With the detection of planes, these trackers allow a more realistic rendering and stable positioning of the virtual objects.

The benefits of detecting other kinds of shapes are even higher. For instance, it can be used to provide haptic feedback on augmented reality applications (Hettiarachchi et al. 2016). Besides that, objects are usually overrepresented when defined using a point cloud because it is not necessary to have so many points to describe it. Therefore, an implicit representation can replace redundant points, which is particularly helpful when targeting devices with memory restrictions, such as robots or UAVs. Furthermore, a tracking system can use these primitives to denoise the reconstructed map or constrain its optimization (Ramadasan et al. 2015), which can reduce tracking errors. For instance, these semantic constraints can make a difference in the optimization phase of the experiment presented in Subsection 3.4.4, in which the execution time increases as the number of points grows.

This chapter explains a method for incrementally modeling and tracking primitives on a sparse point cloud. It uses the generating process of point clouds on SLAM effectively and relies on geometric and statistical analyses to filter unreliable shapes. This approach was presented in (Roberto et al. 2018) and is the extension of the technique published in (Roberto et al. 2017).

4.1 Semantic Modeling

Automatic reconstruction of 3D object shapes is useful for several applications, such as blueprint generation for architecture. Since it is a difficult task to accomplish, it has been a relevant research topic for years. Several methods have been proposed to achieve it by using laser scanners (Zhu et al. 2011) and cameras (Ma et al. 2005). Due to low-cost RGB-D sensors, namely Microsoft Kinect and mobile devices with Google Tango, 3D data acquisition became more common and runs in real time even on such devices. These sensors acquire the depth of an object on a pixel-by-pixel basis and, then, describe the obtained shapes as a point cloud. Although they are very useful for 3D measurement and visualization, there are some aspects to be improved. For instance, the scene is usually represented using a point cloud or a mesh computed from it. The latter simply consists of connected points with little information about the semantic structure.

Several methods have been proposed in the literature to determine semantic in point clouds, most of them dense. One conventional approach is to use reverse engineering techniques to estimate geometric primitives, such as region growing (Holz et al. 2012). It can efficiently deal with large amounts of data because

it makes simple comparisons using the normals to determine if a set of points belongs to the same group. However, this approach is not robust to noisy point clouds because it can lead to a wrong classification. The robustness has been improved using both Hough Transform (Drost et al. 2015) and RANSAC (Lopez-Escogido et al. 2014). Another approach is based on machine learning techniques, which combine local features and AdaBoost to detect complex objects (Pang et al. 2013). Support Vector Machine (SVM), Fast Point Feature Histogram (FPFH) descriptors and RANSAC are also used to extract semantics from a point cloud (Huang et al. 2013a). Recent works used Convolutional Neural Network (CNN) to perform a semantic classification using the keyframes of a dense monocular SLAM and them apply this result to the point cloud (Tateno et al. 2017).

Some methods are able to detect only specific primitives, such as planes (Nguyen et al. 2015; Oehler et al. 2011) or cylinders (Liu et al. 2013; Qiu et al. 2014). Although limited, detecting only one class of shape still has several applications. For example, (Kim et al. 2012) estimates the floor plan of houses from the planes detected in a dense 3D point cloud generated using the Kinect sensor. On the other hand, other methods deal with different classes of shapes at once. For instance, GlobFit (Li et al. 2011) can estimate planes, cylinders, cones and spheres. Some even detect pre-modeled complex shapes along with planes and cylinders in industrial scenarios (Pang et al. 2015).

One advantage of having dense data extracted from laser or infrared sensors is that the acquired point cloud contains several information that the algorithm can use for the detection. Moreover, the data is relatively noiseless, which means that a large number of points can stably fit primitive shapes. Therefore, it is more challenging to extract primitive shapes in a noisy and sparse point cloud of a partially-observed object computed from image-based approaches with mobile devices. Some studies have tackled this issue by limiting specific situations, such as detecting only one shape class, such as ARKit and ARCore. Another example is (Sinha et al. 2008), which estimated planes based on their reconstruction to create textured models. In this context, RANSAC based methods are very promising to work with sparse point clouds. It is because they estimate primitives by initially picking a minimal group of points for each shape and detecting the one that approximates the maximum number of points (Schnabel et al. 2007). Besides, they can also work with data containing a large number of outliers (Roth et al. 1993). However, the performance of such approach for sparse point clouds was never investigated.

Another aspect of existing semantic methods is that they usually work in batch, which means that an input data is analyzed all together only once. It is consistent with the generation method. Usually, the dense sensors generate the entire point cloud at once. However, the performance of visual SLAM system regarding both the accuracy of the reconstruction and the computational cost for real-time applications improved drastically in recent years (Mur-Artal et al. 2015; Uchiyama et al. 2015). One characteristic of several of these SLAM methods is that the 3D map is generated incrementally. This generating process can provide valuable information for a semantic approach in addition to the point cloud itself.

4.2 RANSAC-Based Method on Sparse Point Cloud

The first step in this research to perform semantic modeling and tracking on sparse point cloud was to evaluate how dense methods perform using sparse maps. This could provide valuable information to develop a new approach specific for sparse point clouds or adapt existing ones. As mentioned in the previous section, RANSAC methods seem to be promising for that task. From all approaches found in the literature, Efficient RANSAC (Schnabel et al. 2007) presents the best results. Both regarding execution time and the precision of the primitive estimated. Moreover, it detects multiple classes of shape, which are planes, cylinders, spheres, cones and torus.

4.2.1 Method Overview

Efficient RANSAC requires a point-cloud $\mathscr{P}$ with normals for each point as input. The output is a set of primitive shapes Ψ in which a point $p_N \in \mathscr{P}$ will be assigned to only one shape $\psi_n \in \Psi$ or it will remain unassigned.

In summary, (Schnabel et al. 2007) searches the primitive with maximal score m for each iteration of the technique. The score is a function based on the number of points that can be part of a shape candidate, which consider the distance of a compatible point as well as the deviation of its normal from the one of the primitive. Therefore, for each iteration, the algorithm randomly selects one point of $\mathscr{P}$ and then collects a minimal subset of points that are closer to the first one. It is because (Schnabel et al. 2007) explores the fact that shapes are local phenomena, which means that the probability that two points belong to the same primitive is higher the smaller the distance between the points.

Candidates of all considered shape types are generated for every minimal set and all candidates are collected in the set $\mathscr{C}$. All of the candidates need only three points to describe each type of shape. The score m for each candidate is computed using a statistical approximation that increases the algorithm performance. The candidate is only selected if the probability $P(|m|, |\mathscr{C}|)$ that there is no better candidate is high. In this case, $|m|$ and $|\mathscr{C}|$ are the number of points in the shape and the number of candidates, respectively. When a candidate is accepted, the corresponding set of points is removed from $\mathscr{P}$. The algorithm repeats until the probability $P(\tau, |\mathscr{C}|)$ of not finding new shapes is high given a τ value that is defined by the user.

4.2.2 Evaluation

There is an implementation of Efficient RANSAC available[1]. This version was used to test how it deals with noisy and sparse point clouds. It was created a test scene in which two consecutives keyframes can be seen in Figure 4.1 (a) and (d). The point cloud, shown in Figure 4.1 (b) and (e), were generated using a SLAM system (Uchiyama et al. 2015) that was modified to increase the number of generated 3D points. These clouds have 1,427 and 2,180 points respectively. The reconstruction of the clouds was calibrated using a chessboard pattern so that the clouds were represented in metric scale.

Efficient RANSAC has five heuristic parameters. To deal with a sparse point cloud, more than 180 thousand different combinations of these parameters were automatically tested. The goal was to find the best set that maximized the number of assigned points, which were the ones from the input point cloud that were modified to be fitted to a shape. These parameters should also minimize the distance between these points and the ones on the input point cloud. Figure 4.1 (c) shows the result for one of the best set of parameters, which assigned 95.937% of the points with an average distance of 1.949 millimeters for each assigned point. Efficient RANSAC was able to detect most of the shapes correctly. However, one of the detected primitives was different from the expected result since a plane was identified as a cylinder.

Another issue noted was the inconsistency between keyframes. Figure 4.1 (f) shows the shapes detected using the data from the subsequent keyframe. In the result, 93.896% of the points were assigned with an average distance of 2.098 millimeters. Four detected primitives were wrong, including two planes identified as cylinders. Compared to the previous keyframe, six shapes were different. This unstable result occurred often due to the noisy data combined with the small number of points.

[1] Available at `http://cg.cs.uni-bonn.de/en/publications/paper-details/schnabel-2007-efficient/`

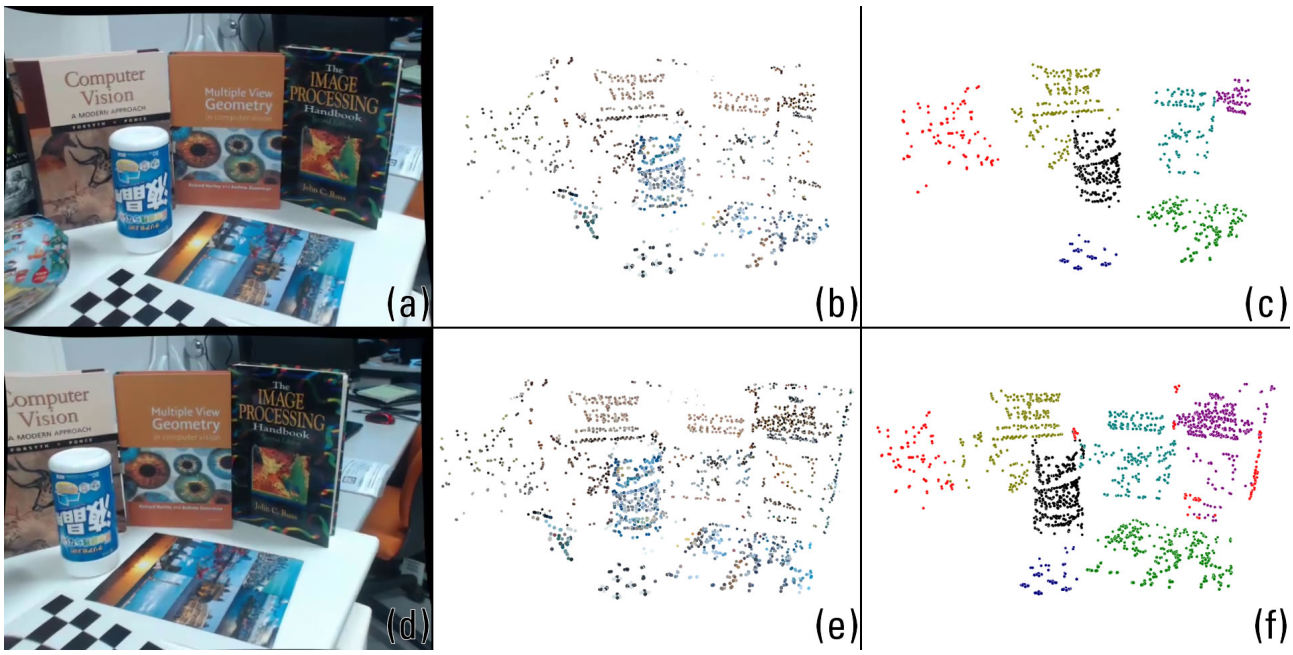


Figure 4.1: Test scene (a) and (d) and their reconstructed point cloud (b) and (e). Eight shapes were detected on the first keyframe (c) and twelve primitives were found on the second one (f). Even with the point cloud being very similar, the red points represent the six primitives that were differently detected between keyframes.

4.3 Geometric and Statistical Incremental Semantic Tracking

From the evaluation, it was clear that Efficient RANSAC achieves good results when detecting primitives in a sparse point cloud. However, it still requires improvements of consistency and precision for various applications. Therefore, it is possible to use the primitive estimation from Efficient RANSAC and the generation process of point cloud from visual SLAM systems to perform an incremental semantic modeling. This approach can improve both the precision and stability of the primitive detection. Moreover, it also allows the tracking of these primitives through the scene.

In summary, this method runs Efficient RANSAC using the sparse point clouds that are incrementally generated from a visual SLAM system. Then, it uses the history information of the estimated primitives and their parameters to match the shapes over time. Also, it estimates the reliability of the detected primitive using the geometry of the shape. When this estimation is not reliable enough, it performs a statistical evaluation using the detection history to eliminate random detected shapes. Figure 4.2 illustrates the flow of the method, which is detailed in the following subsections.

4.3.1 Efficient RANSAC

When a visual SLAM system reaches a keyframe, it updates the map adding new points to it. Then, the Geometric and Statistical Incremental Semantic Tracking method, or simply GS-IST, runs Efficient RANSAC to detect the shapes for every new map. However, it was necessary to make some modifications in both the code and the configuration in the available code of Efficient RANSAC. The configuration changes aim to reduce the number of types of primitives detected. Instead of five, three classes of shapes are tracked: planes, sphere and cylinder. They were selected because they can be used to model most of the objects. For instance, when modeling industrial scenarios, almost 80% of the scene can be represented only by planes and cylinders (Huang et al. 2013a).

There were two code modifications. The first one intends to replace the random selection of the initial

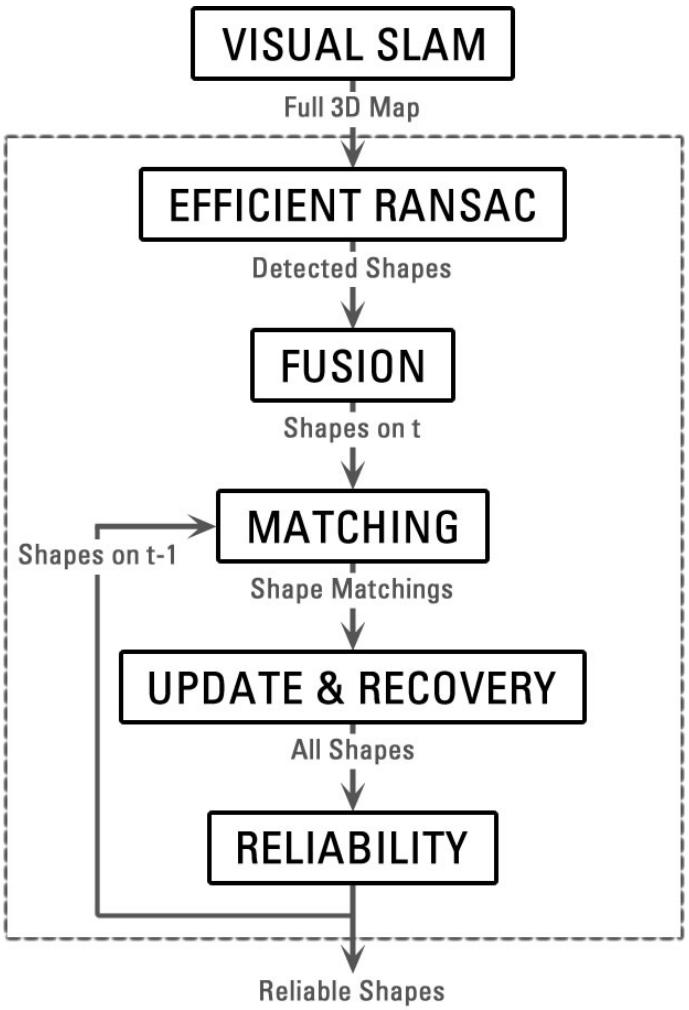


Figure 4.2: Flow of the Geometric and Statistical Incremental Semantic Tracking (GS-IST) approach.

seed to use the same pseudo-random point for every execution, which helps in the evaluation process. The other change was necessary to save the value of intermediary computation in the Efficient RANSAC algorithm that will be used later by GS-IST. This data is the average Euclidean distance between the points in the input point cloud and their respective projection on the estimated shape. Since Efficient RANSAC computes this projection as part of its algorithm, returning this information would be more effective than recalculating it later. Originally, this library returns the list of primitives with their respective parameters and all input points used to estimate them projected on the shapes. After the modification, it also provides for each primitive the sum of the distances between all input points and their projections. This way, the necessary information will be available for a quick access when needed.

All the primitives are passed to the subsequent modules, which will identify and eliminate the unreliable shapes and track the remaining ones.

4.3.2 Shape Fusion

In keypoint-based visual SLAM systems, most of the characteristics are clustered at highly textured areas. For example, in Figure 4.1 (a), two different patterns over the table were observed and formed two distinct clusters of points as illustrated in Figure 4.1 (b). In this case, Efficient RANSAC detected them as two separate planes even though they belong to the same plane on the table.

In order to improve the results from Efficient RANSAC, different shapes that belong to the same

primitive are first fused. It not only provides a more representative primitive but also increases the overall information regarding the shape. Since a primitive with a small number of points can be unreliable, it is better to discard it. However, if more than one shape with similar parameters is detected, even if they are small, it is safe to assume that they correspond to the same element in the scene. The reason is that it is unlikely that two primitives are wrongly detected with the same parameters. Therefore, the fusion of primitives based on parameters helps in the history evaluation. This process uses the similarity of the shape parameters to decide whether to fuse two shapes or not:

- **Plane:** the planes are parallel given an angle threshold α_t and the distance between them is smaller than the distance threshold d_t;
- **Sphere:** the distance between their centers is lower than d_t and the difference between their radii is less than the radius threshold r_t;
- **Cylinder:** the angle between the axis direction of both cylinder is smaller than α_t, the distance between them is less than d_t and the difference between their radii is smaller than r_t.

While d_t and r_t are controlled for each case such that they are 2% of the largest size of the point cloud bounding box, α_t is always set to 5°.

Additionally, it is considered the proximity between the primitives to restrict or widen the similarity thresholds. The principle is that two distant shapes have a smaller possibility to be the same than closer ones. Experimentally, the thresholds are widened by 25% when evaluating the fusion of primitives that have an intersection. Otherwise, it is restricted by 25%. This means that distant shapes have to be more alike to be merged. On the other hand, closer primitives are more likely to be the same and the threshold can be less restricted.

4.3.2.1 Parameter Computation

Similar shapes according to the aforementioned criteria are fused. GS-IST sets the parameters of the resulting fused shape as a weighted average between the parameters of both primitives. The weight is based on the geometric analysis of the points in the detected shape. The main idea is to use the average Euclidean distance of each input point that was used to estimate the primitive to the resulting shape. Even for noisy data, this distance will be smaller on correct estimations than on wrong ones. For instance, considering a globe being tracked that was correctly modeled as a sphere or incorrectly detected as a plane. The average distance of the 3D points in the globe to their projection in the sphere will be smaller when compared to the distance of the same input points to their projection in the wrong plane. Figure 4.3 illustrates this idea.

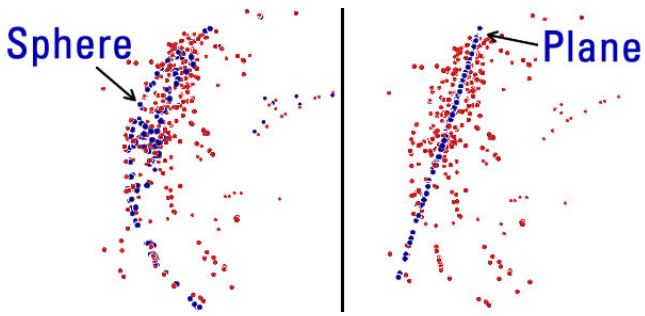


Figure 4.3: Difference between the input points to their correspondent projected points on a shape estimated correctly (left) and incorrectly (right).

Thus, the parameters P_f of the fused shape will be:

$$P_f = \sum_{i=1}^{n} w_i P_i, \tag{4.1}$$

where n is the number of similar shapes to be fused and P_i are their parameters. The weight w_i is:

$$w_i = \frac{np}{\sum_{j=1}^{np} d_j}, \tag{4.2}$$

where d_j is the Euclidean distance between the input points to its projection in the estimated shape and np is the number of points in the estimated primitive. The weights are normalized and $\sum_{i=1}^{n} w_i = 1$.

Using Equation 4.1, every fused shape will influence the resulting primitive. However, it will be closer to the one with the smaller error. This average can be applied to every parameter except the plane and the cylinder position. For the plane position, it is only valid if they are all projections on the other planes. Thus, the position of one point is projected in all others and then the weighted average is computed, as illustrated in Figure 4.4. This will also work in case of parallel planes. As for the cylinder position, this parameter will be the axis intersection. In case of concurrent or parallel axes, the cylinders are fused in pairs if there is more than one. First, it is selected the points on each axis that is closer to the other and the resulting position will be their weight average.

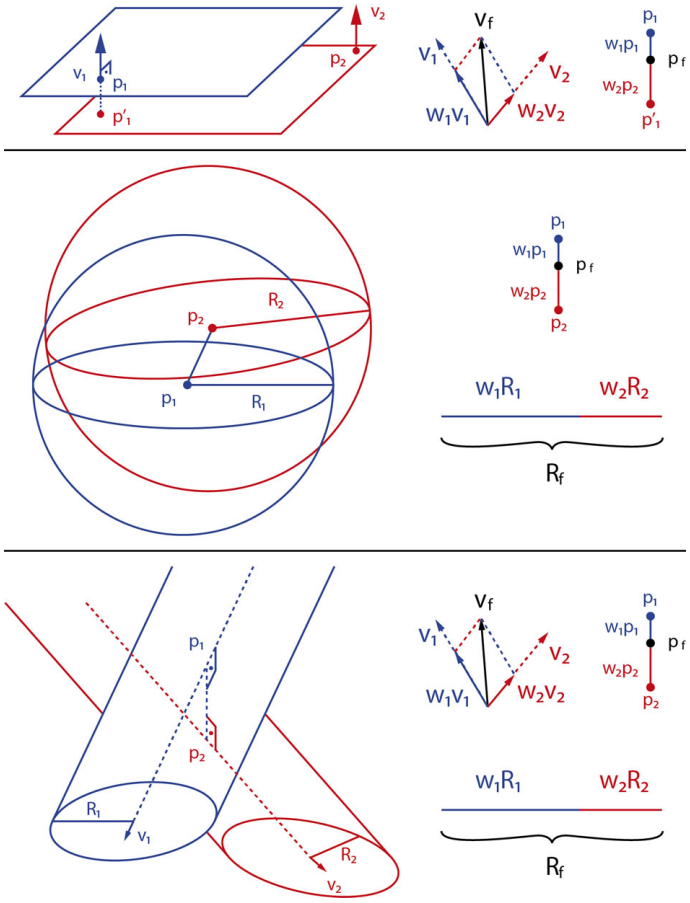


Figure 4.4: Fusion of parameters for different classes of shapes.

4.3.2.2 Inclusion Criteria

In case a shape is not fused with any other, GS-IST performs an initial reliability evaluation to decide whether to keep this detected primitive or not. It is used four geometric characteristics of the primitive to make

this assessment, as described below:

- **Number of Points:** shapes with more points are usually more reliable because the estimation is based on a large amount of data. The number of points in good primitives is *larger than 2%* of the whole point cloud;
- **Dispersion:** it measures how spread are the points in the primitive. Since the keypoints are clustered around highly textured areas, small regions tend to concentrate most of the shape points. The dispersion of reliable shapes is *smaller than 20%* of this value for the whole point cloud;
- **Distance:** it is the same Euclidean distance mentioned previously. Reliable shapes have an average distance *smaller than 5%* of the largest size of the entire point cloud bounding box;
- **Radius:** the sphere and cylinder radius can also provide a hint regarding the shape's reliability. A noisy plane can be estimated as one of these two primitives with a considerable radius. Therefore, good spheres and cylinders have radius *smaller than* the largest size of the entire point cloud bounding box. It should be noted that this criterion is not applied to planes.

The system only keeps shapes that pass in all of these criteria. These values were determined experimentally and they are based on the dimension of the input point cloud because it puts all thresholds in proportional to the scene scale.

4.3.3 Shape Matching

To match the primitives between consecutive keyframes, GS-IST uses the intersection of the 3D bounding box and the distance between the center of mass. Due to several factors, such as inconsistency or the shape volume, a primitive on a given keyframe can match with several others on the previous one, including shapes of a different type. Therefore, it is necessary to detect the shape on the previous keyframe that is the most likely to be the correspondent on the current one. It is computed a s score for each primitive that a given shape on current detection intersects on the previous estimation. This score is proportional to the intersection volume and inversely proportional to the distance between the center of mass:

$$s = \frac{\sum_i^{ns} |\psi_i| p_i}{\sum_i^{ns} |\psi_i| d_i} \frac{|\psi_s|}{\sum_i^{ns} |\psi_i|}, \tag{4.3}$$

where ns are the indexes of the shapes with intersection on previous keyframes, $|\psi_i|$ is the number of points of that shape, p_i and d_i are the intersection ratio and distance between centers of mass, respectively, and $|\psi_s|$ is the number of points in the primitive on the current keyframe. It is selected the one with the maximum score as correspondence.

4.3.4 Shape Update and Recovery

The shape detected on current frame inherits the history data of the one it matched on previous detections. These data contain the primitive class that was detected on every keyframe, as well as the average distance to the original point cloud at that detection. With that information, GS-IST can verify if the current estimation is following the historical data.

The system checks the class that this primitive was detected as over time. If the current shape has the same type of the one that appears in more than half of them, including the current detection, its parameter is updated. This new parameter will be the weighted average of each detection over time. With this update, every

previous detection will influence the final shape, which results in primitives that are more stable through time rather than using just the parameters from the last one.

The process to update the new parameter P_u is similar to the one explained in Subsection 4.3.2:

$$P_u = \sum_{i=1}^{n-1} w_i P_{u-1} + w_n P_f, \tag{4.4}$$

where n is the number of shapes in the past, including current detection and w_i are their respective weights, which are normalized. P_{u-1} and P_f are, respectively, the parameters in previous detections and the current parameter after fusion.

On the other hand, if the current shape has a type that is different from the one that appears most of the time, it is changed to that class of primitive. As for the parameters, it will be the same as the previous P_u from that type.

In this step, GS-IST also evaluates shapes that were not detected on the current keyframe but appeared previously. It recovers these shapes with the same parameters from the last appearance. The history data will be updated using the average distance of the recovered shape, but it will not have a primitive class associated at this particular moment. This shape will eventually disappear when not detected anymore because there will be no class of primitive that appear in more than 50% of the time.

4.3.5 Reliability Computation

At this point, GS-IST has a set of detected shapes and it is necessary to determine which of them are reliable. This reliability computation is based on a geometric and statistical evaluation.

4.3.5.1 Geometric Analysis

Each shape has a history information since its first appearance, which is a list of each primitive it matched in the past keyframes. However, a given shape may have different classes over time due to imprecision or inconsistency. Therefore, to perform the geometric analysis, it is computed the weight w_c for each class of primitive that appears in the history data:

$$w_c = \frac{1}{\sum_{i=1}^{h} d_i}, \tag{4.5}$$

where h is the number of times that each class of primitive appears in the history data and d_i is the average distance of the points in that shape to the original point cloud.

The weights are normalized and the one with the maximum value is the dominant class. GS-IST judges shapes whose dominant primitives have a weight higher than 0.75 as reliable. On the other hand, it considers unreliable those in which all weights are smaller than 0.5. When the weight of the dominant shape is between these two values, its classification will be determined by the statistical analysis. If this evaluation shows that the detection class through history is random, the primitive will be unreliable. Otherwise, it will be set as reliable.

4.3.5.2 Statistical Analysis

GS-IST performs a runs test for randomness to determine if the estimation history is random (Sheskin 2011). Basically, this non-parametric test uses the expected value and standard deviation to estimate the minimum number of runs that a sample can have to be considered random. A run means a sequence of consecutive estimates of one particular class of primitive. However, it was decided to look at the history of classification as binary data because the convergence is faster. Therefore, it was denoted a $+$ for the first

primitive detected. Then, the sign is repeated if the shape class is the same as the previous one. Otherwise, it is inverted. For a 5% level of significance, the sample is random if the number of runs is greater than:

$$N(R) = \mu - 1.65\sigma, \tag{4.6}$$

where the expected value μ and the standard deviation σ for the total number of samples n are:

$$\mu = \frac{2n-1}{3}, \tag{4.7}$$

$$\sigma = \sqrt{\frac{16n-29}{90}}. \tag{4.8}$$

Table 4.1 shows the example of a history information with a sequence of four spheres, followed by one cylinder, one plane and then by two other spheres. There are $R = 4$ runs and $n = 8$ samples. In this case, the maximum number of runs for a nonrandom sample is $N(R) = 3.269$, which indicates a random detection.

Table 4.1: History of the estimated shape from a primitive. For each sample that represents a keyframe K_i, it was classified as plane (P), sphere (S) or cylinder (C).

	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8
Primitive	S	S	S	S	C	P	S	S
Label	+	+	+	+	−	+	−	−

4.4 Evaluation

GS-IST was implemented in C++ using OpenCV[2] and Efficient RANSAC as libraries. For this evaluation, it was compared GS-IST with Efficient RANSAC regarding precision, recall and $F_{0.5}$-Score. The choice of this metric instead of F_1-Score was because it highlights the precision, which is the focus of the method. The only public dataset found is designed to detect primitives based on single-view images (Xiao et al. 2012), which was not suitable for this incremental approach. Since it was not found any dataset that has the generating process of point clouds, it was created one with five different scenarios to evaluate semantic modeling and tracking, which is available for download[3]. This dataset has the RGB images, a text file containing the camera parameters and the rotation and translation for each frame, the list of keyframes and the point cloud generated on each keyframe. It has five scenes targeting distinct types of primitives and different numbers of keyframes, as seen in Table 4.2 and illustrated with some screenshots in Figure 4.7. The number of points in the last keyframe indicates how sparse are the point clouds. It is worth to mention that Efficient RANSAC does not track the primitives, it only detects the shapes at each keyframe because it is a batch-based approach.

It is possible to see in Figure 4.5 that GS-IST obtained 100% precision in all cases while Efficient RANSAC never achieves more than 82%. It was noticed that there are more wrong estimations in the initial keyframes, as seen in the chart on Figure 4.6. It uses the precision of *Case 1* over time to illustrate this behavior, which is expected since the point cloud has few points in the initial reconstructions. The geometric and statistical analyses are able to identify these early incorrect detections. For instance, in the first three keyframes of *Case 2*, the bottle in the right side is assigned as a sphere, then as a cylinder and later as a sphere again because of the small number of noisy points. For the Efficient RANSAC, the bottle is incorrectly estimated as a sphere twice in three consecutive detections. Using this tracker, the bottle is assigned to the

[2] Available at `http://opencv.org/`

[3] Available at `https://github.com/rarrafael/vSLAM-dataset`

Table 4.2: Details of the dataset used for the evaluation, in which P, S and C stands for Plane, Sphere and Cylinder, respectively.

Test Case	Number of Frames	Number of Keyframes	Number of Points in Last Keyframe	Primitives in Scene
Case 1	1,660	31	16,698	P, S, C
Case 2	1,346	24	3,874	C
Case 3	849	20	4,257	S, C
Case 4	405	7	2,781	P
Case 5	499	17	4,445	P

same primitives in the first three keyframes but, each one has a weight based on the geometric analysis. After normalization, the weights of detection history are 0.186 (sphere in the first keyframe), 0.537 (cylinder in the second keyframe) and 0.277 (sphere in the third keyframe). Thus, for GS-IST, the bottle will be assigned as a cylinder because its weight is higher than the 0.463 of the sphere.

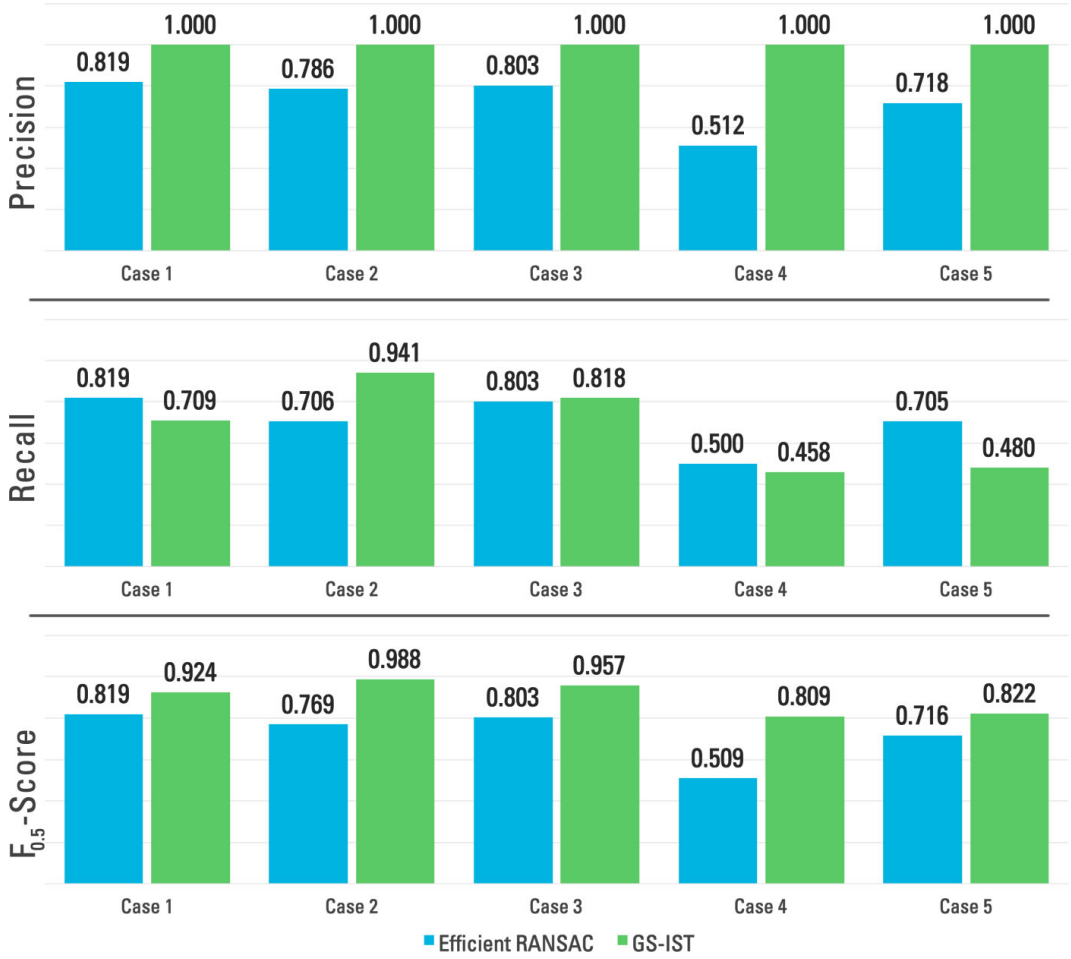


Figure 4.5: Comparison of precision, recall and $F_{0.5}$-Score between Efficient RANSAC (Schnabel et al. 2007) and GS-IST.

Regarding the recall, Figure 4.5 displays that GS-IST is worse than Efficient RANSAC in three of the five cases. This happens because Efficient RANSAC outputs twice more shapes on average than GS-IST, even though some of them are incorrect. Thus, this method compromises recall in order to be entirely sure that the most reliable shapes are selected. Using the same bottle as an example, in the third keyframe the cylinder weight is 0.537, which is below the reliability threshold of 0.75. However, since it is above the 0.5 unreliability mark, it is performed a statistical analysis to verify the randomness of this detection. According to Equation 4.6,

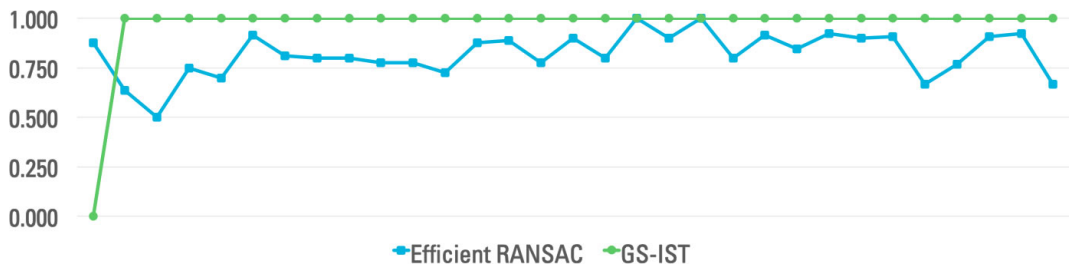


Figure 4.6: Precision over time of Efficient RANSAC (Schnabel et al. 2007) and GS-IST on *Case 1*.

this estimation history is random and the shape is assigned as unreliable.

Concerning $F_{0.5}$-Score, Figure 4.5 shows that GS-IST presents a better result in all cases. The most significant improvement is in *Case 4*, which is very challenging because the reconstruction is very noisy. This evaluation indicates that the restriction imposed improved the precision, but with the cost of having few shapes detected. Therefore, it is possible to adjust the parameters to have more primitives and decrease the precision. These changes will depend on the target application. Table 4.3 provides some examples of possible modifications to make and the outcome for *Case 1*.

Table 4.3: The influence of modifications in GS-IST on the final precision and recall in *Case 1*.

Condition Changed	Precision	Recall
Remove geometrical analysis	-1.409%	+1.846%
Remove statistical analysis	-1.409%	+3.139%
Double elimination thresholds	-0.704%	+2.602%

Figure 4.7 compares the results of both approaches in each test case. It shows situations in which the same object was detected as distinct shapes at different moments by Efficient RANSAC (rows 1 and 4). This phenomenon does not happen with GS-IST (rows 2 and 5). The detected shapes are represented by the projection of the input points used to compute the primitive. Rows 3 and 6 displays one view of the input point cloud in red and some of the estimated shapes with GS-IST in blue. From the last row, it is possible to see how challenging is *Case 4*. Although the books are aligned in real life, the points from the left one are not aligned with the other two.

4.4.1 Metric Evaluation

Case 1 has a chessboard pattern, which means that the reconstruction can be calibrated to the metric scale. This allowed an accuracy evaluation of object pose and parameters for this case in particular since there is no such pattern in the other cases. Considering the absence of ground truth for pose estimation, it was measured the average distance of each input point to its projection on the estimated primitive to assess how close they were. Figure 4.8 compares this distance over time between Efficient RANSAC and GS-IST. The leap in the distance is expected due to the error accumulation of the SLAM method. The average distance was 2.925 ± 0.370 mm for GS-IST while for Efficient RANSAC it was 3.247 ± 0.611 mm. It is worth to mention that this accuracy depends on the quality of the map. In this case, the error accumulation of the SLAM method was not precisely measured but it was around 15 mm, which is compatible with the error of the other systems evaluated in this Ph.D. thesis.

Concerning the parameter accuracy, it was used the radius of the globe and the wipe container, which are 50 and 40 mm respectively. The average radius of the sphere detected with the globe points was 43.147 ± 0.318 mm, resulting in an error of 6.853 mm. As for the cylinder estimated based on the wipe container, the 33.723 ± 1.001 mm radius is 6.277 mm smaller than the real object. Figure 4.9 shows that in the first detection

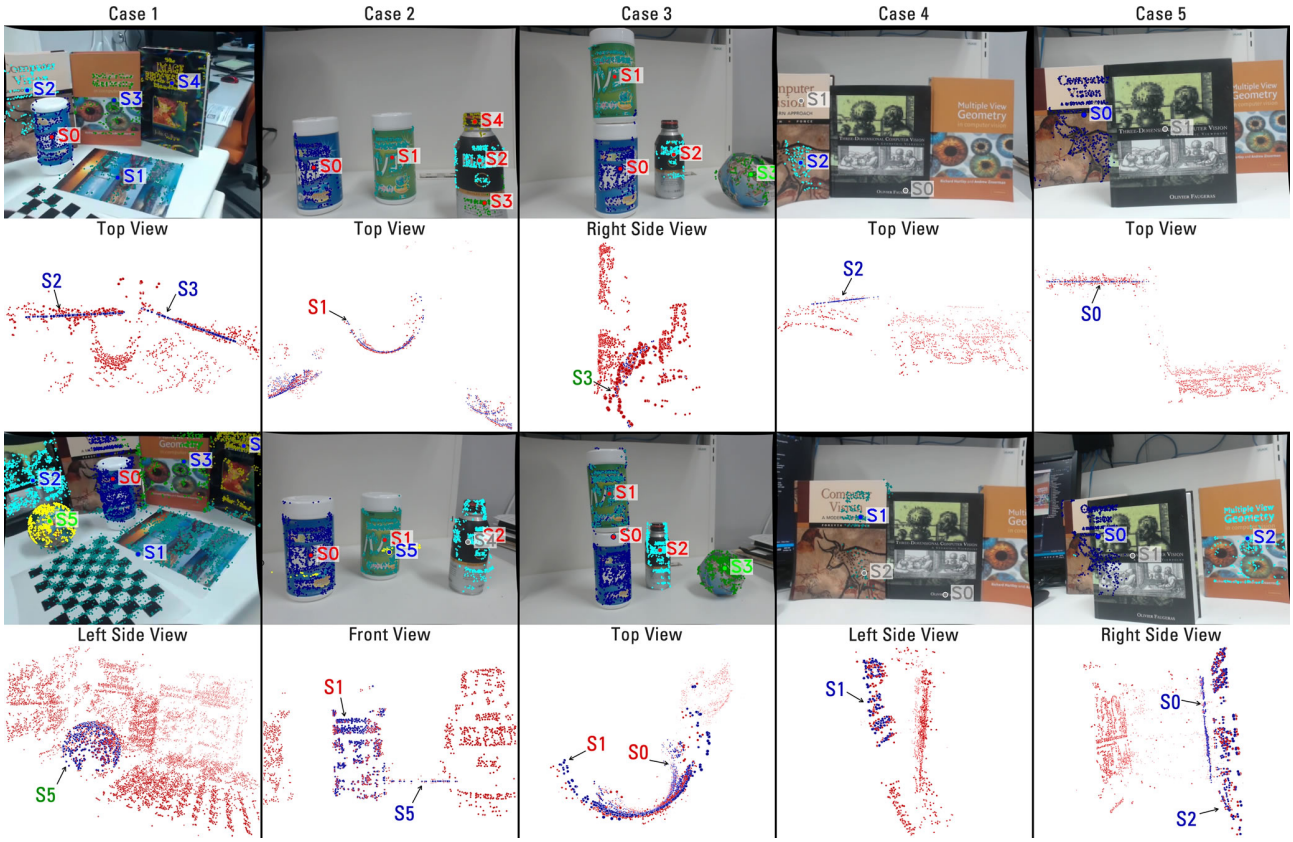


Figure 4.7: The first and third rows show the result of GS-IST on each test case. Blue labels represent planes, green ones are for spheres and red for cylinders. The second and fourth rows show one particular view of the input point cloud (in red) and some of the estimated primitives (in blue)[4].

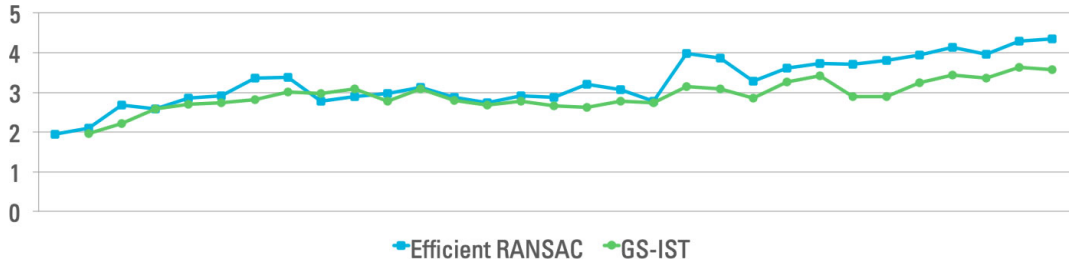


Figure 4.8: Average distance in millimeters of each input point to its projection on the estimated primitive over time for *Case 1*.

the cylinder radius is 31.051 mm and it gradually increases closer to the actual measurement over time, ending with 35.952 mm. These are the largest and smallest error in comparison with the ground truth for both the cylinder and the sphere. It can be credited to parameter update over time and the increase in the number of points that, even with the error accumulation, adds more data for the shape extractor. The sphere radius, on the other hand, decreases around 1 mm from the first to the last keyframe, going to the opposite direction of the actual radius. This can also be credited to error accumulation.

4.4.2 Runtime Evaluation

Concerning the computational cost, Efficient RANSAC takes on average 25.474 $\pm$ 10.380 ms to estimate the primitives in a computer with a Core i7-6820 (2.70 GHz) and 16GB of RAM. The other steps combined run in 14.995 $\pm$ 3.019 ms on average. The bottleneck is the *shape fusion* step, which takes 9.982 $\pm$

[4]A video with this result is also available at `https://goo.gl/5RGrYm`

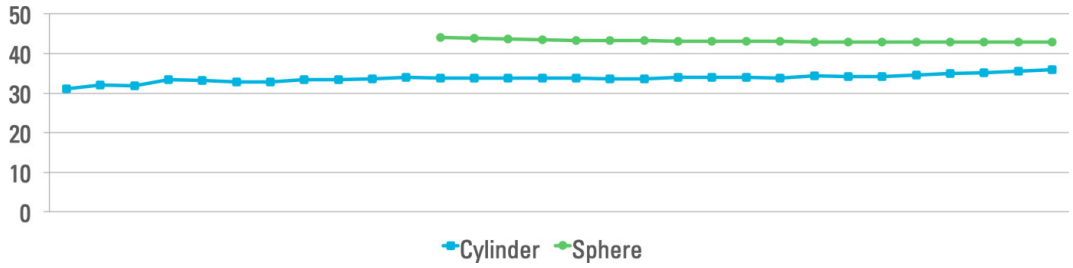


Figure 4.9: Error in millimeters over time of the cylinder and the sphere radii detected in *Case 1*.

2.957 ms of that time. It is worth mentioning that the execution time is related to the number of input points. Therefore, the measurements, which are the averages of all five test cases, were normalized to a group of thousand points.

4.4.3 Segmentation Evaluation

It was also evaluated how GS-IST segments the point cloud, which is a natural outcome of semantic modeling. Several objects have the form of the basic primitives tracked with this method. Looking at an average from all five test cases, 70.85% of the points can be assigned to a plane, sphere or cylinder. Even though the dataset deals with scenes designed with this type of primitives, the number is similar to the study that claims that 78% of all elements in an industrial scenario can be modeled using these three shapes (Huang et al. 2013a). Moreover, the chart on Figure 4.10 shows that only 6.30% of the remaining points were not labeled as any primitive. The other 22.85% points come from primitives that were discarded because they were unreliable.

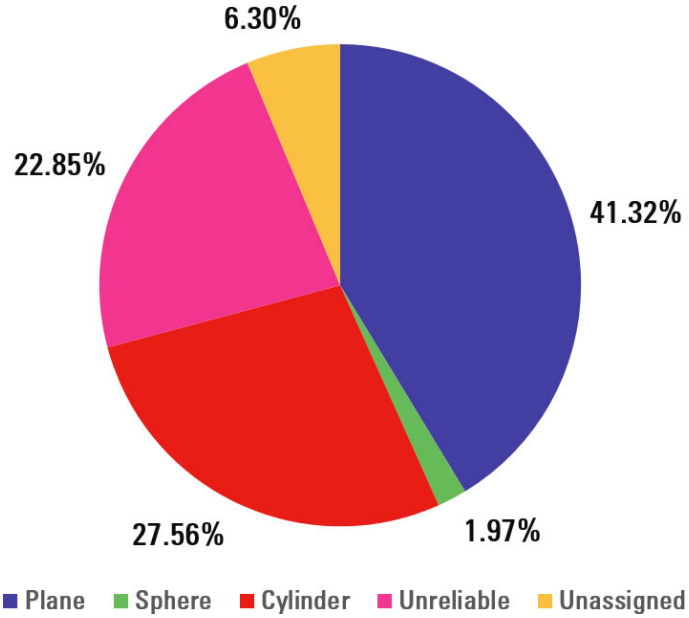


Figure 4.10: Percentage of points that were labeled to each primitive.

4.4.4 Point Cloud Representation Evaluation

Finally, it was compared the scene representation using the point cloud and the modeled primitives. The scene is usually overrepresented when it is described using the points because there are many redundant points. It was measured the memory necessary to represent the reconstruction of each test case using the point cloud and it was compared with the description of the same map using the data structure of the primitives modeled with GS-IST. Figure 4.11 shows this difference in KB between then, ordered by the number of points

from the reconstructed map. The most significant difference is in the last keyframe of *Case 1*, which has 16,302 points. Using the point cloud requires 8.69 times more memory than describing the same map using the six detected primitives plus the 3,915 unreliable and unassigned points.

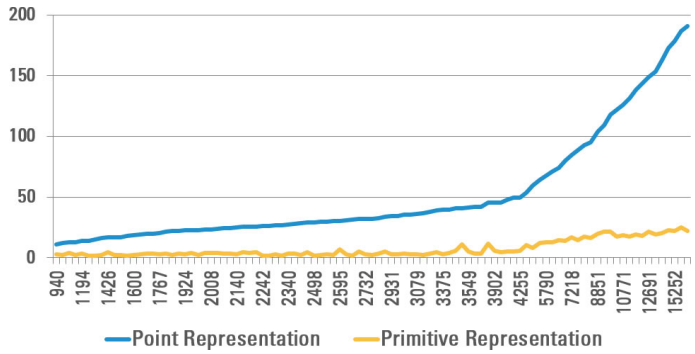


Figure 4.11: Memory (in KB) required to describe a scene using the point cloud and the data structure of the detected primitives for *Case 1*.

Moreover, describing a scene using points commonly results in over and underrepresentation at the same time. For instance, considering only the cylinder detected in the last keyframe of *Case 1*, it can be noticed that there are more points than necessary to describe the textured front side but none to represent the back side. Thus, it is possible to use much less information to define this shape while filling the missing parts. Using this cylinder as an example, it is necessary 24 times more memory to describe it using the points than using the data structure of the detected primitive.

4.4.5 Proof of Concept

It is intended to see how GS-IST responds to an application that benefits from having semantic knowledge of the environment. It was developed Shape Hunt, a system to help children to identify some of the primitives they are learning in school. The idea of this proof of concept is that it draws a shape and he/she has to find real objects with the same geometric form. Since GS-IST has the scene map and can track the objects in the scene, it can identify all selected shapes and the child always has to find a new one. Figure 4.12 shows this proof of concept.

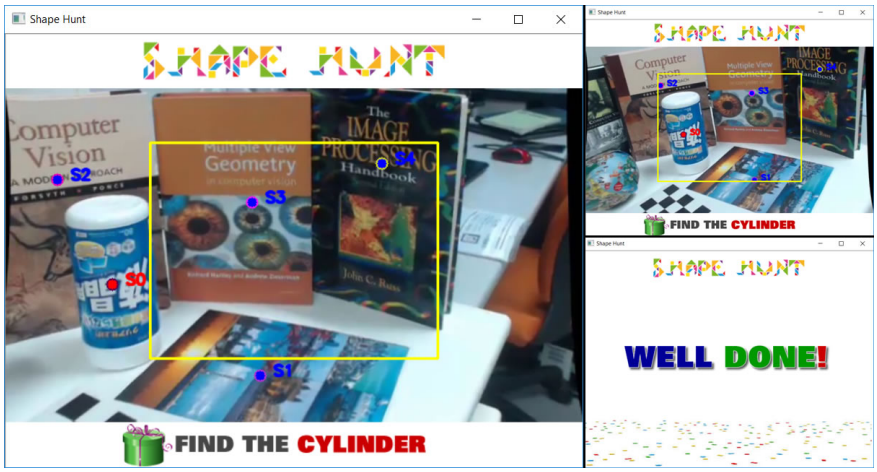


Figure 4.12: The application indicates the shape the user has to find (left image). When the correspondent shape is centered (top-right), it gives a positive feedback (bottom-right) and moves to the next primitive[5].

[5]A video with this result is also available at `https://goo.gl/d76Sfw`

For this proof of concept, it was used the images from the dataset as input for this application. The scenes are limited to a small workspace, which was sufficient for this test. The application behaves as expected. Each primitive was detected only once and none was misclassified.

4.4.6 Evaluation on Dense Point Cloud

Although designed for sparse point clouds, the foundations of this tracker were adapted to work in a dense one. For this situation, it was also used the generating process of the point cloud to create a semantic map. The incremental process is also explored to improve the precision and stability of the shape detection over time. Moreover, the history of the fusing process is used to reduce the influence of error accumulation in the RGB-D SLAM system. Additionally, since the scale is available, it is possible to have the metric information from the modeled primitives. The evaluation showed that error on measuring the radius of spheres and cylinders varies between 0.1 and 4.6 millimeters and that it is difficult to model spheres that have the radius smaller than 30.0 mm. Similar to GS-IST, the execution time is proportional to the number of points. However, the modeling process was executed in less than 100 milliseconds, on average, in a standard desktop. The complete details about the method adaptation for dense point clouds and its evaluation were published in (Olivier et al. 2018).

5
Mobile Evaluation of Incremental Semantic Tracking

Mobile devices are reducing the gap to desktop computers in terms of processing power and memory space. From 2009 to 2015 the desktop CPU clock frequency increased by 33% while the mobile CPU clock grew 252% in the same period (Halpern et al. 2016). This increment in the processing capabilities of mobile devices allowed the development of more complex algorithms, including tracking techniques, which is one of the foundations of augmented reality. This can be linked to the improvement and popularization of augmented reality solutions for such devices.

Some of these recent advancements in tracking techniques involves removing the necessity of any external marker, improving the execution time to achieve real-time performance and having a more stable tracking that is not harmed by jitter or drift. Most of them are thanks to the improvement of the device's sensors that allowed the development of visual-inertial trackers. However, less progress was made in regard to extracting any kind of semantics from the 3D map of the scene. This chapter focus on the evaluation of the incremental semantic tracker from Chapter 4 on mobile devices, showing that it is feasible to extract and track basic primitives on such platforms.

5.1 Semantic Modeling on Mobile Devices

As seen in Chapter 2, tracking techniques on mobile devices had a substantial improvement until 2015. Not only on the number of publications but, most importantly, on tracking capabilities. Since then, the development of visual-inertial techniques become popular and received a lot of attention from the community, which comes in different flavors. There are distributed approaches that automatically selects between a fast visual tracker based on optical flow and a visual-inertial odometry depending on certain quality criteria (Piao et al. 2017). Others are improving the traditional visual-inertial methods, such as adding a lightweight tightly-coupled fusion approach which integrates nonlinear optimization and loop detection (Li et al. 2017). More recently, (Solin et al. 2018) proposed a probabilistic inertial-visual odometry technique that is robust to occlusion and large-scale navigation without the need to perform loop closures. They achieve that by propagating the uncertainty of the measurements to every aspect of the tracking procedure, which includes the camera motion, the geometry and the minimization problem. However, none of these techniques are able to retrieve any type of semantics from the scene.

The recent development of machine learning techniques allowed the possibility to extract semantics from the image based on network architectures designed to deal with constraints of such devices (Sandler et al. 2018). For instance, (Tobías et al. 2016) uses deep learning to perform domain-specific object recognition. They achieve high classification and near real-time execution time running on an iPad. In another example, a Convolutional Neural Network (CNN) was incorporated to a mobile augmented reality application to perform object detection (Rao et al. 2017). They use the inertial sensors and the GPS to track an outdoor environment, detect geographical landmarks using this CNN and render their information with 3D coherence due to the inertial odometry. DeepCham (Li et al. 2016) use CNN to recognize objects as well. They rely on distributed

redundancy, enhanced bounding box and generic deep model to facilitate the creation of training instances. There are examples of geometric techniques for object recognition too. One example is (Chen et al. 2015), which created a distributed approach that uses a cache of frames scheme to improve the method performance. They are able to identify from faces to traffic signs.

Although most of the studies find semantics from images, it is also possible to retrieve these information from point clouds. This is more complex to achieve with mobile devices due to the overload of data to process, especially when dealing with dense point sets. However, recently some systems were developed that are able to extract information from such data. These techniques usually generate the point cloud using Google Tango devices, such as (Runceanu et al. 2017) that apply an iterative RANSAC approach to segment planes and model walls of indoor structures. Using a Tango as well, (Sankar et al. 2017) also detect planes and model indoor environments. Nevertheless, they also detect planes to segment more complex objects and based on their set of points and visual appearance the authors are able to match the object with a 3D model available in the library.

Extracting semantic information using sparse point clouds should be simpler when concerning the amount of data to process. However, this is the same reason why this task is more challenging. Nevertheless, both Google's ARCore and Apple's ARKit incorporate semantic modeling as part of their scene understanding feature. They both extract planes based on the scene 3D reconstruction for creating surfaces in order to have a more stable positioning of the virtual objects.

5.2 Mobile Implementation

In order to evaluate how GS-IST would perform running on mobile devices, the technique presented in the last chapter was ported to the Android platform. Since it was developed in C++, this facilitates the port using the efficient Android architecture in a similar process used to port LBF and STAM to mobile devices. It was also necessary to compile the libraries used in the project to Android: OpenCV and Boost[1]. Efficient RANSAC was treated as a library as well but the compilation process was a little bit different due to the fact that it was added to the project in order to facilitate modifications in the source code.

5.2.1 Evaluation

The two most critical aspects of mobile devices are the energy consumption and the temperature. Although the number of cores and CPU clock have increased lately, the processors' architecture compromise on speed to be more efficient on these two facets (Reiner 2012). Most of the studies aiming mobile devices focus their evaluation only on execution time along with any qualitative assessment that is adequate for the proposed method. Using the 10 papers cited in the previous section and the 25 relevant studies listed in Table 2.3 at Chapter 2 as a sample, 62.9% measured execution time and 45.7% evaluated only this criteria. Energy consumption was evaluated in 17.1%, RAM memory usage in 8.6% and 28.6% performed qualitative assessments alone. Only (Li et al. 2016) evaluates execution time, energy consumption and memory usage. However, they did not detail the methodology used to perform these measurements. These three criteria were used to evaluate GS-IST along with CPU load, which is a good indicator of the potential of parallel execution of a certain application.

This evaluation was executed in devices with different capabilities and from distinct manufacturers, which is important to assess how GS-IST performs in dissimilar conditions. The chosen devices were the Samsung Galaxy S8 and the ASUS ZenFone 3. They were selected using ARCore as a reference. The

[1] Available at `https://www.boost.org`

Galaxy S8 is in the list of the supported devices[2] while the ZenFone 3 was selected to stress GS-IST since it has a configuration inferior to those in that list. Table 5.1 show some of the technical specifications of these smartphones. All the tests were executed with the device fully charged, on airplane mode, with all other applications closed and connected to the computer via the USB cable. The only exception was the energy measurement in which the device was disconnected from the computer.

Table 5.1: Summary of the technical specifications of the devices used for evaluation.

Features	Galaxy S8	ZenFone 3
Android Version	8.0 (Oreo)	7.0 (Nougat)
Display	5.8" (1440 x 2960)	5.5" (1080 x 1920)
Chipset	Qualcomm MSM8998 Snapdragon 835	Qualcomm MSM8953 Snapdragon 625
CPU	Kyro Octa-core (4x2.35 GHz and 4x1.9 GHz)	Cortex-A53 Octa-core (2.0 GHz)
Memory	64 GB, 4 GB RAM	32 GB, 3 GB RAM
Battery	3000 mAh	3000 mAh

Regarding the dataset, it was used the same five scenes generated in the previous chapter. The images and pose files were stored in the device and loaded on every frame. The same happened for the map files, which were loaded on every keyframe. Since the codes are identical, precision and recall are equal to the ones presented in the last chapter. Figure 5.1 shows a few keyframes of GS-IST running on both phones.

5.2.1.1 Execution Time

The execution time is proportional to the number of points processed and all time measurements, which are the averages of all five test cases, were normalized to a group of thousand points. Figure 5.2 shows that the average execution time of GS-IST on ZenFone 3 is 9.9 times slower in comparison with the desktop implementation and it is 8.5 times slower on Galaxy S8. For this test, the same desktop computer with Core i7-6820 (2.70 GHz) processor and 16GB of RAM was used. The *shape fusion* was slower than the average on mobile devices, being the desktop implementation 16.5 (ZenFone 3) and 15.0 (Galaxy S8) times faster.

The CPU load is an important measure because it indicates how much room the GS-IST leaves to perform other processing, such as the SLAM technique. For that evaluation, it was used Qualcomm's Trepn Profiler[3]. This application, available in the Play Store, samples the desired information in a constant time interval. In this test, both the CPU load and the Normalized CPU load were sampled every 100 milliseconds. The operating system imposes a limit on how much processing an application can use. The CPU load represents how much of that limit is being used by the application while the Normalized CPU load indicates how much processing is being used in relation to the total processing power of the device.

It is possible to observe in Figure 5.3 that GS-IST presents some execution peaks. These apexes coincide with the keyframes, which are moments in which the technique extracts the primitives. The maximum normalized load for the ZenFone 3 was 90% of CPU, similar to the 89% value for the Galaxy S8. The average normalized load was also similar for both devices, in which the Samsung device was 30.545% $\pm$ 25.119% of Normalized CPU load while the ASUS mobile phone presented 27.128% $\pm$ 16.353%. This happens because for most of the time the execution is in between keyframes, a moment in which the device is not processing much data. The median of the Normalized CPU load is a numerical indication for that and it was 19% and 21% for the Galaxy S8 and ZenFone 3 respectively. However, there was a difference in the average CPU load, which was 39.283% $\pm$ 27.131% for the Samsung phone and 46.483% $\pm$ 25.040% for the ASUS device. Smaller

[2]Available at `https://developers.google.com/ar/discover/supported-devices`
[3]Available at `https://developer.qualcomm.com/software/trepn-power-profiler`

Figure 5.1: Results of GS-IST running on a Samsung Galaxy S8 and an ASUS ZenFone 3. Blue labels represent planes, green ones are for spheres and red for cylinders[4].

differences between the CPU load and the Normalized CPU load suggests that the device is allowed to use the full potential of the processor. In that case, this value was 8.738% for the Galaxy S8 but it was 19.355% for the ZenFone 3.

[4]A video with this result is also available at `https://goo.gl/A6T51d`

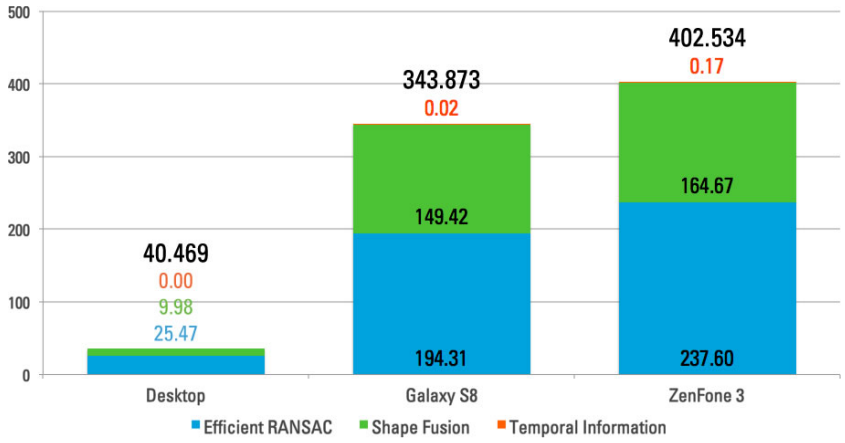


Figure 5.2: GS-IST execution time in milliseconds divided by stages on desktop and two different mobile devices.

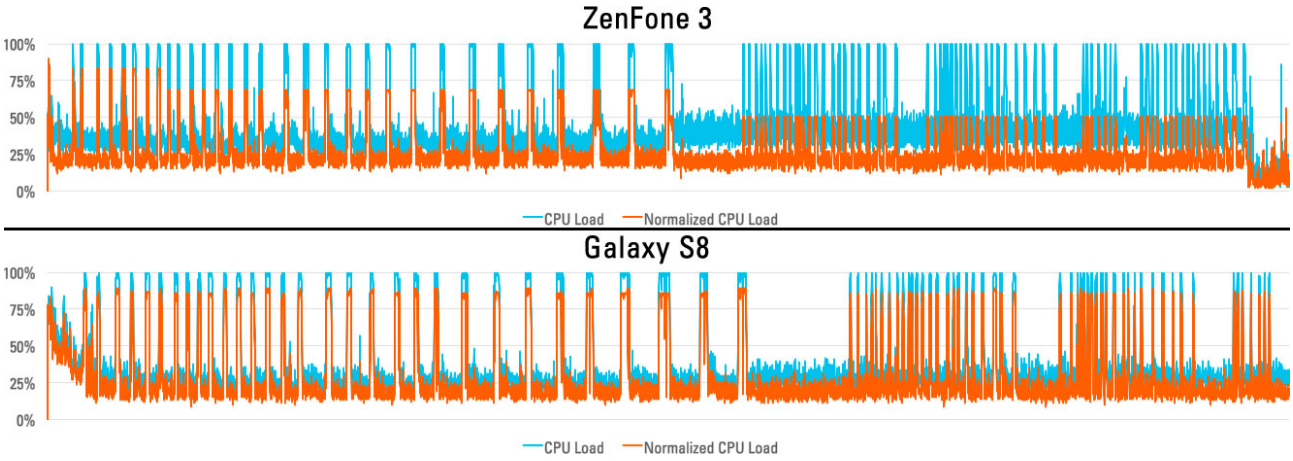


Figure 5.3: CPU load and Normalized CPU load over time of all five test cases on a ZenFone 3 (top) and Galaxy S8 (bottom).

5.2.1.2 Energy Consumption

The most precise method to evaluate energy consumption is by using external instruments that can directly measure the current drained by the device. However, these equipment require opening the device to be attached to the physical battery, which is difficult for most smartphones nowadays since their batteries are not easily accessible. An alternative is to use profiler tools that use the battery API to assess the voltage and the state of charge at certain intervals. This procedure is much more accessible but it is not as accurate as using these external instruments. The latter approach was selected for this evaluation and Trepn Profiler was also used for this task. Qualcomm's tool has an accuracy of 99%, which is reported to be one of the highest for such profilers (Hoque et al. 2015).

As expected, Figure 5.4 shows that energy consumption on both mobile devices follow the same pattern of the CPU load since more energy is required in those most computational-intense moments. The average consumption on the ZenFone 3 was 1.776 $\pm$ 1.162 W every 100 ms. Since the battery capacity of this device is 3000 mAh with 3.85 V, this means that this device could run GS-IST for 5 hours and 12 minutes before drain all the battery when it is fully charged considering an energy efficiency of 80% (Valoen et al. 2007). The Galaxy S8 battery has the same characteristics (3000 mAh and 3.85 V), which determines that its average 2.271 $\pm$ 1.998 W consumption would drain a fully charged battery in 4 hours and 04 minutes.

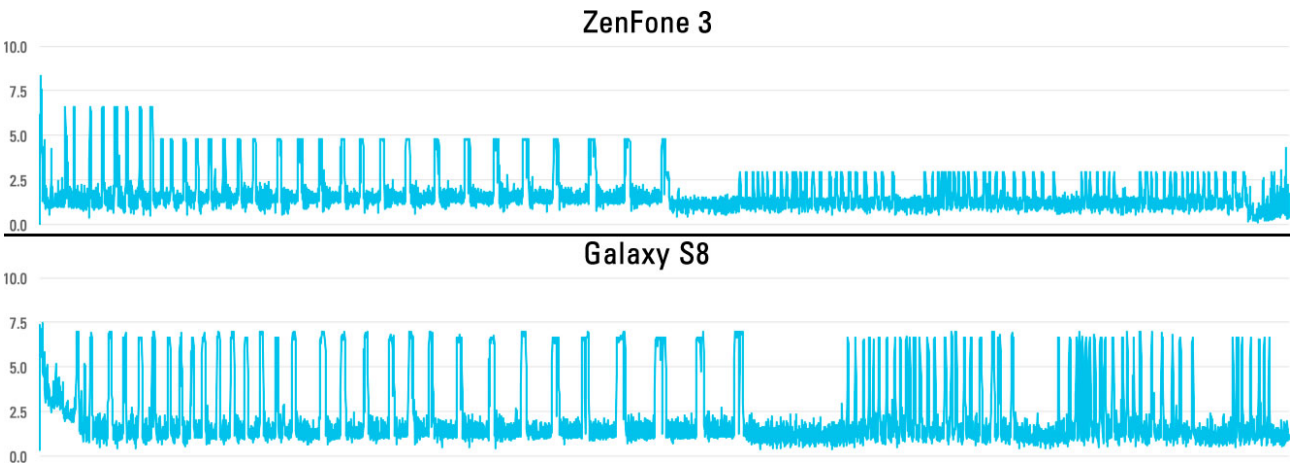


Figure 5.4: Energy consumption (in W) over time of all five test cases on ZenFone 3 (top) and Galaxy S8 (bottom).

5.2.1.3 Memory Usage

Concerning the memory usage, two measurements can be done. One is the storage space the sample APK requires when installed and the other is the RAM memory it uses when running. The former value can easily be found in the device settings. In the ZenFone 3, GS-IST used 45.14 MB of the storage space while in the Galaxy S8 it occupied 47.38 MB.

To evaluate the latter it was used the Android Profiler available on Android Studio, which builds a chart with the RAM memory usage as the application is executed. Similar to the previous evaluation, Figure 5.5 shows that the RAM memory has some peaks when extracting the primitives. For the ZenFone 3, the moment with most memory usage is in the 28th keyframe of *Case 1* with 195.39 MB, which represents 6.4% of the device total memory. When not processing the keyframes, the sample app consumes between 26.14 MB and 38.19 MB.

As seen in Figure 5.6, the memory usage for the Galaxy S8 is similar, although with higher absolute values. The memory peak was 322.97 MB in the 25th keyframe of *Case 1*, being 7.9% of the phone RAM memory. The memory consumption when not extracting primitives was above 119.07 MB and below 151.16 MB.

5.3 Discussion

From Figure 5.2 it is possible to see that GS-IST does not run in real-time on any of the devices it was tested. It is especially true when the number of points increases. However, this approach uses less than a third of the CPU power regardless of the device. This means that GS-IST and a SLAM technique can run in different threads in order that the semantic modeling system and the SLAM method can interact with each other without compromising their performance. Since this mobile version is running on Android devices, it is possible to use ARCore as a SLAM Method and extract the primitives from the point cloud the SDK generates. Nevertheless, it is necessary to perform an evaluation to see if the ARCore map is too sparse. Additionally, there is room for optimization in the implementation.

Moreover, Figure 5.3 shows that Android 7.0 on ZenFone 3 imposed a more restrict limit on how much processing GS-IST could use. In fact, that limit decreased over time. For instance, GS-IST could use 90% of the CPU at the beginning of the execution and that boundary decreased to approximately 75% in the 9th keyframe of *Case 1* and to 50% from *Case 2* and beyond. On the other hand, Android 8.0 on Galaxy S8

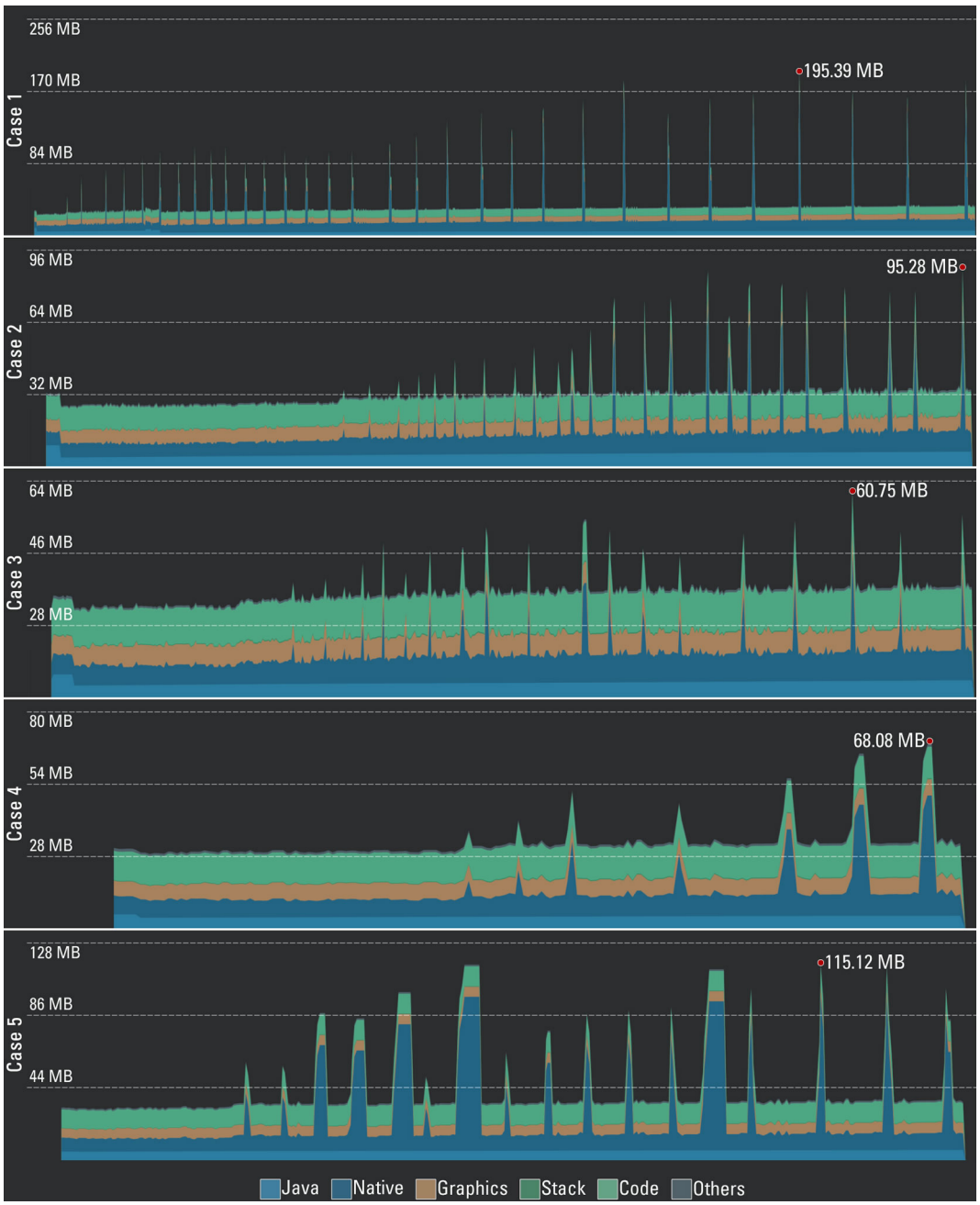


Figure 5.5: Memory usage over time of GS-IST running on ZenFone 3. Each test case has a different time scale.

allowed GS-IST to use around 90% of the CPU from beginning to end.

Regarding energy consumption, the 4 hours GS-IST needs to drain the battery in the worst case is sufficient to use this technique without having extra concerns about the battery. Moreover, considering the habits and the average time users spend on mobile devices (Hacker Noon 2017), it is unlikely that a person would use a specific app for a period so long. In order to put this in perspective, Table 5.2 compares the time other common applications take to fully discharge the battery, such as watching a 1080p video, navigating using Google Maps and playing FIFA Soccer. Besides the video activity, GS-IST has an energy consumption similar to the other applications.

The evaluation showed a noticeable difference in RAM memory usage between both devices. The ASUS phone used approximately 100 MB less memory than the Samsung one. It is possible to see in Figure 5.5 and Figure 5.6 that graphics uses much more memory in Galaxy S8 than in ZenFone 3. The graphics are

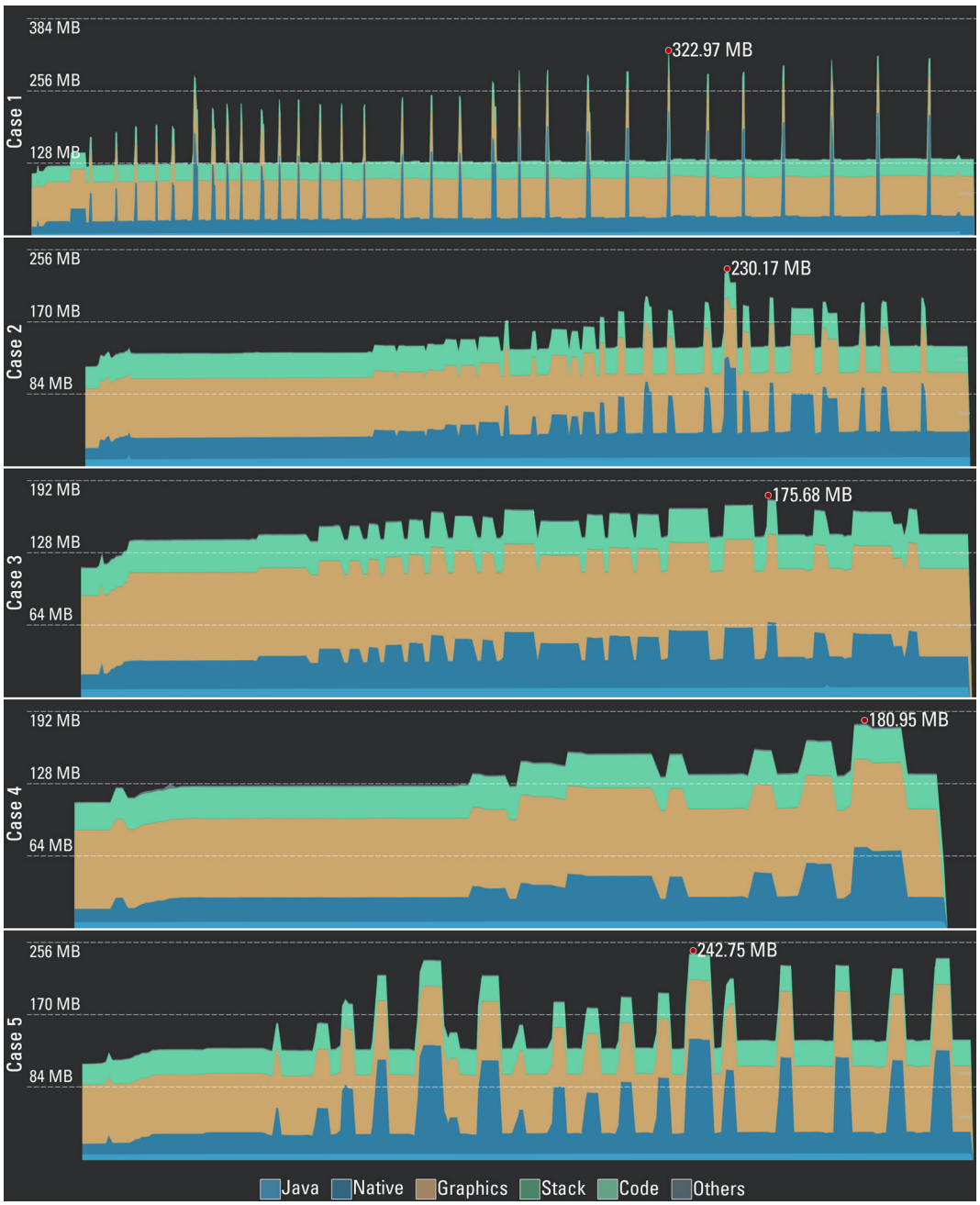


Figure 5.6: Memory usage over time of GS-IST running on Galaxy S8. Each test case has a different time scale.

Table 5.2: Time that different applications take to fully drain a battery fully charged on ZenFone 3 and Galaxy S8 devices.

Application	ZenFone 3	Galaxy S8
1080p video	8h07min	6h31min
Google Maps (using 4G network)	6h32min	4h29min
GS-IST	5h12min	4h04min
Fifa Soccer (using WiFi connection)	5h09min	4h47min

responsible for more than 70% of that difference. It was not found any details for that. However, based on observation using other devices, one possibility is that the ASUS device delegate the rendering activity to the GPU, therefore graphics-related structures uses the GPU memory.

The memory usage of GS-IST was not a concern since, in the worst case, it used less than 8% of

the total RAM memory of the device. This is due to the fact that modern smartphones have a fair amount of RAM memory. For them, replacing the representation from point clouds to primitives does not have any impact on the execution time. However, this change in representation can be important in some situations. Complex algorithms can use a lot of memory and perform several computations for each element in the scene, which can overload the powerful devices even for a sparse map. Tracking optimization methods have that characteristic. In fact, some developers reported that the current version of ARCore (v1.2.1) can run out of memory when it has to perform bundle adjustment with a map whose size is that of a large room. Rendering algorithms are another case in which having a large number of elements can cause the usage of most of the device's resources. Therefore, a more efficient representation can be very helpful in circumstances like that.

6
Final Considerations

Tracking on mobile devices evolved a lot since this Ph.D. started four and half years ago. At that time the phones were not so powerful, which raised big concerns regarding the processing power and memory capabilities required to run real-time tracking of such devices. The perception was that there was a growing interest on having tracking techniques on mobile devices but a few were able to track more than a planar template. The systematic mapping conducted confirmed that perception. Moreover, it also showed that until 2015, most of the works used the device's sensors to compute 2D or 2D + θ pose on location-based applications. Nowadays, the devices improved in great extension and new techniques have emerged, which make possible the development of increasingly popular SDKs that are able to perform real-time tracking with 6D poses.

Based on the findings of the systematic mapping, it was created experimental scenarios aiming to evaluate different tracking approaches for mobile devices. One of these experiments was the proposition of a method to assess the Google Tango platform aiming to establish a reference of the state-of-the-art trackers. It was observed that the motion tracking errors were around 6 and 14 centimeters for small and large indoor scenarios, which is suitable to provide a good user experience, including for augmented reality applications. This experiment was followed by another one that evaluated the use of different forms to develop computer vision systems on mobile devices, such as using parallelism and distributed approach. It indicated that each solution has strengths and limitations depending on the situation. However, native development was the most efficient on average. It was performed experiments to evaluate different tracking techniques that had the potential to be suitable for mobile devices. The first was a face tracking technique using machine learning and local binary features, which was adapted to consider the characteristics of mobile devices, such as camera orientation. The second was a SLAM technique that was originally developed in desktop and ported to a Tango tablet device. The mobile version presented some issues regarding performance, especially when it needs to process a large number of data in situations like bundle adjustment.

One of the main lessons learned in this Ph.D. study was the importance of finding high-level semantic information from a scene. Therefore, it was developed a technique that detects and tracks primitives, called GS-IST. This method uses the generating process of sparse point clouds of visual SLAM systems and applies geometrical and statistical analyses to incrementally estimate and track planes, spheres and cylinders. The evaluation indicated that GS-IST improved precision in all test cases, which outperformed existing methods in this criteria. The developed approach focuses on precision and for that, it compromises recall to assure we have the correct shapes. However, we can modify the parameters to increase recall when necessary. Additionally, this technique was ported to the Android platform and evaluated to assess how it performed running on mobile devices. The evaluation showed that the mobile version is slower when compared with the desktop implementation but it can be executed on a separate thread of the SLAM technique because the CPU load is not so high. Finally, the energy consumption and memory usage were not a concern.

6.1 Future Work

There is still some work that needs to be done after this Ph.D course. One is to investigate the possibility of integrating GS-IST with Google's ARCore, which would allow the development of use cases based on real-world problems that can benefit from having a semantic knowledge of the scene. One way to do that is using a well-structured methodology that is based on the design thinking theory that combines interdisciplinary teams to conceive innovative solutions (Roberto et al. 2016d). There is a team at Voxar Labs creating bridges between academic research and innovative solutions with high impact and the goal is to team up with them to run an instance of their process.

After integrating the semantic tracker with ARCore, it is possible to create new test cases that allow the evaluation on more complex environments. These new scenes can be industrial factories that have several machines, mechanical workshops loaded with equipment, or warehouses, which have various shelves and boxes. Although very challenging, these scenarios have several objects that can be modeled with the primitives GS-IST detects. Therefore, this would allow stressing the technique in order to find points for further improvements.

Further activities include performing a more extensive evaluation of the accuracy of object tracking and its parameters. In order to accomplish that, it is necessary to have a dataset with ground truth pose and measurements. This can be achieved with the creation of other scenes that would include a chessboard pattern or other means to recover the scale. It is also possible to obtaining ground truth for object pose using markers tracked with libraries such as ArUco (Garrido-Jurado et al. 2014). An additional possibility is to create a synthetic case, downsample and apply noisy to the point cloud in order to simulate sparse reconstructions.

Another idea for future work is to use the semantic knowledge of the scene to improve the tracking results of the visual SLAM system. There are some ways to achieve this goal. One is to constrain the map 3D points during the bundle adjustment to move only over the surface of the shape it belongs, which would optimize the point cloud respecting the semantic structure and leading to a faster convergence. A different strategy is to optimize the map using the primitive parameters instead of the points. The idea is to save on computation by minimizing the error of a few parameters rather than do the same operation for hundreds of points that represent the same elements on the scene. This would have an impact on the scalability of the environment to be tracked.

6.2 Contributions

This Ph.D. research produced some contributions to the community:

- A systematic mapping that extensively cataloged and classified the area of tracking for mobile devices, providing a reference for new researchers in the field to have a quick overview of the area;
- A paper catalog adapted from an open-source system, which is an easy way to display the results of the mapping;
- An open-source paper crawler[1], which can help other researchers to gather scientific papers;
- An evaluation of different architectures aiming to efficiently develop tracking techniques on mobile devices;

[1] Available at `http://www.cin.ufpe.br/~rar3/tracking_sm/paper-analysis.tar.gz`

- A face tracker system that was part of an application developed in a project in partnership with a major mobile phone manufacturer;
- A SLAM technique that was developed and used to compete in the 2015 ISMAR Tracking Competition, winning the first place on the “On-Site Category: Level 3”;
- The method used to evaluate Google Tango, which is an easy way to perform preliminary evaluations on mobile motion trackers;
- A technique that uses geometric and statistical evaluation to incrementally perform semantic modeling and tracking of primitives on sparse point clouds;
- A dataset with sparse point clouds of primitives that is helpful to evaluate semantic modeling and tracking[2];
- The port of the incremental semantic tracker to mobile devices;
- A guideline to evaluate computer vision techniques on mobile devices.

6.3 Publications

Some scientific papers were published during this Ph.D. Seven were directly related to this study, as mentioned in the text. One was related to the systematic mapping (Roberto et al. 2016c). Three were about the experiments, one being the evaluation of Google Tango (Roberto et al. 2016a), other was the partial results of the experiment about the architecture of computer vision systems on Android (Lima et al. 2015) and the third one was a publication about the port of STAM to Google’s Project Tango (Araujo et al. 2016). Finally, three more papers were published concerning GS-IST (Roberto et al. 2017; Roberto et al. 2018; Olivier et al. 2018). Additionally, this Ph.D. study was selected to be presented and discussed with the computer vision community in the *PhD Forum* of the IEEE Winter Conference on Applications of Computer Vision (WACV) in 2018.

There were also four publications targeting tracking (Lima et al. 2017) or mobile devices (Lins et al. 2014a; Lins et al. 2014b; Lima et al. 2014) that are not directly related to this work but were important to gain experience in the Ph.D. topics. Additionally, there were seven papers regarding topics not directly related to this study (Mota et al. 2014; Silva et al. 2015b; Silva et al. 2015a; Mota et al. 2015; Roberto et al. 2016d; Silva et al. 2016; Roberto et al. 2016b).

[2] Available at `https://github.com/rarrafael/vSLAM-dataset`

Bibliography

Agarwal, S., Mierle, K., and Others. *Ceres Solver*. http://ceres-solver.org.

Almeida, D. R. O. d., Guedes, P. A., Silva, M. M. O. d., Silva, A. L.B. V. e., M. Lima, J. P. S. d., and Teichrieb, V. "Interactive Makeup Tutorial Using Face Tracking and Augmented Reality on Mobile Devices". In: *Virtual and Augmented Reality (SVR), 2015 XVII Symposium on*. 2015, pp. 220–226. DOI: 10.1109/SVR.2015.39.

Altman, D. *Practical Statistics for Medical Research*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, London, 1990. ISBN: 9780412276309. URL: https://books.google.com.br/books?id=v-walRnRxWQC.

An, J. H. and Hong, K. S. "Finger gesture-based mobile user interface using a rear-facing camera". In: *Consumer Electronics (ICCE), 2011 IEEE International Conference on*. 2011, pp. 303–304. DOI: 10.1109/ICCE.2011.5722596.

Ando, B., Baglio, S., Lombardo, C., and Marletta, V. "An advanced tracking solution fully based on native sensing features of smartphone". In: *Sensors Applications Symposium (SAS), 2014 IEEE*. 2014, pp. 141–144. DOI: 10.1109/SAS.2014.6798934.

Apple Inc. *ARKit - Apple Developer*. [Online; last access: 29-Dec-2017]. 2017. URL: https://developer.apple.com/arkit/.

Araujo, T., Roberto, R., Teixeira, J. M., Simões, F., Teichrieb, V., Lima, J. P., and Arruda, E. "Life Cycle of a SLAM System: Implementation, Evaluation and Port to the Project Tango Device". In: *Virtual and Augmented Reality (SVR), 2016 XVIII Symposium on*. 2016.

Arth, C., Wagner, D., Klopschitz, M., Irschara, A., and Schmalstieg, D. "Wide area localization on mobile phones". In: *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*. 2009, pp. 73–82. DOI: 10.1109/ISMAR.2009.5336494.

Bai, H., Gao, L., and Billinghurst, M. "Poster: Markerless fingertip-based 3D interaction for handheld augmented reality in a small workspace". In: *3D User Interfaces (3DUI), 2013 IEEE Symposium on*. 2013, pp. 129–130. DOI: 10.1109/3DUI.2013.6550212.

Bay, H., Tuytelaars, T., and Van Gool, L. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Ed. by A. Leonardis, H. Bischof, and A. Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

Bradski, G. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

Burgos-Artizzu, X. P., Perona, P., and Dollár, P. "Robust Face Landmark Estimation Under Occlusion". In: *Proceedings of the 2013 IEEE International Conference on Computer Vision*. ICCV '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 1513–1520. ISBN: 978-1-4799-2840-8. DOI: 10.1109/ICCV.2013.191. URL: http://dx.doi.org/10.1109/ICCV.2013.191.

Chen, D., Tsai, S., Vedantham, R., Grzeszczuk, R., and Girod, B. "Streaming mobile augmented reality on mobile phones". In: *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*. 2009, pp. 181–182. DOI: 10.1109/ISMAR.2009.5336472.

Chen, T. Y.-H., Ravindranath, L., Deng, S., Bahl, P., and Balakrishnan, H. "Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices". In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. SenSys '15. Seoul, South Korea: ACM, 2015, pp. 155–168. ISBN: 978-1-4503-3631-4. DOI: 10.1145/2809695.2809711. URL: http://doi.acm.org/10.1145/2809695.2809711.

Cheng, S., Asthana, A., Zafeiriou, S., Shen, J., and Pantic, M. “Real-time Generic Face Tracking in the Wild with CUDA”. In: *Proceedings of the 5th ACM Multimedia Systems Conference*. MMSys ’14. Singapore, Singapore: ACM, 2014, pp. 148–151. ISBN: 978-1-4503-2705-3. DOI: 10.1145/2557642.2579369. URL: http://doi.acm.org/10.1145/2557642.2579369.

Chon, Y., Talipov, E., and Cha, H. “Autonomous Management of Everyday Places for a Personalized Location Provider”. In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 42.4 (2012), pp. 518–531. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2011.2131129.

Cohen, J. “A coefficient of agreement for nominal scales”. In: *Educational and psychological measurement* 20.1 (1960), pp. 37–46.

DAQRI. *DAQRI Smart Helmet – DAQRI*. [Online; last access: 07-May-2016]. 2016. URL: http://daqri.com/home/product/daqri-smart-helmet/.

Drost, B. and Ilic, S. “Local Hough Transform for 3D Primitive Detection”. In: *2015 International Conference on 3D Vision*. 2015, pp. 398–406. DOI: 10.1109/3DV.2015.52.

Elloumi, W., Guissous, K., Chetouani, A., Canals, R., Leconge, R., Emile, B., and Treuillet, S. “Indoor navigation assistance with a Smartphone camera based on vanishing points”. In: *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*. 2013, pp. 1–9. DOI: 10.1109/IPIN.2013.6817911.

Engelke, T., Keil, J., Rojtberg, P., Wientapper, F., Webel, S., and Bockholt, U. “Content first - A concept for industrial augmented reality maintenance applications using mobile devices”. In: *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*. 2013, pp. 251–252. DOI: 10.1109/ISMAR.2013.6671790.

Figueiredo, L. S., Pinheiro, M., Neto, E. V., Chaves, T., and Teichrieb, V. “Human-Computer Interaction – INTERACT 2015: 15th IFIP TC 13 International Conference, Bamberg, Germany, September 14-18, 2015, Proceedings, Part II”. In: ed. by J. Abascal, S. Barbosa, M. Fetter, T. Gross, P. Palanque, and M. Winckler. Cham: Springer International Publishing, 2015. Chap. Sci-Fi Gestures Catalog, pp. 395–411. ISBN: 978-3-319-22668-2. DOI: 10.1007/978-3-319-22668-2_30.

Fischler, M. A. and Bolles, R. C. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: http://doi.acm.org/10.1145/358669.358692.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., and Marín-Jiménez, M. “Automatic generation and detection of highly reliable fiducial markers under occlusion”. In: *Pattern Recognition* 47.6 (2014), pp. 2280 –2292. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2014.01.005. URL: http://www.sciencedirect.com/science/article/pii/S0031320314000235.

Gherghina, A., Olteanu, A., and Tapus, N. “A marker-based augmented reality system for mobile devices”. In: *Roedunet International Conference (RoEduNet), 2013 11th*. 2013, pp. 1–6. DOI: 10.1109/RoEduNet.2013.6511731.

Given, L. M. *The Sage encyclopedia of qualitative research methods*. Sage Publications, 2008.

Google Inc. *ARCore - Google Developer | ARCore | Google Developers*. [Online; last access: 29-Dec-2017]. 2017. URL: https://developers.google.com/ar/.

— *Tango | Google Developers*. [Online; last access: 29-Dec-2017]. 2014. URL: https://developers.google.com/tango/.

Gozick, B., Subbu, K., Dantu, R., and Maeshiro, T. “Magnetic Maps for Indoor Navigation”. In: *Instrumentation and Measurement, IEEE Transactions on* 60.12 (2011), pp. 3883–3891. ISSN: 0018-9456. DOI: 10.1109/TIM.2011.2147690.

Grubert, J., Langlotz, T., and Grasset, R. “Augmented reality browser survey”. In: *Institute for Computer Graphics and Vision, University of Technology Graz, Technical Report* 1101 (2011).

Ha, J., Cho, K., Rojas, F., and Yang, H. “Real-time scalable recognition and tracking based on the server-client model for mobile Augmented Reality”. In: *VR Innovation (ISVRI), 2011 IEEE International Symposium on*. 2011, pp. 267–272. DOI: 10.1109/ISVRI.2011.5759649.

Hacker Noon. *How Much Time Do People Spend on Their Mobile Phones in 2017?* [Online; last access: 21-May-2018]. 2017. URL: https://hackernoon.com/how-much-time-do-people-spend-on-their-mobile-phones-in-2017-e5f90a0b10a6.

Hadid, A., Heikkila, J. Y., Silven, O., and Pietikainen, M. “Face and Eye Detection for Person Authentication in Mobile Phones”. In: *2007 First ACM/IEEE International Conference on Distributed Smart Cameras*. 2007, pp. 101–108. DOI: 10.1109/ICDSC.2007.4357512.

Hagbi, N., Bergig, O., El-Sana, J., and Billinghurst, M. “Shape Recognition and Pose Estimation for Mobile Augmented Reality”. In: *Visualization and Computer Graphics, IEEE Transactions on* 17.10 (2011), pp. 1369–1379. ISSN: 1077-2626. DOI: 10.1109/TVCG.2010.241.

Halpern, M., Zhu, Y., and Reddi, V. J. “Mobile CPU’s rise to power: Quantifying the impact of generational mobile CPU design trends on performance, energy, and user satisfaction”. In: *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2016, pp. 64–76. DOI: 10.1109/HPCA.2016.7446054.

Hartley, R. and Zisserman, A. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

Herling, J. and Broll, W. “Random Model Variation for Universal Feature Tracking”. In: *Proceedings of the 18th ACM Symposium on Virtual Reality Software and Technology*. VRST ’12. Toronto, Ontario, Canada: ACM, 2012, pp. 169–176. ISBN: 978-1-4503-1469-5.

Hettiarachchi, A. and Wigdor, D. “Annexing Reality: Enabling Opportunistic Use of Everyday Objects As Tangible Proxies in Augmented Reality”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. San Jose, California, USA: ACM, 2016, pp. 1957–1967. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858134. URL: http://doi.acm.org/10.1145/2858036.2858134.

Hodaň, T., Matas, J., and Obdržálek, Š. “On evaluation of 6D object pose estimation”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 606–619.

Holz, D., Holzer, S., Rusu, R. B., and Behnke, S. “Real-Time Plane Segmentation Using RGB-D Cameras”. In: *RoboCup 2011: Robot Soccer World Cup XV*. Ed. by T. Röfer, N. M. Mayer, J. Savage, and U. Saranlı. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 306–317. ISBN: 978-3-642-32060-6. DOI: 10.1007/978-3-642-32060-6_26. URL: https://doi.org/10.1007/978-3-642-32060-6_26.

Hoque, M. A., Siekkinen, M., Khan, K. N., Xiao, Y., and Tarkoma, S. “Modeling, Profiling, and Debugging the Energy Consumption of Mobile Devices”. In: *ACM Comput. Surv.* 48.3 (Dec. 2015), 39:1–39:40. ISSN: 0360-0300. DOI: 10.1145/2840723. URL: http://doi.acm.org/10.1145/2840723.

Hu, C.-L., Cho, C.-A., Lin, C.-J., and Fan, C.-W. “A location tracking and messaging system for mobile group communication in IP networks”. In: *Consumer Electronics (ICCE), 2010 Digest of Technical Papers International Conference on*. 2010, pp. 155–156. DOI: 10.1109/ICCE.2010.5418911.

Hu, W., Lian, S., Song, X., and Li, T. “Mobile camera based cross-screen interaction by object matching and tracking”. In: *Consumer Electronics, IEEE Transactions on* 59.3 (2013), pp. 452–459. ISSN: 0098-3063. DOI: 10.1109/TCE.2013.6626224.

Huang, J. and You, S. “Detecting Objects in Scene Point Cloud: A Combinational Approach”. In: *2013 International Conference on 3D Vision - 3DV 2013*. 2013, pp. 175–182. DOI: 10.1109/3DV.2013.31.

Huang, Z., Hui, P., Peylo, C., and Chatzopoulos, D. “Mobile augmented reality survey: A bottom-up approach”. In: *arXiv preprint arXiv:1309.4413* (2013).

Intel Corporation. *Threading Building Blocks*. [Online; last access: 14-May-2016]. 2016. URL: https://www.threadingbuildingblocks.org.

International Symposium on Mixed and Augmented Reality. *ISMAR 2015 Tracking Competition*. [Online; last access: 22-Jun-2016]. 2015. URL: http://ypcex.naist.jp/trakmark/tracking-competition/.

Issartel, P., Gueniat, F., and Ammi, M. “Slicing techniques for handheld augmented reality”. In: *3D User Interfaces (3DUI), 2014 IEEE Symposium on*. 2014, pp. 39–42. DOI: 10.1109/3DUI.2014.6798839.

Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1991. ISBN: 9780471503361. URL: https://books.google.com.br/books?id=CN1QAAAAMAAJ.

Jung, H., Lee, Y.-D., and Jeong, D.-U. “A novel method for energy expenditure using multisensor based activity monitoring”. In: *Computer Sciences and Convergence Information Technology (ICCIT), 2011 6th International Conference on*. 2011, pp. 103–106.

Jurie, F. and Dhome, M. “Real Time Robust Template Matching.” In: *BMVC*. 2002, pp. 1–10.

Kao, W.-W. and Huy, B. Q. “Indoor navigation with smartphone-based visual SLAM and Bluetooth-connected wheel-robot”. In: *Automatic Control Conference (CACS), 2013 CACS International*. 2013, pp. 395–400. DOI: 10.1109/CACS.2013.6734167.

Kazemi, V. and Sullivan, J. “One millisecond face alignment with an ensemble of regression trees”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1867–1874. DOI: 10.1109/CVPR.2014.241.

Keele, S. *Guidelines for performing systematic literature reviews in software engineering*. Tech. rep. Technical report, EBSE Technical Report EBSE-2007-01, 2007.

Khronos Group. *OpenGL - The Industry Standard for High Performance Graphics*. [Online; last access: 22-Jun-2016]. 1997. URL: https://www.opengl.org.

Kim, Y. M., Dolson, J., Sokolsky, M., Koltun, V., and Thrun, S. “Interactive acquisition of residential floor plans”. In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 3055–3062. DOI: 10.1109/ICRA.2012.6224595.

Kitchenham, B., Brereton, P., and Budgen, D. “The educational value of mapping studies of software engineering literature”. In: *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*. Vol. 1. 2010, pp. 589–598. DOI: 10.1145/1806799.1806887.

Klein, G. and Murray, D. “Parallel Tracking and Mapping on a camera phone”. In: *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*. 2009, pp. 83–86. DOI: 10.1109/ISMAR.2009.5336495.

Klein, G. and Murray, D. “Parallel Tracking and Mapping for Small AR Workspaces”. In: *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. ISMAR ’07. Washington,

DC, USA: IEEE Computer Society, 2007, pp. 1–10. ISBN: 978-1-4244-1749-0. DOI: 10.1109/ISMAR.2007.4538852. URL: http://dx.doi.org/10.1109/ISMAR.2007.4538852.

Kuehni, R. G. *Color space and its divisions: color order from antiquity to the present*. John Wiley & Sons, 2003.

Kurz, D. and Benhimane, S. "Gravity-aware handheld Augmented Reality". In: *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*. 2011, pp. 111–120. DOI: 10.1109/ISMAR.2011.6092376.

Kurz, D. and Benhimane, S. "Handheld Augmented Reality involving gravity measurements". In: *Computers & Graphics* 36.7 (2012). Augmented Reality Computer Graphics in China, pp. 866 –883. ISSN: 0097-8493. DOI: 10.1016/j.cag.2012.03.038.

Lambrecht, J., Walzel, H., and Kruger, J. "Robust finger gesture recognition on handheld devices for spatial programming of industrial robots". In: *RO-MAN, 2013 IEEE*. 2013, pp. 99–106. DOI: 10.1109/ROMAN.2013.6628462.

Lane, N., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., and Campbell, A. "A survey of mobile phone sensing". In: *Communications Magazine, IEEE* 48.9 (2010), pp. 140–150. ISSN: 0163-6804. DOI: 10.1109/MCOM.2010.5560598.

Leonard, J. J. and Durrant-Whyte, H. F. "Mobile robot localization by tracking geometric beacons". In: *IEEE Transactions on Robotics and Automation* 7.3 (1991), pp. 376–382. ISSN: 1042-296X. DOI: 10.1109/70.88147.

Lepetit, V., Moreno-Noguer, F., and Fua, P. "EPnP: An Accurate O(N) Solution to the PnP Problem". In: *Int. J. Comput. Vision* 81.2 (Feb. 2009), pp. 155–166. ISSN: 0920-5691. DOI: 10.1007/s11263-008-0152-6. URL: http://dx.doi.org/10.1007/s11263-008-0152-6.

Leshed, G., Velden, T., Rieger, O., Kot, B., and Sengers, P. "In-Car GPS Navigation: Engagement with and Disengagement from the Environment". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: ACM, 2008, pp. 1675–1684. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357316.

Li, D., Salonidis, T., Desai, N. V., and Chuah, M. C. "DeepCham: Collaborative Edge-Mediated Adaptive Deep Learning for Mobile Object Recognition". In: *2016 IEEE/ACM Symposium on Edge Computing (SEC)*. 2016, pp. 64–76. DOI: 10.1109/SEC.2016.38.

Li, M. and Mourikis, A. I. "Vision-aided inertial navigation for resource-constrained systems". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. 2012, pp. 1057–1063. DOI: 10.1109/IROS.2012.6386223.

Li, M., Kim, B. H., and Mourikis, A. "Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. 2013, pp. 4712–4719. DOI: 10.1109/ICRA.2013.6631248.

Li, P., Qin, T., Hu, B., Zhu, F., and Shen, S. "Monocular Visual-Inertial State Estimation for Mobile Augmented Reality". In: *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2017, pp. 11–21. DOI: 10.1109/ISMAR.2017.18.

Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., and Mitra, N. J. "GlobFit: Consistently Fitting Primitives by Discovering Global Relations". In: *ACM Trans. Graph.* 30.4 (July 2011), 52:1–52:12. ISSN: 0730-0301. DOI: 10.1145/2010324.1964947. URL: http://doi.acm.org/10.1145/2010324.1964947.

Lieberknecht, S., Benhimane, S., Meier, P., and Navab, N. "A dataset and evaluation methodology for template-based tracking algorithms". In: *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*. 2009, pp. 145–151. DOI: 10.1109/ISMAR.2009.5336487.

Lima, J. P., Roberto, R., Teixeira, J. M., and Teichrieb, V. “[Poster] Device vs. user perspective rendering in google glass AR applications”. In: *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*. 2014, pp. 279–280. DOI: 10.1109/ISMAR.2014.6948449.

Lima, J. P., Roberto, R., Simões, F., Almeida, M., Figueiredo, L., Teixeira, J. M., and Teichrieb, V. “Markerless tracking system for augmented reality in the automotive industry”. In: *Expert Systems with Applications* 82 (2017), pp. 100 –114. ISSN: 0957-4174. DOI: http://doi.org/10.1016/j.eswa.2017.03.060. URL: http://www.sciencedirect.com/science/article/pii/S0957417417302221.

Lima, J. P., Roberto, R., Teichrieb, V., and Marques, G. “Study about Natural Feature Tracking for Augmented Reality Applications on Mobile Devices”. In: *Virtual and Augmented Reality (SVR), 2015 XVII Symposium on*. 2015, pp. 7–14. DOI: 10.1109/SVR.2015.9.

Lins, C., Arruda, E., Neto, E., Roberto, R., Teichrieb, V., Freitas, D., and Teixeira, J. M. “Animar: Augmenting the Reality of Storyboards and Animations”. In: *Virtual and Augmented Reality (SVR), 2014 XVI Symposium on*. 2014, pp. 106–109. DOI: 10.1109/SVR.2014.40.

Lins, C., Teixeira, J. M., Roberto, R., and Teichrieb, V. “Development of Interactive Applications for Google Glass”. In: *Tendências e Técnicas em Realidade Virtual e Aumentada* 4 (2014), pp. 167 –188. ISSN: 2177-6776.

Liu, H., Darabi, H., Banerjee, P., and Liu, J. “Survey of Wireless Indoor Positioning Techniques and Systems”. In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 37.6 (2007), pp. 1067–1080. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2007.905750.

Liu, Y. J., Zhang, J. B., Hou, J. C., Ren, J. C., and Tang, W. Q. “Cylinder Detection in Large-Scale Point Cloud of Pipeline Plant”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.10 (2013), pp. 1700–1707. ISSN: 1077-2626. DOI: 10.1109/TVCG.2013.74.

Lopez-Escogido, D. and Fraga, L. G. de la. “Automatic extraction of geometric models from 3D point cloud datasets”. In: *2014 11th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*. 2014, pp. 1–5. DOI: 10.1109/ICEEE.2014.6978316.

Lourakis, M. I. A. and Argyros, A. A. “SBA: A Software Package for Generic Sparse Bundle Adjustment”. In: *ACM Trans. Math. Softw.* 36.1 (Mar. 2009), 2:1–2:30. ISSN: 0098-3500. DOI: 10.1145/1486525.1486527. URL: http://doi.acm.org/10.1145/1486525.1486527.

Lowe, D. G. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

Lucas, B. D. and Kanade, T. “An Iterative Image Registration Technique with an Application to Stereo Vision”. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI’81. Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679. URL: http://dl.acm.org/citation.cfm?id=1623264.1623280.

Lv, Z. “Wearable Smartphone: Wearable Hybrid Framework for Hand and Foot Gesture Interaction on Smartphone”. In: *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*. 2013, pp. 436–443. DOI: 10.1109/ICCVW.2013.64.

Ma, Y., Soatto, S., Kosecka, J., and Sastry, S. *An Invitation to 3-D Vision: From Images to Geometric Models*. Interdisciplinary Applied Mathematics. Springer New York, 2005. ISBN: 9780387008936. URL: https://books.google.com.br/books?id=6tUqQmwan4UC.

Martin, P., Marchand, E., Houlier, P., and Marchal, I. “Decoupled mapping and localization for Augmented Reality on a mobile phone”. In: *Virtual Reality (VR), 2014 iEEE*. 2014, pp. 97–98. DOI: 10.1109/VR.2014.6802069.

Michael, K. and Clarke, R. “Location and tracking of mobile devices: Überveillance stalks the streets”. In: *Computer Law & Security Review* 29.3 (2013), pp. 216 –228. ISSN: 0267-3649. DOI: 10.1016/j.clsr.2013.03.004.

Microsoft. *Microsoft HoloLens | The leader in mixed reality technology*. [Online; last access: 29-Dec-2017]. 2016. URL: https://www.microsoft.com/en-us/hololens.

Morency, L.-P. “3D Constrained Local Model for Rigid and Non-rigid Facial Tracking”. In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. CVPR ’12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 2610–2617. ISBN: 978-1-4673-1226-4. URL: http://dl.acm.org/citation.cfm?id=2354409.2354947.

Mota, R. C., Roberto, R. A., and Teichrieb, V. “[POSTER] Authoring Tools in Augmented Reality: An Analysis and Classification of Content Design Tools”. In: *Mixed and Augmented Reality (ISMAR), 2015 IEEE International Symposium on*. 2015, pp. 164–167. DOI: 10.1109/ISMAR.2015.47.

Mota, R., Marques, G., Santos, A., Araujo, J., Roberto, R., and Teichrieb, V. “BlockIT: Tangible Interfaces for Music Education”. In: *Workshop em Realidade Virtual e Aumentada (WRVA)*. 2014.

Mur-Artal, R., Montiel, J. M. M., and Tard?s, J. D. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. ISSN: 1552-3098. DOI: 10.1109/TRO.2015.2463671.

Nguyen, G., Andersen, H., and Hoilund, C. “Street navigation using visual information on mobile phones”. In: *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*. 2010, pp. 37–42. DOI: 10.1109/ISDA.2010.5687295.

Nguyen, T., Reitmayr, G., and Schmalstieg, D. “Structural Modeling from Depth Images”. In: *IEEE Transactions on Visualization and Computer Graphics* 21.11 (2015), pp. 1230–1240. ISSN: 1077-2626. DOI: 10.1109/TVCG.2015.2459831.

Normand, J.-M. and Moreau, G. “DoF-based classification of Augmented Reality Applications”. In: *IEEE ISMAR workshop “Classifying the AR presentation space”*. 2012, pp. 1–8.

Oehler, B., Stueckler, J., Welle, J., Schulz, D., and Behnke, S. “Efficient Multi-resolution Plane Segmentation of 3D Point Clouds”. In: *Intelligent Robotics and Applications: 4th International Conference, ICIRA 2011, Aachen, Germany, December 6-8, 2011, Proceedings, Part II*. Ed. by S. Jeschke, H. Liu, and D. Schilberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 145–156. ISBN: 978-3-642-25489-5. DOI: 10.1007/978-3-642-25489-5_15. URL: https://doi.org/10.1007/978-3-642-25489-5_15.

Olivier, N., Uchiyama, H., Mishima, M., Thomas, D., Taniguchi, R.-i., Roberto, R., Lima, J. P., and Teichrieb, V. “Live Structural Modeling using RGB-D SLAM”. Unpublished paper accepted at the 2018 IEEE International Conference on Robotics and Automation (ICRA). 2018.

Olsson, T. and Salo, M. “Online user survey on current mobile augmented reality applications”. In: *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*. 2011, pp. 75–84. DOI: 10.1109/ISMAR.2011.6092372.

OpenSignal, Inc. *The State of LTE February 2016 - OpenSignal*. [Online; last access: 22-Jun-2016]. 2016. URL: http://opensignal.com/reports/2016/02/state-of-lte-q4-2015/.

Oui, W., Ng, E., and Khan, R. “An Augmented Reality’s framework for mobile”. In: *Information Technology and Multimedia (ICIM), 2011 International Conference on*. 2011, pp. 1–4. DOI: 10.1109/ICIMU.2011.6122762.

Pang, G., Qiu, R., Huang, J., You, S., and Neumann, U. “Automatic 3D industrial point cloud modeling and recognition”. In: *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*. 2015, pp. 22–25. DOI: 10.1109/MVA.2015.7153124.

Pang, G. and Neumann, U. “Training-Based Object Recognition in Cluttered 3D Point Clouds”. In: *2013 International Conference on 3D Vision - 3DV 2013*. 2013, pp. 87–94. DOI: 10.1109/3DV.2013.20.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. “Systematic Mapping Studies in Software Engineering”. In: *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*. EASE’ 08. Italy: British Computer Society, 2008, pp. 68–77.

Petit, A., Caron, G., Uchiyama, H., and Marchand, E. “Evaluation of Model based Tracking with TrakMark Dataset”. In: *2nd Int. Workshop on AR/MR Registration, Tracking and Benchmarking*. Basel, Switzerland, Switzerland, 2011. URL: https://hal.inria.fr/hal-00639700.

Pew Research Center. *Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies | Pew Research Center*. Ed. by Pew Research Center. [Online; last access: 15-May-2018]. 2016. URL: http://www.pewglobal.org/2016/02/22/smartphone-ownership-and-internet-usage-continues-to-climb-in-emerging-economies/.

Piao, J.-C. and Kim, S.-D. “Adaptive Monocular Visual–Inertial SLAM for Real-Time Augmented Reality Applications in Mobile Devices”. In: *Sensors* 17.11 (2017). ISSN: 1424-8220. DOI: 10.3390/s17112567. URL: http://www.mdpi.com/1424-8220/17/11/2567.

Pirchheim, C., Schmalstieg, D., and Reitmayr, G. “Handling pure camera rotation in keyframe-based SLAM”. In: *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*. 2013, pp. 229–238. DOI: 10.1109/ISMAR.2013.6671783.

Pirchheim, C. and Reitmayr, G. “Homography-based planar mapping and tracking for mobile phones”. In: *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*. 2011, pp. 27–36. DOI: 10.1109/ISMAR.2011.6092367.

Polo, A., Viani, F., Giarola, E., Oliveri, G., Rocca, P., and Massa, A. “Semantic wireless localization enabling advanced services in museums”. In: *Antennas and Propagation (EuCAP), 2014 8th European Conference on*. 2014, pp. 443–446. DOI: 10.1109/EuCAP.2014.6901787.

Qiu, R., Zhou, Q.-Y., and Neumann, U. “Pipe-Run Extraction and Reconstruction from Point Clouds”. In: *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*. Ed. by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars. Cham: Springer International Publishing, 2014, pp. 17–30. ISBN: 978-3-319-10578-9. DOI: 10.1007/978-3-319-10578-9_2. URL: https://doi.org/10.1007/978-3-319-10578-9_2.

Qualcomm Technologies, Inc. *Mobile Multimedia Optimization - Mobile Technologies - Qualcomm Developer Network*. [Online; last access: 03-February-2016]. 2015. URL: https://developer.qualcomm.com/software/hexagon-dsp-sdk.

Rai, H., Deepak, K., Syed, S., and Krishna, P. “A Smart Mobile Application for Identifying Storage Location of Small Industrial Assets”. In: *Mobile Data Management (MDM), 2012 IEEE 13th International Conference on*. 2012, pp. 332–335. DOI: 10.1109/MDM.2012.71.

Raj, C., Tolety, S., and Immaculate, C. “QR code based navigation system for closed building using smart phones”. In: *Automation, Computing, Communication, Control and Compressed Sensing (iMac4s),*

2013 International Multi-Conference on. 2013, pp. 641–644. DOI: 10.1109/iMac4s.2013.6526488.

Ramadasan, D., Chateau, T., and Chevaldonné, M. "DCSLAM: A dynamically constrained real-time slam". In: *2015 IEEE International Conference on Image Processing (ICIP)*. 2015, pp. 1130–1134. DOI: 10.1109/ICIP.2015.7350976.

Ramanan, D. "Face Detection, Pose Estimation, and Landmark Localization in the Wild". In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. CVPR '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 2879–2886. ISBN: 978-1-4673-1226-4. URL: http://dl.acm.org/citation.cfm?id=2354409.2355119.

Rao, J., Qiao, Y., Ren, F., Wang, J., and Du, Q. "A Mobile Outdoor Augmented Reality Method Combining Deep Learning Object Detection and Spatial Relationships for Geovisualization". In: *Sensors*. 2017.

Reiner, H. "The Paramountcy of Reconfigurable Computing". In: *Energy-Efficient Distributed Computing Systems*. Wiley-Blackwell, 2012. Chap. 18, pp. 465–547. ISBN: 9781118342015. DOI: 10.1002/9781118342015.ch18. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118342015.ch18. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118342015.ch18.

Ren, S., Cao, X., Wei, Y., and Sun, J. "Face Alignment at 3000 FPS via Regressing Local Binary Features". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1685–1692. DOI: 10.1109/CVPR.2014.218.

Roberto, R., Freitas, D., Simoes, F., and Teichrieb, V. "A Dynamic Blocks Platform Based on Projective Augmented Reality and Tangible Interfaces for Educational Activities". In: *Journal on Interactive Systems, SBC* 4.2 (2013), pp. 8–18. ISSN: 2236-3297.

Roberto, R., Lima, J. P., Araújo, T., and Teichrieb, V. "Evaluation of Motion Tracking and Depth Sensing Accuracy of the Tango Tablet". In: *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*. 2016, pp. 231–234. DOI: 10.1109/ISMAR-Adjunct.2016.0082.

Roberto, R., Lima, J. P., Uchiyama, H., Arth, C., Teichrieb, V., Taniguchi, R. i., and Schmalstieg, D. "Incremental Structural Modeling Based on Geometric and Statistical Analyses". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 955–963. DOI: 10.1109/WACV.2018.00110.

Roberto, R., Lima, J. P., Mota, R., and Teichrieb, V. "Authoring Tools for Augmented Reality: An Analysis and Classification of Content Design Tools". In: *Design, User Experience, and Usability: Interactive Experience Design: 5th International Conference, DUXU 2016, Held as Part of HCI International 2016*. Ed. by A. Marcus. Springer International Publishing, 2016.

Roberto, R., Lima, J. P., and Teichrieb, V. "Tracking for mobile devices: A systematic mapping study". In: *Computers & Graphics* 56 (2016), pp. 20 –30. ISSN: 0097-8493. DOI: http://dx.doi.org/10.1016/j.cag.2016.02.002. URL: http://www.sciencedirect.com/science/article/pii/S0097849316300115.

Roberto, R., Silva, M. da, Freitas, D., Lima, Y., Silva, V., Araujo, C., Teixeira, J. M., and Teichrieb, V. "Voxar Puzzle Motion: An Innovative AR Application Proposed Using Design Techniques". In: *K-12 Embodied Learning through Virtual & Augmented Reality (KELVAR), IEEE Virtual Reality 2016 Workshop on*. 2016.

Roberto, R. A., Uchiyama, H., Lima, J. P.S. M., Nagahara, H., Taniguchi, R.-i., and Teichrieb, V. "Incremental structural modeling on sparse visual SLAM". In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (2017), p. 5. ISSN: 1882-6695. DOI: 10.1186/s41074-017-0018-3. URL: https://doi.org/10.1186/s41074-017-0018-3.

Robinson, S., Jones, M., Vartiainen, E., and Marsden, G. “PicoTales: Collaborative Authoring of Animated Stories Using Handheld Projectors”. In: *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*. CSCW ’12. Seattle, Washington, USA: ACM, 2012, pp. 671–680. ISBN: 978-1-4503-1086-4. DOI: 10.1145/2145204.2145306.

Roth, G. and Levine, M. D. “Extracting Geometric Primitives”. In: *CVGIP: Image Underst.* 58.1 (July 1993), pp. 1–22. ISSN: 1049-9660. DOI: 10.1006/ciun.1993.1028. URL: http://dx.doi.org/10.1006/ciun.1993.1028.

Roy, A., Zhang, X., Wolleb, N., Quintero, C. P., and Jägersand, M. “Tracking benchmark and evaluation for manipulation tasks”. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. 2015, pp. 2448–2453. DOI: 10.1109/ICRA.2015.7139526.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

Runceanu, L., Becker, S., Haala, N., and Fritsch, D. “Indoor Point Cloud Segmentation for Automatic Object Interpretation”. In: *Publikationen der Deutschen Gesellschaft für Photogrammetrie* (2017), pp. 147–159. URL: https://www.semanticscholar.org/paper/Indoor-Point-Cloud-Segmentation-for-Automatic-Runceanu-Becker/181fd778e36b935479a3eb33abe2d68dbc876f12.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. “Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation”. In: *arXiv preprint arXiv:1801.04381* (2018).

Sankar, A. and Seitz, S. M. “Interactive Room Capture on 3D-Aware Mobile Devices”. In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. UIST ’17. Québec City, QC, Canada: ACM, 2017, pp. 415–426. ISBN: 978-1-4503-4981-9. DOI: 10.1145/3126594.3126629. URL: http://doi.acm.org/10.1145/3126594.3126629.

Santos, J., Rodrigues, F., and Oliveira, L. “A Web & Mobile City Maintenance Reporting Solution”. In: *Procedia Technology* 9.0 (2013). CENTERIS 2013 - Conference on ENTERprise Information Systems / ProjMAN 2013 - International Conference on Project MANagement/ HCIST 2013 - International Conference on Health and Social Care Information Systems and Technologies, pp. 226 –235. ISSN: 2212-0173. DOI: 10.1016/j.protcy.2013.12.025.

Schall, G., Mulloni, A., and Reitmayr, G. “North-centred orientation tracking on mobile phones”. In: *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*. 2010, pp. 267–268. DOI: 10.1109/ISMAR.2010.5643600.

Schnabel, R., Wahl, R., and Klein, R. “Efficient RANSAC for Point-Cloud Shape Detection”. In: *Computer Graphics Forum* 26.2 (2007), pp. 214–226. ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2007.01016.x. URL: http://dx.doi.org/10.1111/j.1467-8659.2007.01016.x.

Schöps, T., Engel, J., and Cremers, D. “Semi-dense visual odometry for AR on a smartphone”. In: *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*. 2014, pp. 145–150. DOI: 10.1109/ISMAR.2014.6948420.

Schöps, T., Sattler, T., Häne, C., and Pollefeys, M. “Large-scale outdoor 3D reconstruction on a mobile device”. In: *Computer Vision and Image Understanding* 157 (2017). Large-Scale 3D Modeling of Urban Indoor or Outdoor Scenes from Images and Range Scans, pp. 151 –166. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2016.09.007. URL: http://www.sciencedirect.com/science/article/pii/S1077314216301412.

Shanklin, T. A., Loulier, B, and Matson, E. T. “Embedded sensors for indoor positioning”. In: *Sensors Applications Symposium (SAS), 2011 IEEE*. 2011, pp. 149–154. DOI: 10.1109/SAS.2011.5739822.

Sheskin, D. *Handbook of Parametric and Nonparametric Statistical Procedures, Fifth Edition*. Taylor & Francis, 2011. ISBN: 9781439858011. URL: https://books.google.com.br/books?id=YDd2cgAACAAJ.

Shibata, F., Ikeda, S., Kurata, T., and Uchiyama, H. “An intermediate report of TrakMark WG international voluntary activities on establishing benchmark test schemes for AR/MR geometric registration and tracking methods”. In: *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*. 2010, pp. 298–302. DOI: 10.1109/ISMAR.2010.5643613.

Shin, B.-J., Lee, K.-W., Choi, S.-H., Kim, J.-Y., Lee, W. J., and Kim, H. S. “Indoor WiFi positioning system for Android-based smartphone”. In: *Information and Communication Technology Convergence (ICTC), 2010 International Conference on*. 2010, pp. 319–320. DOI: 10.1109/ICTC.2010.5674691.

Shin, B., Lee, J. H., Lee, H., Kim, E., Kim, J., Lee, S., Cho, Y. su, Park, S., and Lee, T. “Indoor 3D pedestrian tracking algorithm based on PDR using smarthphone”. In: *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*. 2012, pp. 1442–1445.

Shin, H., Chon, Y., and Cha, H. “Unsupervised Construction of an Indoor Floor Plan Using a Smartphone”. In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 42.6 (2012), pp. 889–898. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2011.2169403.

Silva, M., Roberto, R., and Teichrieb, V. “Evaluation of Augmented Reality Technology in the English Language Field”. In: *Informática na Educação (SBIE), Anais do XXVI Simpósio Brasileiro de*. 2015, pp. 577–586. DOI: 10.5753/cbie.sbie.2015.577.

Silva, M. da, Roberto, R., Teichrieb, V., and Cavalcante, P. “Towards the Development of Guidelines for Educational Evaluation of Augmented Reality Tools”. In: *K-12 Embodied Learning through Virtual & Augmented Reality (KELVAR), IEEE Virtual Reality 2016 Workshop on*. 2016.

Silva, V. E., Lins, C., Silva, A., Roberto, R., Araújo, C., and Teichrieb, V. “Voxar Puzzle: An Innovative Hardware/Software Computer Vision Game for Children Development”. In: *Virtual and Augmented Reality (SVR), 2015 XVII Symposium on*. 2015, pp. 147–153. DOI: 10.1109/SVR.2015.29.

Sim, R. and Roy, N. “Global A-Optimal Robot Exploration in SLAM”. In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. 2005, pp. 661–666. DOI: 10.1109/ROBOT.2005.1570193.

Simões, F. P. M. “Object Detection and Pose Estimation from Natural Features for Augmented Reality in Complex Scenes”. PhD thesis. Federal University of Pernambuco, 2016.

Sinha, S. N., Steedly, D., Szeliski, R., Agrawala, M., and Pollefeys, M. “Interactive 3D Architectural Modeling from Unordered Photo Collections”. In: *ACM Trans. Graph.* 27.5 (Dec. 2008), 159:1–159:10. ISSN: 0730-0301. DOI: 10.1145/1409060.1409112. URL: http://doi.acm.org/10.1145/1409060.1409112.

Solin, A., Cortes, S., Rahtu, E., and Kannala, J. “PIVO: Probabilistic Inertial-Visual Odometry for Occlusion-Robust Navigation”. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 616–625. DOI: 10.1109/WACV.2018.00073.

Song, H., Liu, H., and Chen, D. “An automatic GUI adjustment method for mobile computing”. In: *Computer Science and Automation Engineering (CSAE), 2011 IEEE International Conference on*. Vol. 3. 2011, pp. 206–210. DOI: 10.1109/CSAE.2011.5952665.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. “A benchmark for the evaluation of RGB-D SLAM systems”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 573–580. DOI: 10.1109/IROS.2012.6385773.

Takacs, G., Chandrasekhar, V., Tsai, S., Chen, D., Grzeszczuk, R., and Girod, B. “Unified Real-Time Tracking and Recognition with Rotation-Invariant Fast Features”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 2010, pp. 934–941. DOI: 10.1109/CVPR.2010.5540116.

Tamura, H. and Kato, H. “Proposal of international voluntary activities on establishing benchmark test schemes for AR/MR geometric registration and tracking methods”. In: *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*. 2009, pp. 233–236. DOI: 10.1109/ISMAR.2009.5336441.

Tan, D. and Ilic, S. “Multi-forest Tracker: A Chameleon in Tracking”. In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. 2014, pp. 1202–1209. DOI: 10.1109/CVPR.2014.157.

Tanskanen, P., Kolev, K., Meier, L., Camposeco, F., Saurer, O., and Pollefeys, M. “Live Metric 3D Reconstruction on Mobile Phones”. In: *Computer Vision (ICCV), 2013 IEEE International Conference on*. 2013, pp. 65–72. DOI: 10.1109/ICCV.2013.15.

Tateno, K., Tombari, F., Laina, I., and Navab, N. “CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6565–6574. DOI: 10.1109/CVPR.2017.695.

Teixeira, J. M.X. N. “Analysis and Evaluation of Optimization Techniques for Tracking in Augmented Reality Applications”. PhD thesis. Federal University of Pernambuco, 2013.

Teraura, N. and Sakurai, K. “Preventing the Access of Fraudulent WEB Sites by Using a Special Two-Dimensional Code”. In: *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*. 2012, pp. 645–650. DOI: 10.1109/IMIS.2012.57.

The Next Generation Mobile Networks Ltd. *NGMN - 5G Work Overview*. [Online; last access: 03-February-2016]. 2015. URL: http://www.ngmn.org/work-programme/5g-work-overview.html.

Tobías, L., Ducournau, A., Rousseau, F., Mercier, G., and Fablet, R. “Convolutional Neural Networks for object recognition on mobile devices: A case study”. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. 2016, pp. 3530–3535. DOI: 10.1109/ICPR.2016.7900181.

Tomasi, C. and Manduchi, R. “Bilateral filtering for gray and color images”. In: *Computer Vision. Sixth International Conference on*. 1998, pp. 839–846. DOI: 10.1109/ICCV.1998.710815.

Uchiyama, H., Taketomi, T., Ikeda, S., and M. Lima, J. P. S. d. “[POSTER] Abecedary Tracking and Mapping: A Toolkit for Tracking Competitions”. In: *ISMAR*. 2015, pp. 198–199. DOI: 10.1109/ISMAR.2015.63.

Ullah, A., Islam, M., Aktar, S., and Hossain, S. “Remote-touch: Augmented reality based marker tracking for smart home control”. In: *Computer and Information Technology (ICCIT), 2012 15th International Conference on*. 2012, pp. 473–477. DOI: 10.1109/ICCITechn.2012.6509774.

Valoen, L. and I Shoesmith, M. “The Effect of PHEV and HEV Duty Cycles on Battery and Battery Pack Performance”. In: *Plugin Highway 2007 Conference*. 2007, pp. 1–9. URL: http://umanitoba.ca/outreach/conferences/phev2007/PHEV2007/proceedings/PluginHwy_PHEV2007_PaperReviewed_Valoen.pdf.

Ventura, J., Arth, C., Reitmayr, G., and Schmalstieg, D. “Global Localization from Monocular SLAM on a Mobile Phone”. In: *Visualization and Computer Graphics, IEEE Transactions on* 20.4 (2014), pp. 531–539. ISSN: 1077-2626. DOI: 10.1109/TVCG.2014.27.

Ventura, J. and Hollerer, T. “Wide-area scene mapping for mobile visual tracking”. In: *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*. 2012, pp. 3–12. DOI: 10.1109/ISMAR.2012.6402531.

Vineet, V., Miksik, O., Lidegaard, M., Nießner, M., Golodetz, S., Prisacariu, V. A., Kähler, O., Murray, D. W., Izadi, S., Pérez, P., and Torr, P. H. S. “Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 75–82. DOI: 10.1109/ICRA.2015.7138983.

Wagner, D., Schmalstieg, D., and Bischof, H. “Multiple target detection and tracking with guaranteed framerates on mobile phones”. In: *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*. 2009, pp. 57–64. DOI: 10.1109/ISMAR.2009.5336497.

Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., and Schmalstieg, D. “Real-Time Detection and Tracking for Augmented Reality on Mobile Phones”. In: *Visualization and Computer Graphics, IEEE Transactions on* 16.3 (2010), pp. 355–368. ISSN: 1077-2626. DOI: 10.1109/TVCG.2009.99.

Wagner, D., Mulloni, A., Langlotz, T., and Schmalstieg, D. “Real-time panoramic mapping and tracking on mobile phones”. In: *Virtual Reality Conference (VR), 2010 IEEE*. 2010, pp. 211–218. DOI: 10.1109/VR.2010.5444786.

Wieringa, R., Maiden, N., Mead, N., and Rolland, C. “Requirements Engineering Paper Classification and Evaluation Criteria: A Proposal and a Discussion”. In: *Requir. Eng.* 11.1 (Dec. 2005), pp. 102–107. ISSN: 0947-3602. DOI: 10.1007/s00766-005-0021-6. URL: http://dx.doi.org/10.1007/s00766-005-0021-6.

Xiao, J., Russell, B. C., and Torralba, A. “Localizing 3D Cuboids in Single-view Images”. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 746–754. URL: http://dl.acm.org/citation.cfm?id=2999134.2999218.

Xu, L., Li, S., Bian, K., Zhao, T., and Yan, W. “Sober-Drive: A smartphone-assisted drowsy driving detection system”. In: *Computing, Networking and Communications (ICNC), 2014 International Conference on*. 2014, pp. 398–402. DOI: 10.1109/ICCNC.2014.6785367.

Yovcheva, Z, Buhalis, D., and Gatzidis, C. “Smartphone Augmented Reality Applications for Tourism”. In: *e-Review of Tourism Research (eRTR)* 10.2 (2012), pp. 63–66. URL: http://eprints.bournemouth.ac.uk/20219/.

Zhang, L., Liu, J., Jiang, H., and Guan, Y. “SensTrack: Energy-Efficient Location Tracking With Smartphone Sensors”. In: *Sensors Journal, IEEE* 13.10 (2013), pp. 3775–3784. ISSN: 1530-437X. DOI: 10.1109/JSEN.2013.2274074.

Zhou, F., Duh, H. B.-L., and Billinghurst, M. “Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR”. In: *Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on*. 2008, pp. 193–202. DOI: 10.1109/ISMAR.2008.4637362.

Zhu, L., Hyyppä, J., Kukko, A., Kaartinen, H., and Chen, R. “Photorealistic Building Reconstruction from Mobile Laser Scanning Data”. In: *Remote Sensing* 3.7 (2011), pp. 1406–1426. ISSN: 2072-4292. DOI: 10.3390/rs3071406. URL: http://www.mdpi.com/2072-4292/3/7/1406.