UNIVERSIDADE D
COIMBRA

João Pedro Ferreira Simões

Augmented reality applied to the industry

Dissertation supervised by Professor Dr. Paulo Jorge Carvalho Menezes and submitted to the Electrical and Computer Engineering Department of the Faculty of Science and Technology of the University of Coimbra, for the Degree of Master in Electrical and Computer Engineering, branch of Computers.

October 2021

UNIVERSIDADE D
COIMBRA

João Pedro Ferreira Simões

Augmented Reality Applied to the Industry

Thesis submitted to the
University of Coimbra for the degree of
Master in Electrical and Computer Engineering

Supervisor:
Prof. Paulo Jorge Carvalho Menezes, PhD

Jury:
Prof. Jorge Manuel Moreira de Campos Pereira Batista, PhD
Prof. Paulo José Monteiro Peixoto, PhD

Coimbra, 2021

This work was developped in collaboration with:

University of Coimbra

Department of Electrical and Computer Engineering

Institute of Systems and Robotics

Esta cópia da tese é fornecida na condição de que quem a consulta reconhece que os direitos de autor são da pertença do autor da tese e que nenhuma citação ou informação obtida a partir dela pode ser publicada sem a referência apropriada.

This thesis copy has been provided on the condition that anyone who consults it understands and recognizes that its copyright belongs to its author and that no reference from the thesis or information derived from it may be published without proper acknowledgement.

Agradecimentos

Quero desde já agradecer ao meu orientador Professor Doutor Paulo Jorge Carvalho Menezes pelo tempo, dedicação, suporte e conhecimentos partilhados ao longo deste trabalho. Sem a sua ajuda este trabalho não teria sido possível. Além do trabalho em si, permitiu evoluir muito a minha visão e mentalidade quanto ao método de trabalho, assim como na integração a um grupo onde os conhecimentos devem ser partilhados para que todos evoluam em conjunto. Além disto sei que com a sua orientação, possuo neste momento uma análise científica e escrita mais crítica do que quando entrei pela primeira vez no laboratório.

Laboratório ao qual, pertencendo ao Instituto de Sistemas e Robótica da Universidade de Coimbra, agradeço pelas condições que possibilitaram a realização do trabalho. Não tive qualquer falta de material e condições básicas para a realização do projeto.

A todas as pessoas que estiveram à minha volta, quer presentemente no laboratório, quer "na porta ao lado", e me ajudaram em qualquer situação necessária, orientaram-me com a sua experiência e conhecimentos. Mas também pelos momentos fora trabalho, o espírito sentido no ambiente onde me integrei é algo que acredito ser raro de se encontrar.

Levo na memória todos os amigos e conhecidos que o curso me trouxe, todas as gargalhadas dadas no eterno "bar do Vítor", todas as saídas, momentos de estudo, desporto, camaradagem, bondade, preocupação e sobretudo alegria. À geração "MiEEC" 2015!

Agradeço do fundo do coração a toda a minha família pelo apoio incondicional, por terem deixado coisas por fazer, por terem dado do seu tempo e posses para me verem agora nesta etapa. Principalmente ao meu pai e mãe por sempre me dizerem que sou eu quem decide o meu futuro e que eles estarão sempre lá para me apoiar a ser feliz e realizado. Sei que também eles o são ao me ver em tal estado. Aos meus avós que sempre me disseram ser um orgulho e sei que chorarão quando receberem a notícia de que esta minha etapa chegou ao fim. Sempre desejei conseguir terminar esta etapa com eles presentes, não tem sentido de outra forma, são eternos.

Agradeço a todos os meus amigos que conheci ao longo dos tempos, alguns dos quais presentes na minha vida toda, outros que só apareceram mais tarde, mas têm uma importância e preocupação por e para mim semelhante.

Agradeço por último à minha namorada que me apoiou em todo o percurso académico, foi e continua a ser uma força insubstituível. É alguém preocupada, amável, que gosta de ver os outros

bem e de construir futuro com uma rede de amigos sempre muito presente. Deu-me muita força, nomeadamente nos momentos em que me senti mais em baixo e sem energia ou paciência. É alguém que espero levar comigo para a vida e conseguir orgulhar cada vez mais.

A todos os mencionados e aos que pela brevidade da situação não consigo mencionar, um grande e sincero obrigado!

Abstract

The recent surge in Augmented Reality (AR) led to its application in numerous areas, particularly industry. This technology is one of the pillars of the transition we are now living towards, the fourth significant industrial change termed ”Industry 4.0”, which requires the maturity of simultaneous mapping the real world and localising the camera, performed dynamically in real-time with higher confidence and robustness than in the past.
Using Smart Glasses with a monocular camera system on the user's forehead, this approach encompasses recording the environment, where freely placed fixed fiducial squared markers of multiple sizes are distributed and used to estimate the camera pose. To do this, the user navigates in the environment, and the markers pose estimation is done with the aid of an Extended Kalman Filter applied to the ”Perspective-N-Points”. This filter provides the pose estimation and uncertainty per marker, allowing a relation between markers with uncertainty associated if at least two markers are detected in the video frame, enabling their mapping. The updated map leads to a faster and more robust camera pose estimation while allowing for scale determination, drifting effect reduction, and providing the algorithm with an absolute axis. Moreover, this method presents a more flexible solution to improving the mapping and localisation of the known areas and expanding to the unknown by adding more markers. Finally, this project allows estimating where the user is looking to provide superimposed information about its work, particularly in an industrial environment of a mould assembly line, possibly improving its workflow while being a time-saver.
Nevertheless, this method can be used for multiple other areas such as other industrial applications, mobile robot localisation applications, video games, and drones.

Keywords: Augmented Reality, Industry 4.0, Fiducial Marker Map, Simultaneous Localization and Mapping

Resumo

A recente explosão da Realidade Aumentada (RA) levou à sua aplicação em diversas áreas, e em particular na indústria. Esta tecnologia é um dos pilares que sustenta a quarta e atual grande revolução industrial denominada ”Indústria 4.0”, que requer maturidade no mapeamento e localização da câmara em simultâneo, que deve ser realizada dinamicamente e em tempo real com maior confiança e robustez do que no passado.

Com auxílio de “Smart Glasses”, que possuem um sistema de câmara monocular na testa do utilizador, esta abordagem abrange a gravação do ambiente onde são distribuídos marcadores fiduciais fixos de vários tamanhos e utilizados para realizar a estimação da pose da câmara. Para tal, o utilizador deve navegar no ambiente onde a estimação da pose dos marcadores é feita com a ajuda de um Filtro de Kalman Estendido aplicado a ”Perspective-N-Points”. Este filtro fornece a estimação da pose e incertezas de cada marcador, permitindo que estes se relacionem entre si com uma incerteza associada quando pelo menos dois marcadores são detetados na imagem atual do vídeo, o que permite o seu mapeamento. Este mapa, uma vez atualizado, consegue suportar uma estimação da pose da câmara mais rápida e mais robusta, enquanto permite a determinação da escala, redução do efeito de “drifting”, e estabelece ao algoritmo com um eixo absoluto. Além disso, este método apresenta uma solução mais flexível na melhora do mapeamento e a localização de áreas conhecidas e expansão para desconhecidas através da introdução de mais marcadores. Finalmente, este projeto permite estimar para onde o utilizador está a olhar de forma a fornecer informação sobre o seu trabalho sobreposta ao mundo real, particularmente num ambiente industrial de uma linha de montagem de moldes, o que permite uma possível melhora no seu fluxo de trabalho enquanto otimiza tempo.

Este método pode ainda ser utilizado para diversas outras áreas, tais como outras aplicações industriais, aplicações de localização de robôs móveis, jogos e “drones”.

Palavras-Chave: Realidade Aumentada, Indústria 4.0, Mapa de Marcadores Fiduciais, Localização e Mapeamento Simultâneos

List of Figures

List of Tables

Contents

1

Introduction

The presented M. Sc. Dissertation was developed sharing multiple objectives with the mobilizing project "Tooling4G" [64] and aims to aid the Augmented Reality penetration in the industry, specifically the mould industry. Augmented Reality is a recent technology that has been attracting many professionals looking to achieve "Industry 4.0", which has massive potential for growth in the following years.

1.1 Motivation

The mould factory environment is divided into various sectors from the moment the design is created, typically in software applications such as "CAD", to its final assembly. According to the desired application of a particular mould, it might be composed of numerous multiple minor parts with different complexities, therefore, varying the complexity of the assembly process accordingly. To assemble such parts into the final mould, a specialised factory worker must understand the multiple files defining the work in hands which can be a heavy cognitive duty. This project proposes the introduction of Augmented Reality to the daily work of assembly lines workers to alleviate this cognitive load by becoming a time-saving assistant and relieving the worker's effort on existing tasks by providing valuable and adaptable superimposed information directly in his visual field. Augmented Reality application is ideal for the specific area of the mould industry because the moulds are not mass produced, leading to a non-optimisable production line. Furthermore, it is highly adaptable and can be implemented dedicated to various functions according to the factory demands.

One aspect to consider is Augmented Reality as an extension to the worker's workspace to lower its effort when searching for information. It must never be over implemented in a way that burdens the user, either by occluding its workspace unnecessarily or blocking its field of view. It must then be highly and dynamically adaptable to fit the user job efficiently, and all the virtual elements must then be user-controllable (scale, position, rotation, toggle visibility, ...).

When used correctly, Augmented Reality can enrich the user with real-world superimposed virtual information such as pieces, tools, task sequences, machinery conditions, step-by-step instructions, models, and visual representations of problems and flaws with instant fixes or workarounds. Furthermore, multiple people can share virtual elements simultaneously, ideal for operators doing

cooperative work and providing potential clients with a visual representation of what they are buying with possible customisations and modifications on the fly. By doing this, all parties can provide feedback, whether physically present or not.

1.2 Objectives and Contributions

Augmented Reality, as previously introduced, can deliver the assembly line worker with easy to visualise information by superimposing it with the real world. To display the before-mentioned information, it is required to know where the user is looking and have a metric system to fuse the correct information accordingly. Multiple sensors can estimate the user head position and orientation, but the most typical is using a camera sensor and then displaying the information with specialised hardware such as Head Mounted Displays (HMD's). In particular Smart Glasses, most of which already have at least a camera on the user's forehead, are ideal as the same device can record the video feed of the environment and display the virtual elements to the worker.

Techniques using the extraction of structural features and their use to localise and map the environment exist, but as the user workspace is prone to constant change or have a limited extractable feature set, they are not reliable in this scenario. With this in mind, this project proposes using a single standard camera (monocular camera system) and the application of its video feed in computer vision techniques to fastly detect and estimate the positions and orientations of squared fiducial markers related to the camera. According to the environmental conditions presented, these markers are printed on a traditional piece of paper and freely placed in the real world. In other words, the positions of the markers are dynamic *a priori* of the mapping process, and the user decides where to place the markers after analysing the environment. As the mapping process evolves, the markers mapped must stay still.By estimating the location of each marker relative to the camera per frame of the video sequence, it is possible to relate the markers if at least two markers are detected simultaneously. This requirement is the most significant limitation of the proposal, as the creation of a relation between two markers encompasses the detection of both simultaneously. Using this knowledge, the first contribution of this work encompasses the use of a monocular camera system and fiducial markers of possibly different sizes to efficiently, cheaply, and with good performance create a technique to do simultaneous location and mapping. The difference in marker sizes is essential to this application as it enables the map of the user workspace with small markers and complex areas such as corridors with bigger ones. As they are known physical objects, by knowing their sizes, the map's scale is also known, and they provide the absolute localisation of the camera in the map. The more markers mapped and currently detected, the more accurate and robust the localisation as the algorithm has more information to work with, allowing the user to do its natural activities without worrying about the occlusion of some of the markers. The second contribution is an improved estimation of the camera by implementing the work proposed in "EKFPnP: Extended Kalman Filter for Camera Pose Estimation in a Sequence of Images" [36], and its input is all the four corners of all the marker corners detected simultaneously. The third, fourth and fifth contributions, using the EKFPnP implementation, provide a per marker improved position and ori-

entation estimation, the application of the per marker uncertainties provided by the Kalman Filter to the mapping process and lastly, taking advantage of the predictive characteristics of the filter, improve the frame rate presented to the user by predicting the camera position in-between frames.

1.3 Manuscript structure

This manuscript is divided in the following chapters:

- **Chapter 1:** Presents an Introduction, motivation and objectives of the proposed work;
- **Chapter 2:** Describes previous works that allow the mapping and localisation of the camera system and evolution towards the requirements previously described in chapter 1;
- **Chapter 3:** Provides basic knowledge required to understand the topics of this work more efficiently;
- **Chapter 4:** Presents the proposed system for mapping and localising the camera in the real world based on fiducial markers, enabling the user's field of view estimation;
- **Chapter 5:** Integrates the work of UcoSLAM as a separate option for the objective of this work, even though it is not ideal in all scenarios;
- **Chapter 6:** A brief discussion of the two methods used, how and where each one is most suited for, as well as possible mutual improvements.

2

Related Work

Augmented Reality has been a hot topic of research recently as it has vast application scenarios such as gaming, rehabilitation, teaching, and architecture. Furthermore, its application in the real world is steadily more and more conceivable for mobile devices as the hardware improves, and the computationally intensive algorithms required to estimate such devices position and orientation become more and more efficient and precise, something essential to display the correct superimposed information in the correct place of the real world. Besides the display of virtual elements, the interaction with virtual elements has also been an intensive area of research. This interaction can have multiple inputs, such as voice recognition [59] or tracking the user's hands [61] and placing virtual GUI like elements at arms reach.

All previous progress leads to the surge of Augmented Reality application in the industry to aid the worker with valuable information more naturally and practically than the traditional GUI, which evolved from the preceding Command Line Interface (CLI), being the most frequent technique used to show information to the user, including the operators in most industrial use cases [48]. Although the GUI has evolved massively from its beginnings to be user friendly, responsive, snappy, and be used on a broad range of devices, including laptops, tablets, and smartphones, to mention a few. It still requires the user to potentially stop, drop equipment or machinery, and reach out to the device's location. The surge of Augmented Reality provides information directly in the user field of view without any of the difficulties described. It is believed to be the next stage in displaying information and a massive step towards Industry 4.0 [30][58].

The information to be presented via Augmented Reality requires a camera-based localisation that enables the estimation of the user head position and orientation in most use cases. Such estimation is essential to know where the user is looking at and display the correct virtual elements. Computer vision [54] is a handy tool that enables the extraction of meaningful information about the images provided, particularly about the environment, objects, or the camera itself, by using Smart Glasses [62][23]. It is a remarkably non-invasive, accurate and low-cost technique tool, particularly when compared with other mechanical, magnetic and ultrasonic trackers. Inertial trackers can also be used, but they suffer from the drifting effect. On the other hand, it requires strenuous effort, research and development but the last decades as shown evolution by giant steps. There are, however, problems such as lighting, fast movements causing motion blur on the image, shadows, sensor limitations (particularly CMOS sensors that suffers a lot from the Rolling effect as shown

in Fig. 3.3) and others. These problems can negatively impact the keypoints extraction and/or robustness, leading to non-unique related descriptors. Variations on the scene may entirely modify the keypoints cloud, which significantly impacts tracking algorithms. Tracking based on Computer Vision requires processing after keypoint extraction, providing information about the scene, such as the depth, objects, and occlusion map. It can also be used to create maps that allow navigation with reliable localisation in real-time. These keypoints can be extracted from structural corners, edges or textures or even from artificially added markers. Thus, the camera pose estimation can be defined as ”Marker/Markerless”, depending on the case.

The extraction of these keypoints has the objective of detecting, describe and matching local features among pairs of images and is done based on image processing algorithms such as SIFT[35], which is a keypoint detector and descriptor proven remarkably successful as it is invariant to scale and rotation. This algorithmm is computationally expensive, therefore not ideal for real-time application. To solve this issue, multiple techniques that do not require dedicated hardware such as GPU's were developed, leading to a new proposal with similar properties but lower complexity designated SURF. ORB keypoints [57] emerged of FAST [55][56] keypoint detector and BRIEF [9] descriptor, as a solution that mitigates their individual limilations by:

- Adding Fast and precise orienting component to FAST;
- Providing fast and efficient BRIEF features computation;
- Correlation and variance analysis of oriented BRIEF features;
- Introducing a modified learning approach for de-correlating BRIEF features while maintaining rotational invariance, improving performance in nearest-neighbour applications;

ORB also enabled scene recognition, tracking, and mapping with features which are quick to extract and match, allowing real-time performance. Furthermore, they are resistant to rotation and scaling, and they have good invariance to camera autogain, autoexposure, and lighting changes and perform well in bag-of-words location identification [46]. These were first applied to a relocalisation method for keyframe-based SLAM, up to our knowledge, that can deal with severe viewpoint change in ”Fast Relocalisation and Loop Closing in Keyframe-Based SLAM” [41].

SLAM [29] is a well-known technique that tries to generate a real-world map using one or more sensors, such as LIDAR, video and depth cameras, odometer, and an inertial sensor, while simultaneously navigating and localising itself in that mapped area. If camera sensors are used, it is known as visual SLAM [63], and in particular, monocular visual SLAM has seen a significant improvement in the performance of ”Parallel Tracking and Mapping” (PTAM) [26]. According to this research, PTAM was the first to use ”Bundle Adjustment” (BA) [38] and the pioneer of parallelization applied to SLAM algorithms as it takes advantage of two threads of a dual-core processor where the first thread does the tracking, and the second is responsible for managing the 3D point cloud based on the previous frames using FAST keypoints using patches. This is not ideal for relocalization (e.g. detect when the sensor returns to a mapped region and possibly fix the accumulated error), leading to large loops not being detected as relocalization is based on the

correlation of low-resolution thumbnails of the keyframes, yielding a low viewpoint invariance. Based on PTAM and ORB groundbreaking works, ORB-SLAM[42], a keyframe-based SLAM algorithm for the monocular camera system, achieved outstanding accuracy and performance for indoor tracking and mapping relocalisation and loop closure using ORB keypoints and a BoW[17] approach, being since a pilar of multiple projects and methodologies. This project was later expanded to include monocular, stereo and RGB-D cameras with the name ORB-SLAM2[40], a very recognized work. However, these natural keypoints approaches still show a number of drawbacks:

- Unknown scale of the produced maps, therefore, worthless for self-navigation;
- Failure in pure rotational motions;
- Minimum degree of texture, which might not be available in multiple interiors,

The majority of the previous projects used structural keypoints in order to estimate the relation between the images. A different approach utilises reliable hand placed fiducial markers to help locate and assure the existence of the keypoints and provide metrics.
Among the first squared fiducial marker proposals is ARToolKit[24], where the authors used a custom pattern that can be identified using template matching. However, it does not handle lightning well and is prone to errors. Furthermore, the method does not handle well the growth of the number of markers. This project was modified multiple times to solve its problems which led to the creation of ARToolKit+[65]. Other projects such as ARTag[15] are also recognizable, but it was also discontinued. The necessity of having a customizable marker code emerged, being BinARyID[16] one of the firsts. AprilTag[47] emerged with the advantages of error detection and correction, but it also did not scale well with large number of markers. According to the authors, ArUco[19] is probably the most popular system, and this research led as to believe in this claim. It was proposed in 2014 as a paper intending to create a versatile fiducial squared markers specially appropriated for camera pose estimation in multiple applications such as augmented reality, robot localization, etc. This paper had three main objectives:

- The creation of marker dictionaries wich generation is configurable in size and number of bits, following a criterion to maximize the inter-marker distance and the number of bits transition[18];
- Propose a method of automatic detection of the markers with automatic error identification and correction, based on Hamming Code[60], inside the same dictionary;
- Solution for the occlusion problem in the augmented reality application,

Following the previous work, the authors proposed ”Speeded Up ArUco”[53]. This proposal aims to solve the time-consuming marker detection process, particularly in high-resolution images, by proposing a multi-scale strategy for speeding the marker detection in video sequences without sacrificing accuracy or robustness. The authors reduce the input image to detect and identify the markers but estimate the four positions of the corners by later upsampling the input using an image pyramid technique.

Combining the knowledge acquired by the previous works, the research group ”Applications of Artificial Vision” (A.V.A) of the University of Cordoba [2], the same group that developed ArUco and Speeded Up ArUco, envisioned a SLAM algorithm that uses Squared Fiducial Markers instead of the typical natural keypoints approach to counter the drawbacks presented previously. The group developed “Marker Map” [44] as a starting point. This offline method enables the map of ArUco markers in large indoor environments after distributing the printed markers on the area to be mapped. The algorithm requires submitting a collection of frames where at least two markers are visible in each frame to enable the pose relation of the markers iteratively until no new frame has valuable information that can be introduced. However, it has several limitations:

- Errors can not be detected until the whole process is finished.
- The method is not incremental, meaning that if map editing is required, it is necessary to repeat the whole process from the start.
- The method can not be employed in real-time systems with limited computational resources.

With the experience gathered, SPM-SLAM [43] was developed to improve the previous work. This keyframe based SLAM algorithm proposed a real-time solution to the problems of simultaneously localising the camera and building a map of ArUco markers. This paper contributes to multiple problems when solving SLAM based on squared planar markers, such as the ambiguity problem (the authors believe this might be why the use of fiducial markers have been avoided so far).
Their latest work, UcoSLAM, has the objective of reducing the problems associated with most SLAM algorithms that use natural keypoints, as well as the ones found in SPM-SLAM.
To do this, the authors did a complete recodification of ORB-SLAM2 in order to improve its performance, including parallelization of the keypoint detection, essential modifications to make it work in real-world applications (notoriously the save and load of maps system), as well as improvements in monocular camera system scenarios.
Besides the modification to ORBSLAM2, the extension and merge with SPD-SLAM made possible the combining of natural and artificial landmarks to map and localize the camera in the real world in real-time by taking the best part of each system to complement each other, improving the overall result where the squared fiducial markers improve the SLAM algorithm particularly on tracking and relocalization.

Immediate results from the inclusion of SPD-SLAM:

- Finding the map’s scale as soon as the first marker is detected;
- The advantage of working in problematic areas for natural keypoints algorithms, such as corridors, given that the algorithm can use natural, artificial or both kinds of landmarks simultaneously;
- The removal of visual ambiguities provide a massive help in the relocalization;
- Reduction in the drifting effect;
- Long term mapping.

On the other hand, the SLAM also improves the SPD-SLAM by:

- Removal of the requirement of having two markers detected simultaneously to relate their poses.
- Removal of the necessity of having a large number of markers sparsely displaced to have a useful mapping of the environment and decent localization.

Even though UcoSLAM is an outstanding work, it still has some limitations. In particular, it does not allow multiple marker sizes as it was built from SPM-SLAM that did not account for it. By turning off the ORBSLAM2 based algorithm in UcoSLAM, the algorithm runs SPM-SLAM standalone, and by turning off SPD-SLAM, the algorithm runs an improved ORB-SLAM2 algorithm standalone.

3

Background

This chapter summarises the information required to understand crucial concepts and provide the reader with the basis to understand the following chapters.

3.1 Augmented Reality

3.1.1 History of Augmented Reality

According to the work of [4][11], augmented and virtual reality emerges following multiple inventions between 1950 and 1990, years were the firsts inventions related to the topic were firstborn. Morton Heilig, a cinematographer, imagined in the 1950s that cinema would be available to all five senses rather than just the two major ones of sight and sound, authoring a paper explaining his vision in 1955 and constructing a device named ”Sensorama” [21] in 1962. Because Morton invented this equipment before the computing era, some believe he was ahead of his time, and many consider him a precursor of today’s Augmented Reality. He even patented ”Stereoscopic-television device for individual use,” which is quite related to today’s Head-Mounted Displays, in 1960[22]. It does not, however, include head-tracking technology.
Ivan Sutherland invented the first fully head-mounted display in 1966. In 1968, with the aid of his pupils, he created the first Augmented Reality system gadget. Finally, this system incorporated 3D head tracking and a virtual environment. The system was given the moniker ”The Sword of Damocles.”
Tom Caudell and David Mizell coined the term ”Augmented Reality” in 1990, and it has since been a hot topic of discussion and study. In 1992, the two writers released ”Augmented Reality: An application of heads-up display technology to manual manufacturing process” [12] in a paper. The concept of a ”heads-up, see-through, head-mounted display” (HUDSET) is used to the construction of auxiliary aircraft in this article. Particularly in the aircraft’s wiring. Nonetheless, Ronald Azuma proposed the notion of Augmented Reality in 1997, describing it as the merging of the actual environment with the overlay of computer-generated information.

3.1.2 What is Augmented Reality?

Augmented reality is an approach that attempts to dynamically and in real-time augment or improve the real world with computer-generated data. This approach has two primary problems, it must

estimate the accurate overlay of the coordinate systems of the real and virtual worlds, as well as, possibly, capture the 3D environment, such as objects, in order for the user to interact with both worlds [27]. Once the coordinate systems are fully aligned, computer-generated data may be shown to the user using specialized augmented reality display devices that allow for a visual merge of the two worlds. This allows the user to further acquire knowledge about the real world by incorporating information directly with the 3D reality or displaying basic information such as assignments on the fly without other devices or pauses.

It differs from Virtual Reality because, while Augmented Reality uses the real world as the base for the computer generated data displayed, Virtual Reality enclosure the user in a complete virtual world where nothing is real, everything is a digital simulation of the real world [3].

3.1.3 How to achieve Augmented Reality?

The pure view of Reality as we know it is the foundation of Augmented Reality. It is founded on cognitive evolution, which entails the acquisition of concepts, mental models, and natural sensory mechanisms. The sense of Reality depends on comparing the information gathered by sensory organs to what was collected beforehand [37].
Because our sensory organs are limited, our cognitive processes assist in creating and developing tools that allow us to verify and accept information acquired from the real world that we have come to acknowledge throughout our lives. As a result of this learning, our inference, reasoning, and recognizing abilities have evolved, allowing us to perceive Reality even when we only have incomplete knowledge. Human eyesight is an example of incomplete information acquisition. It is based on a fronto-parallel sensory configuration of the human eyes, which is crucial to the tridimensional estimate of objects and surroundings based on a 2D projection of the 3D world.
Our brain has been specially evolved to detect flaws and visual discrepancies. With this in mind, Augmented Reality must stimulate the sensory organs through technology that introduces virtual aspects that precisely mimic what we perceive about the real world. The sensory organs must be activated by introducing such virtual elements that are coherent with all of the knowledge we have gathered throughout our lives, matching our mental representations. Otherwise, the user would immediately suspect that something is wrong. As a result, they must respect physical principles, such as pose consistency, rotation, and perspective, and respect real objects in the user's field of view to manage occlusion and, finally, all user motions properly. Other factors, such as lighting, can also be addressed, although they are outside this project's scope.

3.1.4 Augmented Reality Displays

Augmented Reality, according to [5], is the outcome of a set of optical, electrical, and mechanical components that generate an image in the user's field of view, which may be created on planar or non-planar surfaces depending on the optics employed. The Fig. 3.1 illustrates the multiple possibilities of where the image is formed, where the displays are located and the type of produced image.

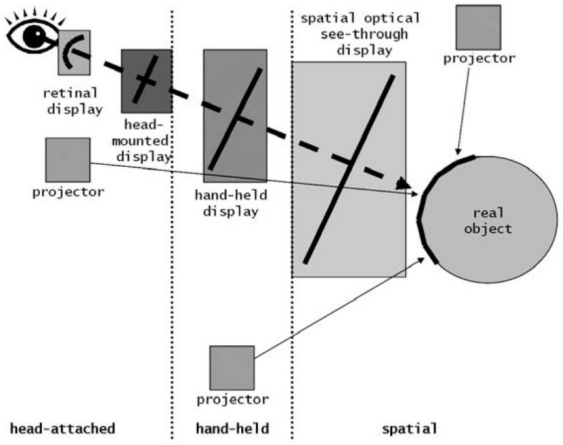

Figure 3.1: Image Generation for Augmented Reality Displays (from [5])

Head Attached Displays

The head attached displays requires the user to wear the display system on his head and are separated into three main types depending on the image generation technology:

- **Retinal Displays:** where low power lases are projected into the retina of the eye;
- **Head Mounted Display:** where the image is presented in a display in front of the user eye and can be divided into two groups, video see-through and optical see-through.

  Smart glasses[50][28], in particular, are wearable HMD's with a high-value Augmented Reality offer comparable to regular glasses, which are expected to have a significant impact on human media consumption. They provide the user with augmented reality by sensing the user's surroundings using computer vision, for example, and superimposing virtual objects in the real world.
  This technology is still in the research stage, but it is expected to be one of the most disruptive technologies in the near future as it becomes less obtrusive to consumers. Previous devices were, among other problems, were large, heavy, and had limited autonomy. However, as technology advances, mass-market Smart Glasses are on the horizon, and we want to be prepared with the software when such hardware becomes available, especially for true optical see-through. Two examples of these devices that show substantial evolution compared with previous devices are Microsoft Hololens[23] and Google Glass[49].
  Policies are also a significant source of concern since their use in public areas might, for example, infringe on other people's privacy [66].

  They can be divided into two main categories:

  – **Video see-through**

    This smart glass category uses a typical closed Head Mounted Display, using its cameras to project both the reality and the virtual elements on the screen to make the user experience Augmented Reality.

  - **Optical see-through**

    This smart glass category uses a transparent plane where virtual elements are projected. The core functionality is similar to the holographic displays. However, they are much more appropriate to the end-user as they can visualize the over-imposed information and still have all the natural movement.

- **Head Mounted Projector:** where the image is projected on the surfaces of the real world,

Hand-Held Displays

In general, this devices are within arms reach and typically combine processor, memory, display, and interaction technology into one single device. They aim at supporting a wireless and unconstrained mobile handling.
Video see-through is the preferred concept for such approaches, here, the integrated video cameras capture live video streams of the environment that are overlaid by graphical augmentations before displaying them.

Spatial Displays

This Approach detach most of the technology from the user and integrate it into the environment.

- **Screen Based Video See-Through Displays:** also known as ”window on the world”, make use of video mixing and display the merged images on a regular monitor;
- **Spatial Optical See-through Displays:** In contrast to head-attached or hand-held optical see-through displays, spatial optical see-through displays generate images that are aligned within the physical environment, giving the user the chance to move around them;
- **Projection-Based Spatial Displays:** Use front-projection to seamlessly project images directly on a physical objects surfaces,

3.2 Camera

As described before, the Augmented Reality application of this project relies on utilising a monocular camera system. The camera’s data must be as precise as possible to create a mixed reality believable and natural for the user to represent the real world correctly, such as the scale of the environment, which implies a camera calibration ”a priori” of the system utilisation. The camera calibration method can estimate two types of camera parameters, intrinsic and extrinsic.

3.2.1 Digital Camera: A quick overview

Most digital cameras nowadays have CCD or CMOS sensors [34]. Both share the same philosophy, acquire light intensity information based on a grid of sensors placed below a colour differentiation filter such as the ”Bayer filter”, and amplify the reading to get a voltage value that can be processed later. The most significant difference between the two is that the photosins in the CCD sensor are

passive while active in the CMOS sensor, this can be verified in the Fig. 3.2 and Fig. 3.3, respectively. It means that once the CCD sensor captures the information, the information is moved, and the amplification is done elsewhere. In the CMOS sensor, however, each photosin has its local amplifier enabling local processing. Having a local amplifier can seem to be preferable at first, but CMOS cameras, particularly the cheaper ones such as those found in most smartphones, tend to provide ”rolling artefacts” when the camera moves faster than the speed the photosins can absorb the information of the whole image, as visualised in the Fig. 3.4. These artefacts can be extremely harmful when applied to Augmented Reality applications because they can create a warping on the image, leading to an entirely useless image that can damage the algorithm. This subject is being discussed as a recommendation to the final user as it is recommended to use CCD-based cameras for Computer Vision applications, including Augmented Reality. CMOS cameras can still be used as long as the user test it a priori. In this project, it was found that particularly older and cheaper web cameras tend to suffer a lot of the rolling effect, while recent smartphones have a fast enough image acquisition for this problem to be mostly bypassed.

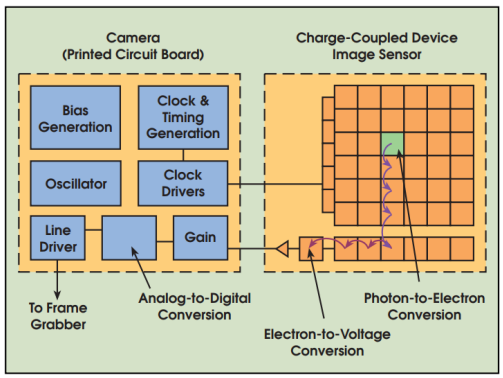

Figure 3.2: CCD (from [34])

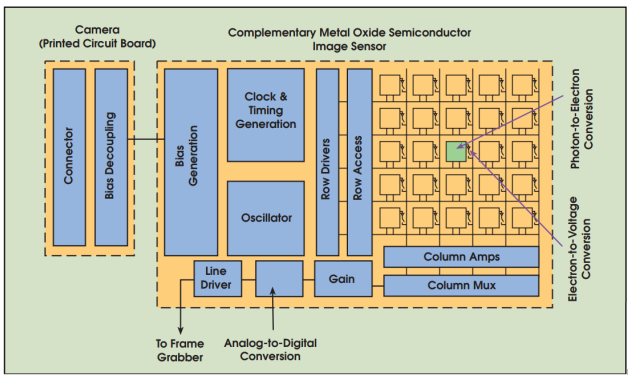

Figure 3.3: CMOS (from [34])

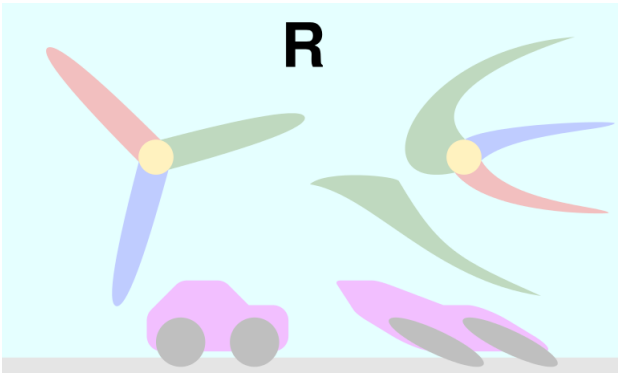

Figure 3.4: RollingEffect (from [52])

In sum, apart from the type of sensor, the ordinary user must generally consider the size of the camera sensor in megapixels, the lens, the post-processing software, and shuttering. Ideally, the higher the megapixels, the better the image quality, offering a better keypoint estimation, particularly for detecting the corners of the markers far from the camera. On the other hand, it also leads

to a slower algorithm, confirming the necessity of the work [53]. Shuttering is the ability to start and stop exposure arbitrarily, therefore, lower shutter speed leads to a reduced frame motion and Rolling Effect.

3.2.2 Pinhole Imaging

When an object is hit by light in the real world, it tends to reflect light in all directions. If we take a light sensor and try to take a photograph on these conditions, each sensor's photosin will accumulate the light stimuli of all the light beams that every object in the Field of View of the photosin provides. The result is the utilization of every scene point in every single pixel. This is known as a "Bare Sensor", as represented in Fig. 3.5. Using a bare sensor, the image is a complete mess being completely blurred and useless. Nothing can be perceived as every pixel has the information of everything on the scene.

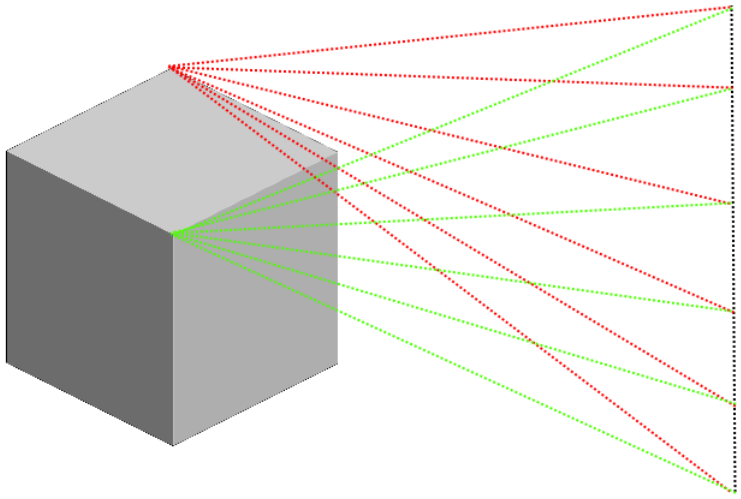

Figure 3.5: Bare Sensor

To get anything useful from the camera sensor, we have to control the light each photosin receives. This control is created using a pinhole imaging. The pinhole imaging, introduces a diaphragm that has an infinitesimal hole. In theory, this diaphragm creates a barrier blocking everything but a single ray to each photosin, as visualised in Fig. 3.6. It is defined as the aperture of the camera, which enables the variation amount of light that hits the sensor. As a result, we have the creation of a 2D copy of the real world, but this image is inverted and scaled.

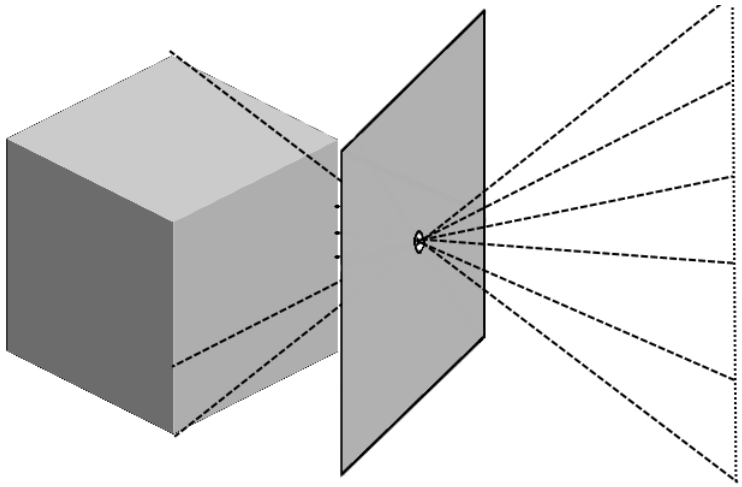

Figure 3.6: Pinhole

In the pinhole model, the only amount of light that goes thought and hits the sensor is the one that passes the pinhole. This means that we can not take advantage of the multiple paths of the projection of light from the scene, which is why pinhole camera systems have been replaced with cameras with a lens system for most of the cases. The lenses acquire the multiple light path from the scene and converge the light rays into a single point, allowing the amount of light that reaches the sensor to be much more intense, providing a better image. This can be visualised in Fig. 3.7. To get the best of the cameras, we have to consider two important characteristics, aperture and focal length. As seen before, the aperture is directly connected with the pinhole itself, and it allows that more or less light to hit the sensor. However, more light results from more rays from different scene point hitting a single photosin, blurring the image. Focal Length can be confusing as it differs from camera to camera system. In the pinhole camera system, the focal length is an intrinsic parameter of the lens and is the same as the focal distance. It can be described as the distance between the aperture plane and the image plane. In systems with lenses, a Gauss lens model must be taken into consideration, and, in this case, the focal length (f) is the distance where the parallel rays intersect, the focal distance (D') is the distance from the camera aperture plane to the image plane, and the object distance is the distance between the lens and the object (D). This can be visualised in Fig. 3.7 and the equation of Gauss lens model in Eq. 3.1.

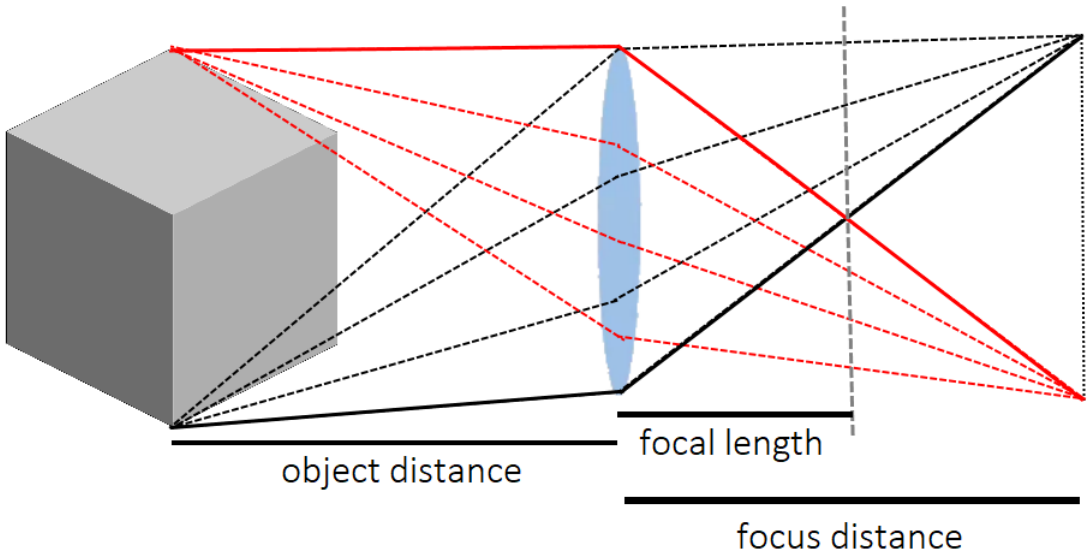

Figure 3.7: GaussLensModel

$$\frac{1}{D} + \frac{1}{D'} = \frac{1}{f} \tag{3.1}$$

The camera model can be simplified, allowing the pinhole camera model with lenses as it is possible to derive properties and descriptions that hold for both camera models if:

- we only use central rays
- we assume the lens is in focus
- we assume that the focus distance of the lens is equal to the focal length of the pinhole camera

3.2.3 Perspective Projection Model

The 3D world space can be projected into a 2D image via a transformation from the euclidean world system to a correspondent 2D point in a plane, as illustrated in Fig. 3.8.

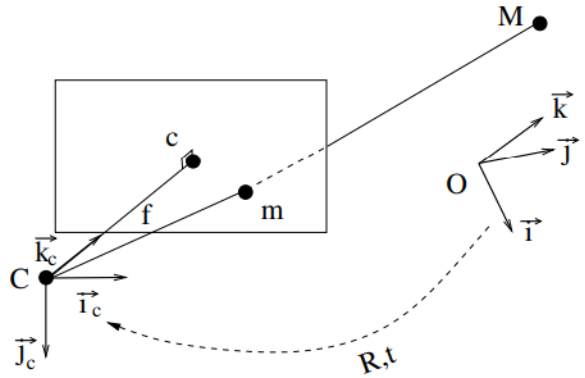

Figure 3.8: Perspective Projection Model (from [31])

If we consider a 3D euclidean point as Eq. 3.2 and the corresponding 2D points as Eq. 3.3, both in homogeneous form,

$$M^{'} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^{T} \tag{3.2}$$

$$m^{'} = \begin{bmatrix} u & v & 1 \end{bmatrix}^{T} \tag{3.3}$$

Given the scale factor natural from the pinhole camera implied by the focal length as seen before, let us call such scale as “s”. We can estimate a 3x4 matrix “P” that transforms the 3D euclidean points into the 2D representation, as in Eq. 3.4. The projection matrix “P” is defined up to a scale factor and thus is dependent on 11 parameters. When applied to the camera, the matrix projection can be decomposed into the Eq. 3.5.

$$sm^{'} = PM^{'} \tag{3.4}$$

$$P = K[R|t] = K \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{3.5}$$

3.2.4 ”Rearranged” Pinhole Camera Model

In order to use all the equations seen in the previous sections we will assume that the image plane from the pinhole is not behind the Aperture of the camera, but in front of it as represented in Fig. 3.9 and assuming the triangle relations of Eq. 3.6,

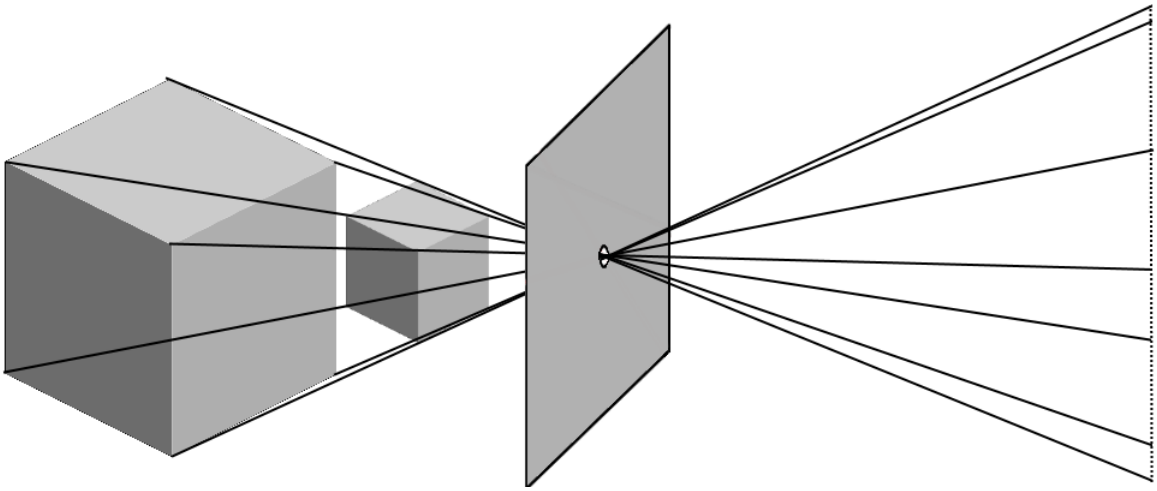

Figure 3.9: Rearranged Pinhole Camera

$$[XYZ]^T \rightarrow \left[f\frac{X}{Z}, f\frac{Y}{Z}\right]^T \tag{3.6}$$

It is possible to represent the camera model as described in Eq. 3.7:

$$P = K[I|0] = \begin{bmatrix} fx & s & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.7}$$

Where K is a 3x3 matrix representing the intrinsic parameters of the camera. It is a homogeneous transformation from 2D to 2D, accounting for not unit focal length and origin shift. $[I|0]$ is a 3×4 matrix representing the extrinsic parameters of the camera. It is a homogeneous transformation from 3D to 2D, assuming image plane at $Z = 1$ and shared camera/image origin. However, we need to know the transformation of the camera to the camera coordinate system, which can be obtained via an homogeneous rigid body transformation as in Eq. 3.8. Combining Eq. 3.7 and Eq. 3.8, we can create a complete camera model as represented in 3.9, which is similar equation as Eq. 3.4.

$$\begin{bmatrix} X_c & Y_c & Z_c \end{bmatrix}^T = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w & Y_w & Z_w & 1 \end{bmatrix}^T \tag{3.8}$$

$$P = K[R : -RC] = K[R|t] = \begin{bmatrix} fx & s & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{3.9}$$

3.2.5 Camera Parameters

Based on Eq. 3.9, it is possible to represent the camera model and divide its parameters as intrinsic and extrinsic.

- **Intrinsic Parameters (Defined by the matrix K)**
  Answers the question: What kind of camera?
  - **Focal Length fx,fy:** distance between the camera aperture and the image plane. In the ideal camera they are they are the same;
  - **Principal Point cx,cy:** image coordinates that represent the perpendicular intersection of the optical axis and the image plane. In the ideal camera they are located at the center of the image, as represented in Fig. 3.10;

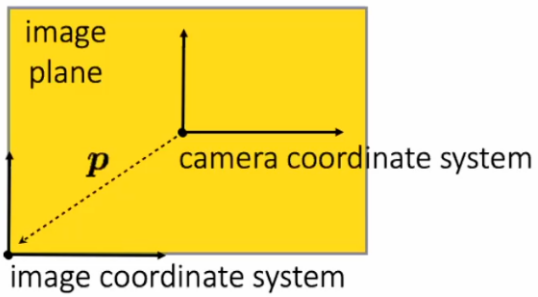

Figure 3.10: Camera Coordinate System To Image Coord System (from [20])

  - **Skew s:** This parameter is a representation of the angle between the horizontal and vertical axis. It can usually be considered 0 in modern cameras as the grid of pixels is close to perfectly perpendicular. In Fig. 3.11 it is possible to visualise how the value change based of the axis angle,

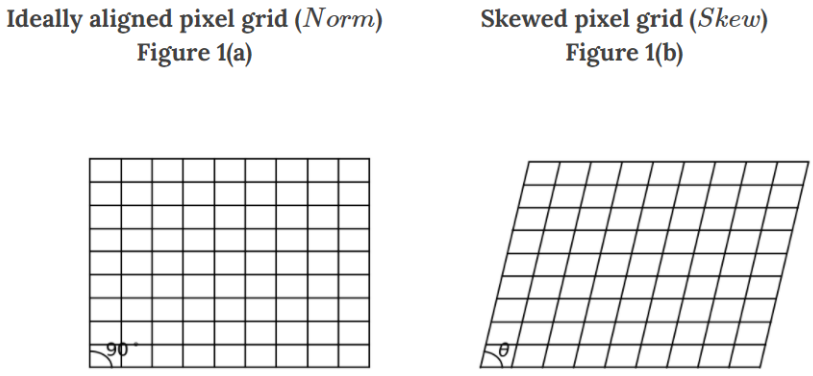

Figure 3.11: Skewed Pixel Grid (from [10])

  A camera is considered calibrated if the internal parameters are known.

- **Extrinsic Parameters (Defined by the matrix [R|t])**

Answers the question: where is the camera?
The extrinsic parameters represents the translation and the rotation of the camera. They are defined as a 3x4 matrix and define the camera pose, which is a representation of the rigid body transformation from the euclidean coordinate system of the real world to the camera coordinate system.

- **Rotation Matrix (R):** 3x3 Matrix representing the rotation;
- **Translation vector(t):** 3x1 vector representing the translation,

Accounting the 4 DOF from the internal parameters, assuming skew equals 0, and the 6 degrees of freedom from the external parameters, we have a total of 10 DOF in the model.

3.2.6 Lens Distortion

In the real world, it is challenging to build a perfect camera lens. Thus, an additional computation must be done to map the camera pixel's most suitable position related to the scene because of the distortion that the lens suffers. The distortion created by the lens makes this mapping non-linear and still a big research topic to minimize the distortion effects. Multiple models can approximate this 2D deformation. The one used in this project is based on the work of the "Brown-Conrady" model (Brown[7] continued Conrady[14] work) that presented a mathematical model based on series to approximate such distortion and later improved by Zhang [68]. This work proposed taking the first elements of an infinite series to get a decent approximation of the lens distortion, where the radial distortion is represented in Eq. 3.10 and tangential distortion in Eq. 3.11.

$$\begin{aligned} x_{distorted} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{distorted} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{3.10}$$

$$\begin{aligned} x_{distorted} &= x + [2p_1 xy + p_2(r^2 + 2x^2)] \\ x_{distorted} &= y + [p_1(r^2 + 2y^2)2p_2 xy] \end{aligned} \tag{3.11}$$

As a result, the lens distortion can be defined if the five parameters $[k_1,k_2,p_1,p_2,k_3]$ are known.

3.2.7 Camera Calibration using Zhang's method

An option to estimate the camera matrix widely adopted is the correspondence between known 3D euclidean points in the real world and 2D points of the image. Using Zhang's method [8], the correspondences can be obtained using a fixed camera and a moving pattern or a moving camera and a fixed pattern or both. The important is having multiple poses (at least 2). Zhang proposed a method that reduces the complexity of the DLT algorithm by reducing the number of degrees of freedom by defining the calibration pattern on a plane which is typically the corners on a white and

black grid. By doing this, it enables the calibration of the camera to be done using homography matrices.

Estimating Homography For Each View

Corresponding the 3D euclidean points to the correspondent 2D image coordinates based on Eq.3.4, assume that the camera's intrinsic parameters are common in every view, and the scale "s" is arbitrary and non-zero. If we use a cloud of points like the one used with the chessboard method, the Z-axis is assumed to be 0 and so we have a plane rotation in the X-axis and Y-axis. In this case we dont need to use the full Eq. 3.4, as $Z = 0$ is redundant, we can substitute the projective matrix "P" by an homography matrix "H", creating a correspondence between the planes created by the point cloud of the 3D points that are now represented as 2D plane as follows:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \alpha H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3.12}$$

Direct Linear Transformation

The Direct Linear Transformation method (DLT), assumes that the planes created by the point cloud can be related using homographies matrix by expanding the linear Eq. 3.13, taking the scale factor Eq. 3.14 and rearranging the terms Eq. 3.15.

$$\begin{aligned} u &= \alpha H_{0,0} X + H_{0,1} Y + H_{0,0} \\ v &= \alpha H_{1,0} X + H_{1,1} Y + H_{1,0} \\ 1 &= \alpha H_{2,0} X + H_{2,1} Y + H_{2,0} \end{aligned} \tag{3.13}$$

$$\begin{aligned} u &= \frac{H_{0,0} X + H_{0,1} Y + H_{0,0}}{H_{2,0} X + H_{2,1} Y + H_{2,0}} \\ v &= \frac{H_{1,0} X + H_{1,1} Y + H_{1,0}}{H_{2,0} X + H_{2,1} Y + H_{2,0}} \end{aligned} \tag{3.14}$$

$$\begin{aligned} H_{2,0} X u + H_{2,1} Y u + H_{2,0} u - H_{0,0} X - H_{0,1} Y - H_{0,0} = 0 \\ H_{2,0} X v + H_{2,1} Y v + H_{2,0} v - H_{1,0} X - H_{1,1} Y - H_{1,0} = 0 \end{aligned} \tag{3.15}$$

By writing it in matrix form and stacking together the constraint from multiple point correspon-

dences Eq. 3.16 is obtained, which is typically written as 3.17.

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & X & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & X & yx' & x' \\ & & & & ... & & & & \\ -x_i & -y_i & -1 & 0 & 0 & 0 & X_i & yx'_i & x'_i \\ 0 & 0 & 0 & -x_i & -y_i & -1 & X_i & yx'_i & x'_i \end{bmatrix} \begin{bmatrix} H_{0,0} \\ H_{0,1} \\ ... \\ H_{i,0} \\ H_{i,1} \end{bmatrix} = 0 \tag{3.16}$$

$$Ah = 0 \tag{3.17}$$

We are now facing a homogeneous linear least squares problem where the obvious solution would be h=0. However, this solution is of no interest, so the solution lies in the Single Value Decomposition (SVD for short) of A.

$$A = USV^T \tag{3.18}$$

From the Eq. 3.18 above the matrix S is obtained, which is a $2n \times 9$ diagonal matrix of singular values, all of them non negative. Each column of V represents a solution for $Ah = 0$ and the eigenvector corresponds to the minimal eigenvalue. Of interest is a vector in V that corresponds to the smallest singular value of S. Usually more than 4 points are introduced and in that case an overdetermined solution is calculated. In order to get the best result, it is possible to solve a minimal least square problem as follows:

$$||M.h||^2 \rightarrow min \tag{3.19}$$

In many of SVD solutions, the matrix S is given so that $\mathbf{s_i}, \mathbf{i} = \mathbf{0}$,...,$\mathbf{8}$ are ordered in not incremental values, which means that the solution to the least square problem lies on the single value $\mathbf{s_8}$.

To achieve brevity, we will stop here, but further improvements can be made. Once we have a set of homography matrices, it is possible to estimate the intrinsic and extrinsic parameters of the camera radial distortion and finally refine all parameters by minimizing the total projection error.

3.2.8 Perspective n Points

Estimating the camera's external parameters is commonly done after the internal parameters have been estimated via a calibration process, called "Perspective n Points" (PnP). It allows for a more computationally intensive calibration because it can usually be done only once and offline, fur-

thermore, the fixed internal parameters help achieve robustness. Extraction of 3D points and establishing a correspondence to the 2D points of the image plane are required to estimate external parameters. In most cases, a minimum of six correspondences is required. However, for multiple approaches, if the selected points are coplanar and non-collinear, it may be reduced to only 4 points, which is why fiducial squared markers are so attractive. It must be considered that based on the number of 3D to 2D correspondences, the typical methods difficulty and computational effort required scales fast.

EPnP - Eficcient Perspective n Point

This approach solves the PnP problem in a considerably more efficient, non-iterative, and time-consuming manner. The algorithm's prominent characteristic is that the number of unknowns grows considerably more slowly than other techniques as the number of correspondences between 3D and 2D points grow [32]. In other words, this proposal's computing complexity grows linearly with the number of correspondences. Furthermore, it can solve the PnP issue with just three planar configuration correspondences and four non-coplanar configuration correspondences (the general case), which is ideal to use with fiducial squared markers.

Instead of solving for the depths of the reference points in the camera coordinate system, this technique expresses their coordinates as a weighted sum of virtual control points. Because of this, the number of unknowns grow much slower than previous approaches.

Having n 3D reference points in the world coordinate system as seen in Eq. 3.20 and four control points expressed in Eq. 3.21, it is possible to express the reference points as a weighted sum of control points as seen in Eq. 3.22 in the world coordinate system and Eq. 3.23 in the camera coordinate system. α_{ij} are homogeneous barycentric coordinates, which are uniquely defined and easy to estimate.

$$\mathbf{p}_i, i = 1,...,n \tag{3.20}$$

$$\mathbf{c}_i, i = 1,2,3,4 \tag{3.21}$$

$$\mathbf{p}_i^w = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^w, \alpha_{ij} = 1 \tag{3.22}$$

$$\mathbf{p}_i^c = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c \tag{3.23}$$

The vector that resides in the kernel of a matrix $\mathbf{M}$, of size $2n \times 12$ (general case), is the answer to this problem, and it is equal to the weighted sum of the null eigenvectors. To extract it, DLT like technique is used. Let $\mathbf{A}$ be the intrinsic parameters of the camera, $\mathbf{u}_{i=1,...n}$ the 2D projections

of the reference points $\mathbf{p}_{i=1,...n}$, and ω_i the scalar projective parameters. The resulting Equation is shown in Eq. 3.24 and, once expanded, results Eq. 3.25.

$$\forall i, \omega_i \begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} = \mathbf{A}\mathbf{p}_i^c = \mathbf{A} \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c \tag{3.24}$$

$$\forall i, \omega_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} \tag{3.25}$$

Developing Eq. 3.25, we end up with Eq. 3.26 and Eq. 3.27, which by concatenating all n reference points generates a linear system as shown in Eq. 3.25, where $\left[c_c^{1T}, c_c^{2T}, c_c^{3T}, c_c^{4T}\right]$ is a vector of 12 unknowns, $\mathbf{M}$ is a $2n \times 12$ matrix that results by rearranging the coefficients.

$$\sum_{j=1}^{4} \alpha_{ij} f_u x^{jc} + \alpha_{ij}(u_c - u_i) z_j^c \tag{3.26}$$

$$\sum_{j=1}^{4} \alpha_{ij} f_v y^{jc} + \alpha_{ij}(v_c - v_i) z_j^c \tag{3.27}$$

$$\mathbf{M}\mathbf{x} = 0 \tag{3.28}$$

The solution is then found in the kernel of $\mathbf{M}$, where the set $\mathbf{v}_i$ in Eq. 3.29 are the right-singular vectors of $\mathbf{M}$ corresponding to the null singular values of $\mathbf{M}$, which are found efficiently because the null eigenvectors of $\mathbf{M}^T\mathbf{M}$ have a constant size of 12×12 and this computation has O(n) complexity.

$$\mathbf{x} = \sum_{i=1}^{N} \beta_i \mathbf{v}_i \tag{3.29}$$

This explanation will stop here as the essential knowledge to understand how the algorithm obtain such good performance is already explained. But it is advised to read the original paper in order to understand the next stages, where it it shown how to choose the right linear combination where β_i is estimated.

In sum, the necessary weights are computed cheaply by solving a few quadratic equations, which are fast to compute. Therefore, when the number of correspondences are above 15 (according to the author's tests), the algorithm's computational effort is based on the $\mathbf{M}^T\mathbf{M}$, which scales linearly with the number of 3D-to-2D point correspondences.

3.3 RANSAC

"RANdom SAmple Consensus" (RANSAC) is an algorithm for ensuring the robustness of estimations [67]. RANSAC has the benefit over minimization approaches since it does not require an initial guess. Even if it is less accurate, its output might be utilized as the first guess of minimization methods to enhance accuracy. It is non-deterministic because the solution provides the minor portion of the total inputs that solves a specific algorithm. Because the inputs of RANSAC are randomly chosen, it is conceivable that the algorithm will not produce the correct answer. The method performs several iterations with random inputs to increase the chances of obtaining the solution, and the iteration with the highest number of inliers or with a high enough percentage of them is the one that yields the final answer. RANSAC can be used merely to select inliers for later use in a better method rather than presenting a solution.
In Fig. 3.12, we can find a flowchart of a typical RANSAC algorithm.

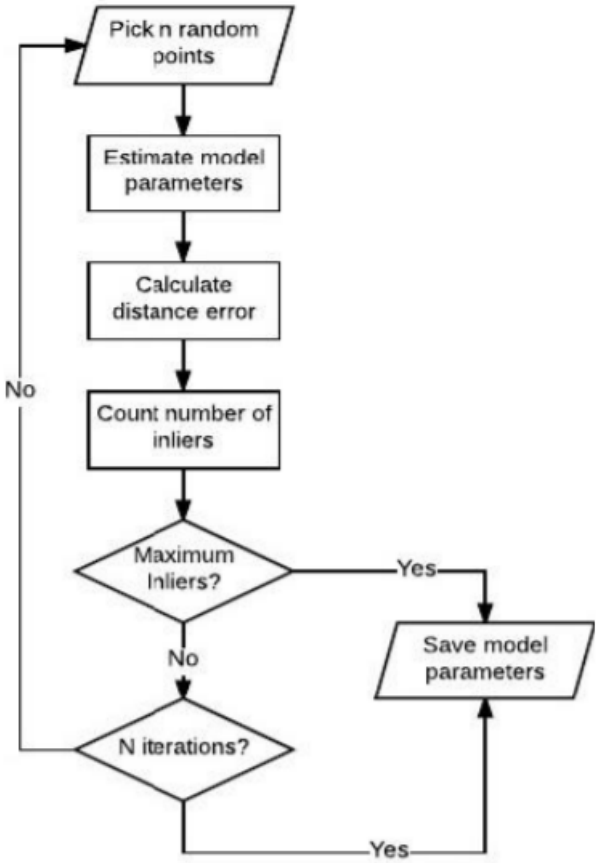

Figure 3.12: General RANSAC Flowchart (from [67])

By defining "n" as the number of inputs per iteration, "p" as the desired probability of success, "w" as the probability of a single input in a particular iterarion being a inlier (fraction between the number of inlers divided by the total number of inputs, "$0 \leq w \leq 1$)", and finaly "k" as the total number of iterations, it is possible to estimate the minimum number of iteration required to hit a

certain "w". Estimating "k" can be done by knowing that the probability of never selecting a set filled with inliers in "k" iterations is equal to the probability of selecting at least one inlier per set. Therefore we have Eq. 3.30 which leads to 3.31 applying the logarithm to both terms and isolating "k".

$$1 - p = (1 - w^n)^k \tag{3.30}$$

$$k = \frac{1 - p}{1 - w^n} \tag{3.31}$$

When using RANSAC, the user must take into consideration that the result and k value vary based on "n", "w", and "p". In general, enlarging "w" and lowering "n" and "p", "k" is reasonable. Therefore, the common essential inputs of RANSAC are the total input data, the maximum number of iterations, and the desired probability of success because, if the desired probability is hit, the algorithm can stop before completing all the iterations. Depending on the previous parameters, it may take longer than expected even when the estimations run parallel, taking advantage of multiple threads.

Ransac applied to the PnP Problem

In this project scope, RANSAC is applied in two instances to the estimation the of camera pose. In the first, RANSAC runs the EPnP algorithm by inputting four correspondences of 2D to 3D points as they are the minimum amount to obtain the general case of the pose estimation. In the second, RANSAC is used merely as an outlier filter for the EKFPnP (discussed in section 4.3).

3.4 Kalman Filter

Kalman filter, named after Rudolf Emil Kalman (1930-2016), the primary developer of its theory. Kalman filter is an optimal estimation algorithm that tries to approximate the real values based on measurements taken over time. This filter has the characteristics of being recursive and predictive, allowing to estimate future states based on the past and present, even in the presence of noisy measurements. Furthermore, it allows the estimation of the system state when the measurement cannot be measured directly and enables the fusion of multiple sensor measurements that might input their observations at different frequencies. Merging their data improves the estimation and correct possible errors mutually. It completely revolutionized the field of estimation and control systems and has been subject to intense research throughout the years, suffering multiple alterations to adjust it to multiple applications.

Taking advantage of this filter, a widespread use case is in the GPS of vehicles. In particular, the sensors can estimate their absolute position relative to a satellite and estimate the car's position over time using error-prone inertial sensors. Suppose less uncertainty is given to the GPS signal. Even if the inertial sensors drift, their errors are minimized every time the satellite signal updates the car's absolute position. On the other hand, even the little weight given to the inertial systems

corrects for possible significant deviations of the absolute position. In this case, we have the inertial sensors that cause incremental uncertainty due to their natural double differentiation in the estimate but are excellent when calculating small translations and rotations. The satellite signals can have noisy data but give the absolute position of the car. Together, they work amazingly well given the fast frequency but drift-prone inertial sensors and the low frequency but the reliable position of the satellite system. Other common use cases are the improved tracking of objects, such as an aeroplane or a ball in a sports event [1], and, more related to our project, the filtering of poses. Following the last use case, the use of robots filtered poses by an Extended Kalman Filter results in the possibility to map the markers based on the robot's movement [33]. On [45], the authors propose the fusion of visual and inertial data to improve the camera pose estimation while being robust against fast motions and changing lighting conditions. The Extended version of the filter will be discussed in section 3.5.
In sum, Kalman Filter is an iterative mathematical process that uses a set of equations and consecutive data inputs to quickly estimate the object's actual value under measurement, especially when the measured values have unpredicted/random error, uncertainty, or variation. While being fantastic to estimate future predictions, all of this because, even though the algorithm starts by taking an initial estimate, it does not seriously affect the filter's performance as it quickly narrows to the actual value as new data enters the system. As new data enters the iterative system and the estimation obtained narrows to somewhere close to the actual value, the variations will get exceedingly small, reducing the influence of measurement errors.

Kalman Filter Basis - Flow Chart

In every iteration, the filter solves three main equations:

- Calculate the Kalman gain
- Calculate the current estimate
- Calculate the new uncertainty in the estimate

In Fig. 3.13, it is shown a flowchart of a general Kalman Filter.

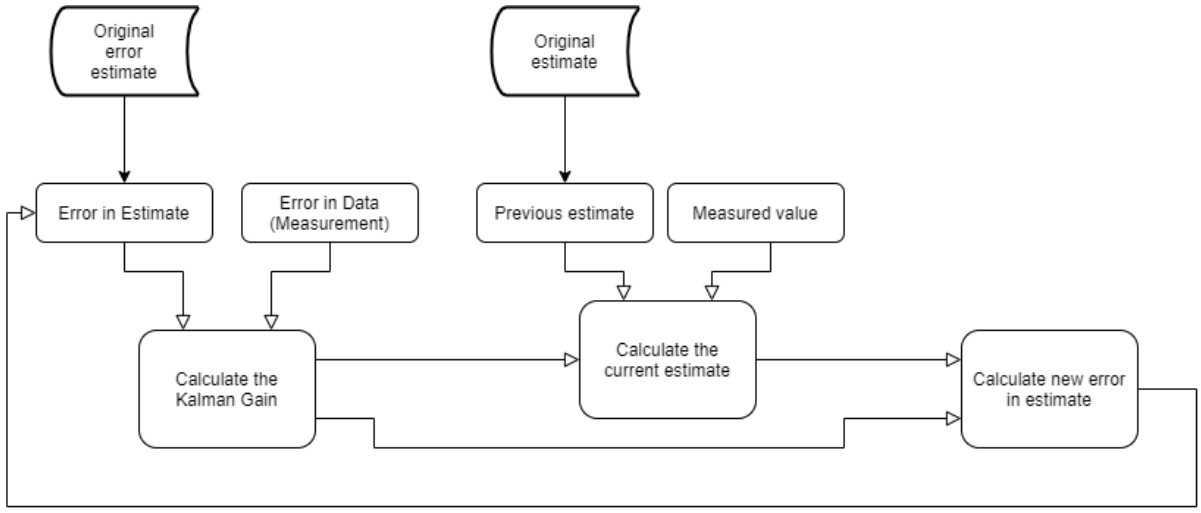

Figure 3.13: Kalman Filter General Flowchart

Kalman Filter Basis - Kalman Gain

The Kalman gain allows the algorithm to weigh the importance of the estimate's uncertainty against the measurements' uncertainty. If the estimate's uncertainty is smaller than the data's, it will have more importance, contributing more to the estimation and vice-versa. What feeds the current estimate's recalculation depends on how much we trust the previous estimate and the sensors' data. In other words, when the Kalman gain is significant, the error in the estimate is large compared to the error in the measurement. Moreover, when the Kalman gain is small, the error in the measurement is significant compared to the error in the estimate. This relation between the estimate and measurement errors are relatable by the Eq. 3.32.

$$KG = \frac{E_{est}}{E_{est} + E_{mea}}, 0 \leq KG \leq 1 \tag{3.32}$$

Current Estimate

The current estimate is calculated based on the previously calculated Kalman gain and the previous estimate due to a feedback loop, except the first iteration where the user gives this previous estimate. As the filter converges very quickly to the correct value in a few iterations, the original estimate does not have to be exact as the Kalman Gain will provide more weight to the measurements in the initial iterations, thus reducing its effects. Besides the previous estimation, we also require a new data measurement to improve the Kalman filter. However, the data measurement is optional because the Kalman filter is predictive, which means that even without measurement, the filter will provide an estimation, predicting the next state. This prediction works well, but keep in mind that the estimation's uncertainty will increase every iteration if no new data is introduced to the system due to no corrections being applied. If no measurement is taken to correct the estimation for a while, eventually, the uncertainty will be so considerable that we cannot trust the prediction.

The Kalman is then used to calculate the new estimate represented in Eq. 3.33.

$$EST_t = EST_{t-1} + KG(MEA - EST_{t-1}) \tag{3.33}$$

Summing, as the Kalman Gain varies from 0 to 1, depending on the current estimate and the measurement, the higher it is, the more weight is given to the measurements, meaning they are trusted more, and the estimates are precarious. On the other hand, the lower the Kalman Gain, the lower the trust of the measurements and thus, more weight will be provided to the estimation.

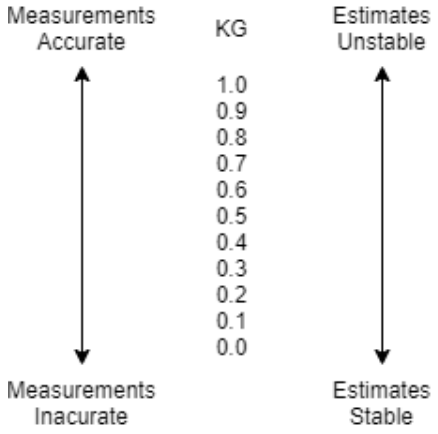

Figure 3.14: Kalman Gain Variations Based On The Estimation and Measurement Errors

As iterations pass, the Kalman Gain gets lower and lower as the predictions get better and better, meaning that the Kalman Gain would provide estimations primarily based on the prediction and only a tiny portion by adjustments based on the observations, thus ignoring erroneous measurements.

Calculate New Errors

Once the new estimate is calculated, the new errors based on the new estimate are calculated. This error update must happen because the new errors are calculated compared to the errors from the previous iteration. Knowing the current estimate and the gain based on the current estimate is mandatory to calculate the new errors.
Take into consideration that the equation factors the multiplication of the inverse on the Kalman Gain, meaning that if the Kalman Gain is large, the errors in the measurements are minor. On the other hand, if the Kalman Gain is small, the errors in the measurements are significant, as represented in Eq.3.34.

$$E_{EST} = (1 - KG)(E_{EST_{t-1}}) \tag{3.34}$$

Kalman Filter - Multidimensional Model

Fig. 3.15 presents the Kalman Filter in a multidimensional model, allowing multiple parameters to enter the filter in matrices representation.

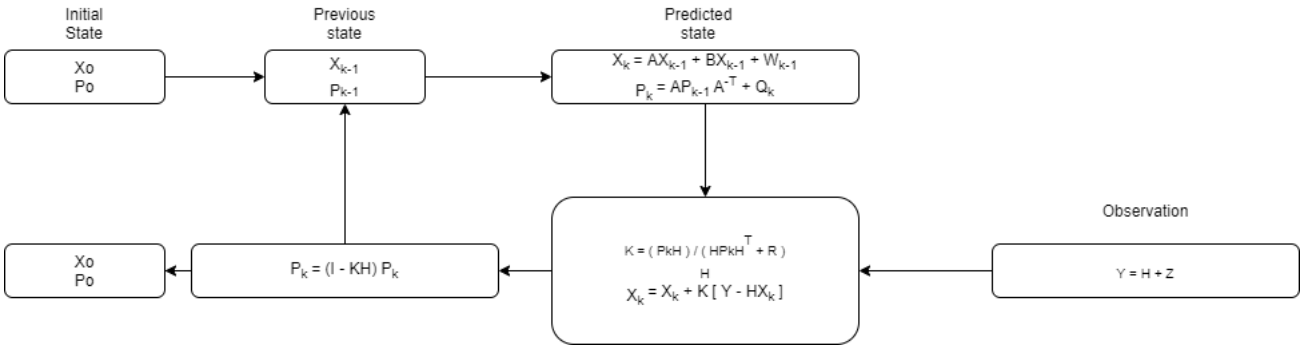

Figure 3.15: Kalman Multi Dimensional Model

- x_k = State Vector
- x_{k-1} = Previous state Vector
- P = Process Covariance Matrix
- K = Kalman Gain
- R = Sensor Noise Covariance Matrix
- I = Identity Matrix
- Y = Measurement of the state
- Z_k = Measurement Noise
- u = Control Variable Matrix
- Q = Process Noise Covariance Matrix
- A - Status transition Matrix
- B,C - Adaptation Matrices (Converts from one form to the other)

In general, Kalman filter iterations are divided in two main steps, Prediction and Correction.
The Prediction Multidimensional equations, based on the system model and the previous state are defined as:

- Predicted state:

$$x_k = Ax_{k-1} + Bu_k + W_k \tag{3.35}$$

- Predicted Covariances:

$$P_k = AP_{k-1}A^{-T} + Q_k \tag{3.36}$$

The Correction Multidimensional equations, update the Predictions based on the new measurements and the Kalman Gain that results from the comparisons between the predicted and new

measurements errors. Such equations are defined as:

- Kalman Gain:
$$K_k = \frac{P_k H^T}{(H P_k H^T + R)} \tag{3.37}$$

- Current State:
$$x_k = x_k + K[z_k - H x_k] \tag{3.38}$$

- Current Covariances:
$$P_k = (I - KH) P_k \tag{3.39}$$

3.5 Extended Kalman Filter

Because non-linear systems are highly prone to surge in the vast amplitude of applications, a non-linear version of the Kalman Filter method must be taken into consideration.
Such an approach estimates an approximation of the non-linear system via the first-order non-linear Taylor expansion. It works as long as the system state and observation equations are close to linear and continuous [25], which means that the equations used in the Extended Kalman Filter are the same as the standard Kalman Filter but linearized in the operation point.
Therefore, the equations of the Prediction phase of the filter are:

$$s_{k|k-1} = f(s_{k-1|k-1}, u_k) \tag{3.40}$$

$$P_{k|k-1} = F_{k-1} P_{k-1|k-1} F_{k-1}^T + G_{k-1} Q_{k-1} G_{k-1}^T \tag{3.41}$$

- $s_{k|k-1}$ - Priori State Vector
  - f(.) - nonlinear dynamic model of the process
  - u_k - dynamic model noise vector.
- $P_{k|k-1}$ - Priori Covariance Matrix
  - F_k - derivative of the dynamic model with respect to **s**.
  - G_k - derivative of the dynamic model with respect to u_k.
  - Q_k - noise covariance

As described before, if there are observations, they will be considered in the correction phase. In this phase, the equations of the filter are:

$$s_{k|k} = s_{k|k-1} + K_k(z_k - h(s_{k|k-1})) \tag{3.42}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{3.43}$$

$$K = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + U_k)^{-1} \tag{3.44}$$

- $s_{k|k}$ - Posteriori State Vector
  - z_k - observations
  - h(.) - observation model mapping the state space into the observed space z_k
- $P_{k|k}$ - Posteriori Covariance Matrix
  - H_k - is the derivative of h(.) with respect **s**
- K - Kalman Gain
  - U_k - observation noise covariance

4

Mapping Artificial Keypoints Based on Fiducial Markers

This section aims to overview the proposed methodology for artificial mapping keypoints based on fiducial markers to perform camera localisation in real-time. Once defined, an improvement to the PnP problem is implemented using Extended Kalman Filter, in which uncertainty is reused to improve the mapping process.

4.1 Methodology Overview

In this methodology, the user navigates freely in the real world with a monocular camera system such as a smartphone, and a method of computer vision extracts the important information about the environment called keypoints. If an essential element of the world can be identified, it is taken as the reference frame followed by an iterative process that searches for other elements that can be easily identified and robust in the future. If it succeeds, the algorithm then tries to relate them to create a topologic map of the environment's elements.

The elements chosen to create this map are artificial keypoints based on squared fiducial markers called "ArUco". These markers have the advantage of being easily detected and can be identified by their internal black and white grid that defines their ID. Furthermore, their location and orientation can be effortlessly estimated using the previously described algorithm "Perspective n Points". This algorithm has the characteristic that if the points are coplanar and there are no triplets of collinear points, then the solution is unique if at least four correspondences are provided. Because of this, the four corners of each marker are enough to estimate its relative pose to the camera. The correspondences provided to the PnP algorithm are the 2D correspondences of the fours corners in the image plane and its four 3D correspondences in the real world. By placing each marker frame in its centre, the 3D points corresponding to its corners are in the X-Y plane. Therefore, the four possible combinations for its coordinate system are (+-size/2, +-size/2, 0). This allows the update of the topologic map if at least two markers are visible in an image and their relative pose to the camera is known.

As the position of the markers is known, a 3D point cloud can be created by transforming each marker corner by the rigid body transformation obtained by its relative pose to the reference marker.

This point cloud is the key to improving camera pose estimation by using the PnP algorithm once again. However, its input is now based on all the markers detected in the scene at each camera frame, enabling a more robust and smoother camera pose estimation over time.
Keeping in mind that the markers are related in a tree-like relation, it must be considered that the relation between a marker to the marker that it is linked with has an associated error that tends to grow as the distance to the tree root increases. It depends on various factors, such as the resolution of the camera or the relation marker-camera itself.

The Fig. 4.1 presents a visualisation of the flowchart of the required steps for mapping the markers and pose estimation.

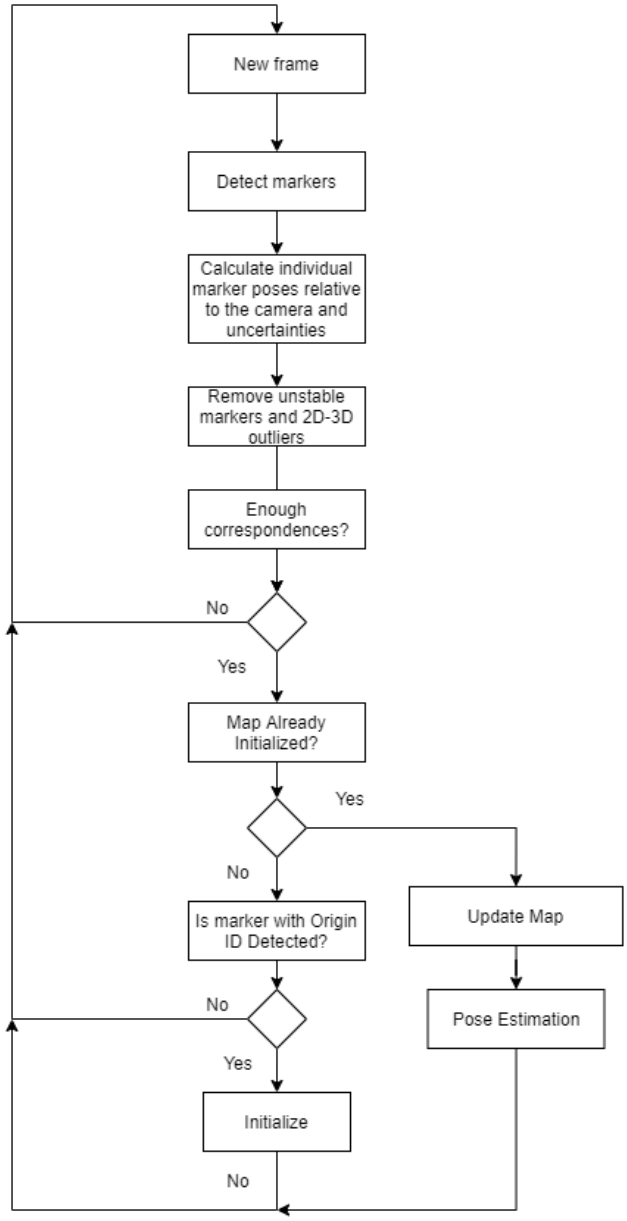

Figure 4.1: Marker Methodology Flowchart

4.2 Naive Mapping Process

Every iteration of the algorithm, starts by reading the frame provided by a video sequence, either live or pre-recorded, and sends each frame to the ArUco Detector algorithm, which returns the 2D corners, ID and pose of the camera relative to each marker detected as visualised in the Fig. 4.2.

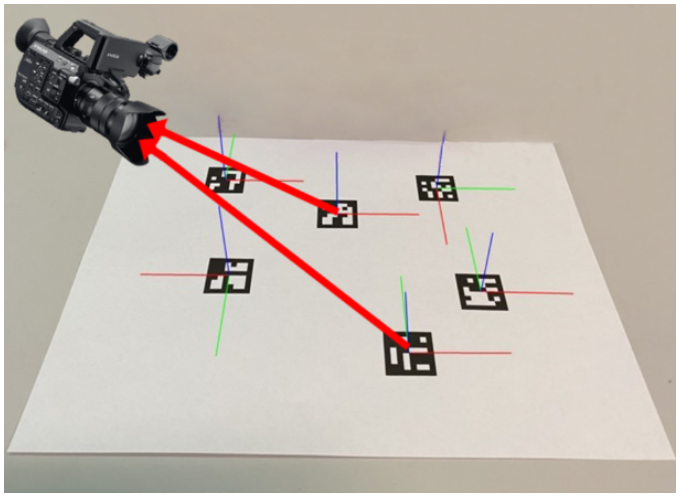

Figure 4.2: Detection of multiple markers

This naive implementation was initially developed as a proof of concept, as well as to study the difficulties of the mapping process. Following this implementation, multiple improvements were developed.

Map Initialization

The naive implementation of the mapping process is an iterative process that starts by looking for the marker with the ID that the user chooses to be the map's origin. When detected, the algorithm adds this marker information to the root of the tree-like data structure, allowing future markers to be added. This first node will have no parent, and therefore, it is hardcoded to have the parent ID "-1", and the relative pose to it is the identity matrix by default, meaning no translation or rotation is applied to it (it is on the origin), this identity matrix can be changed to allow the changing of the origin as the user see fit.

Adding New Markers

As new markers are detected, if at least one of them is already mapped, it is possible to estimate the relative poses between two markers to map the unmapped ones. If that is the case, the new marker is added to the map relative to one previously present. Estimating the relative pose between the two markers requires estimating each one individually using its four corners as described before. Once the camera pose to each marker is known, the parent pose relative to the camera can be obtained by calculating the inverse of the pose matrix that maps the camera to the marker. The last piece of the puzzle is the product between the pose of the child marker with the inverse pose of the parent calculated previously.

In summary, by knowing the camera's pose relative to each detected marker, we can estimate the pose of any detected marker relative to any other, allowing the creation of the marker map.

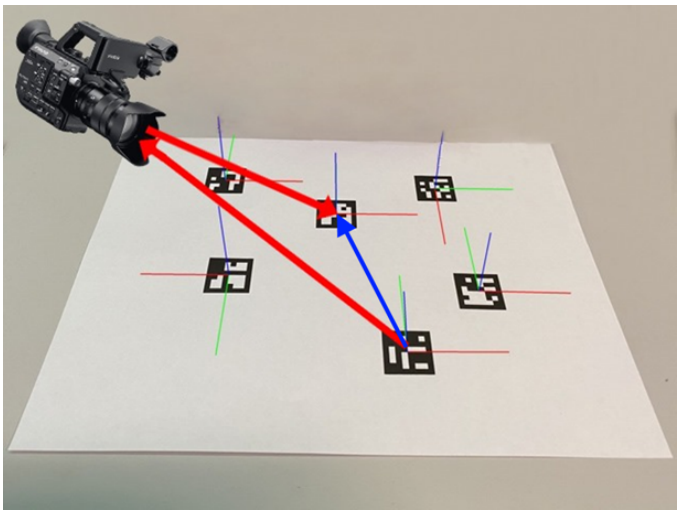

Figure 4.3: Relative pose between two markers

The more markers captured, the more markers get mapped, resulting in a higher quantity of 2D-3D correspondences available for the PnP pose estimation algorithm.

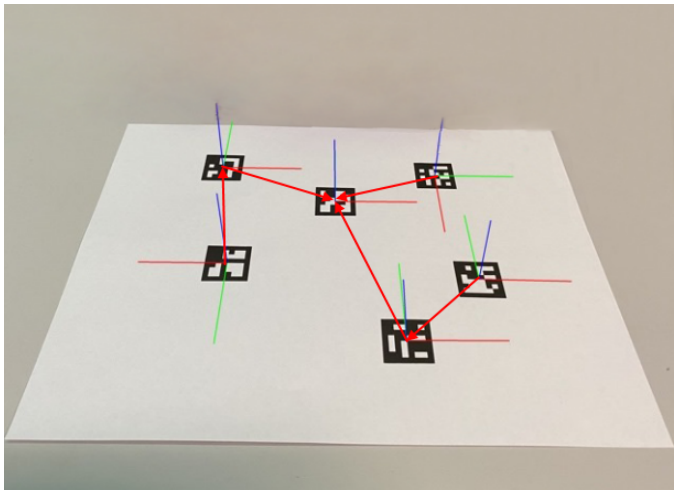

Figure 4.4: Relative pose between multiple markers

In the ideal scenario, the mapping process would work well. However, we must consider that marker detection is noisy, resulting in accumulative errors from erroneously mapped markers. The detection errors can have a vast source, such as the limited camera resolution, lighting, shadows, contrast, flickering caused by the electrical system frequency, among many other causes. Therefore, we must implement an algorithm that calculates how much the pose of each marker can be trusted using uncertainties.

4.3 Improving Pose Estimation Using EKFPnP

In order to improve the results obtained previously, the implementation of EKFPnP was done, following the work [36]. In this work, the authors developed an algorithm that uses the Extended Kalman filter to improve the estimation of the pose based on the PnP problem by taking advantage of the temporal dependency of the camera pose and the uncertainty of features. This implementation leads to an improved estimation of the localisation and orientation of the camera, being the last presented in quaternion representation. This representation deals with the constraint rotation matrix orthonormal constraint $RR^t = I$ and the determinant constraint $det(R) = 1$, while more compact (four parameters). Furthermore, they can be easily converted to rotation matrices if required and do not suffer from the gimbal lock problem, where at least one degree of freedom is lost (ambiguity), typical of the Euler representation because they require three rotations to be processed. On the other hand, quaternions encode only one well-defined rotation, avoiding the gimbal lock and being faster and more efficient to compute when compared to rotation matrices, providing a sweet spot between memory utilization and speed.

Based on Eq. 3.9, the transformation of a 3D point $\mathbf{x}^w = [x_i,x_k,x_k]$ referenced to the world coordinate system can be expressed as shown in Eq. 4.1 and expressed relative to the camera center as shown in Eq. 4.2.

$$\mathbf{x}^c = \mathbf{R}\mathbf{x}^w + \mathbf{t} \tag{4.1}$$

$$\mathbf{x}^c = \mathbf{R}(\mathbf{x}^w - \mathbf{c}) \tag{4.2}$$

Below is formulated the Extended Kalman Filter state vector on Eq. 4.3 and dynamic model on Eq. 4.4 where the camera is considered smooth and with constant velocity. Furthermore, both accelerations are modeled with zero-mean Gaussian noise $\omega = [a,\alpha]^T$.

$$\mathbf{s}_{13\times1} = [\mathbf{c}^T_{3\times1}, \mathbf{q}^T_{4\times1}, \mathbf{v}^T_{3\times1}, \omega^T_{3\times1}] \tag{4.3}$$

$$\mathbf{f}_{13\times1}(\mathbf{s},n) = \begin{bmatrix} \mathbf{c}_k + (\mathbf{v}_k + v_k)\Delta t \\ \mathbf{q}_k * \mathbf{q}((\omega_k + \Omega_k)\Delta t) \\ \mathbf{v}_k + v_k \\ \omega_k + \Omega_k \end{bmatrix} \tag{4.4}$$

- **c** - camera center
- **q** - camera orientation
- **v** - linear velocity
- v - velocity caused by linear acceleration ($v = a\Delta t$)

- ω - angular velocity
- Ω - velocity caused by angular acceleration ($\Omega = \alpha \Delta t$)

The state prediction covariance depends, as seen in Eq. 3.41, on the Jacobian of the non-linear dynamic model. Therefore, **F** is the derivative of the dynamic model relative to **s** and **G** is the derivative of the dynamic model relative to ω and are defined respectively in Eq. 4.5 and Eq.4.6. Information about quaternion derivatives can be found in [13].

$$\mathbf{F}_{13\times13} = \begin{bmatrix} I_3 & 0 & \Delta t I_3 & 0 \\ 0 & \frac{\delta \mathbf{q}_{k+1}}{\delta \mathbf{q}_k} & 0 & \frac{\mathbf{q}_{k+1}}{\delta \omega_k} \\ 0 & 0 & I_3 & 0 \\ 0 & 0 & 0 & I_3 \end{bmatrix} \tag{4.5}$$

$$\mathbf{G}_{13\times6} = \begin{bmatrix} \Delta t I_3 & 0 \\ 0 & \frac{\delta \mathbf{q}_{k+1}}{\delta \mathbf{q}_k} \\ I_3 & 0 \\ 0 & I_3 \end{bmatrix} \tag{4.6}$$

The observation model is made up of 2D points in the image plane which are projections of the real world 3D reference points. To put it another way, they are the outcome of the projection function mapping 3D points into the image plane. Therefore, the n 2D world coordinate system points are provided by the observation vector in Eq. 4.7. To do this, the n 3D reference points are converted from world to the camera coordinate system in quaternion form Eq. 4.8 and subsequently the conversion to the image plane is performed following Eq. 4.9. Therefore, the observation model is defined in Eq. 4.10.

$$\mathbf{z}_{2n\times1} = \begin{bmatrix} x_i^1 & x_j^1 & \dots & x_i^n & x_j^n \end{bmatrix}^T, \tag{4.7}$$

$$h^{WC}(\mathbf{s},\mathbf{x}^w) = \begin{bmatrix} 0 \\ \mathbf{x}^c \end{bmatrix}^T = \mathbf{q} * \begin{bmatrix} 0 & \\ \mathbf{x}^w & -\mathbf{c} \end{bmatrix}^T * \mathbf{q}^*, \tag{4.8}$$

$$h^{CI}(\mathbf{x}^c) = \mathbf{x}^I = \begin{bmatrix} f\frac{x_i^c}{x_k^c} & f\frac{x_j^c}{x_k^c} \end{bmatrix} \tag{4.9}$$

$$h_{2n\times1} = h^{CI}(h^{WC}(\mathbf{x}^w)) \tag{4.10}$$

The uncertainty of the tracked 2D points in the image sequence is represented by the observation covariance matrix **U**, which is diagonal since the feature tracking errors are expected to be independent and identically distributed with a zero-mean Gaussian distribution. Finally, the Kalman

Gain and the posterior state covariance matrix require the Jacobian of the observation model with respect to the state vector **s**:

$$H_{2n\times 13} = \frac{\delta h}{\delta \mathbf{s}} = \frac{\delta h^{CI}}{\delta h^{WC}} \frac{\delta h^{WC}}{\delta \mathbf{s}} \tag{4.11}$$

This method substitutes the previous methods seen in 3.2.8, but not entirely as because the filter work by approximation, a typical initial estimation when the filter is created is essential.
In particular, the RANSAC algorithm applied to the PnP problem was tested, and a significant reduction of the errors influenced by outliers was visually apparent. However, a change in the pose estimation was perceptible from frame to frame as the inliers correspondences used were constantly changing, leading to a non-smooth experience. Furthermore, the traditional methods are naive because they estimate based on the information they are provided. On the other hand, EKPnP has the advantage of knowing the information about the temporal dependencies of camera pose and uncertainty of features.

In the original paper, the authors provide quantitative information against multiple other methods, therefore proving their claim to improve the pose's estimation.

Improving The Naive Mapping Process

As discussed previously, marker corner detection is noisy and unreliable. Therefore it cannot be trusted because it leads to an erroneous mapping process of the markers. To diminish this problem, we can implement a method that calculates a marker's reliability by creating a unit of uncertainty. This uncertainty is non-negative and inverse to the algorithm trust in the marker. Meaning that, as the uncertainty is closer to 0, the more the marker can be trusted. On the other hand, the higher the uncertainty, the lower the trust.

Marker Uncertainty - Naive

This was the first implementation as a starting point. It was later improved using EKFPnP.
As the markers pose are estimated in the mapping process, the algorithm ignores the markers that are not stable in consecutive frames and those with an angle relative to the camera plane higher than 45º (variable by the user, he/she must understand that the more perpendicular the angle, the less prone to errors the estimations). When a marker is detected and not visible in the previous frame, its translation and rotation (in quaternion representation) are stored. The algorithm jumps the check for stability for the current iteration, setting the counter to 0. In other words, the algorithm now knows that the marker with this specific ID exists and can proceed to the mapping process if the marker is stable in the following frames. For each identified marker stability check, the procedure is divided into two steps. The first step checks the translation stability by setting a sphere around the stored marker translation. Suppose the translation estimation in the subsequent frame is inside this sphere. In that case, the counter can be updated. If not, the marker would immediately be ignored as if it was not visible. If the translation check is passed successfully, the algorithm will now check for the stability of the rotation. A relative quaternion between the stored quaternion and

the current estimate is calculated and represents the change in rotation between the two. Analysing the real part of the quaternion can verify if the rotation was minimal because, in that case, its value is as close to 1 as possible.
Once these requirements are met, the uncertainty of the markers is then quantified, based on the distance, size and orientation estimated according to the Eq. 4.12, where "angle" is the angle between the camera and marker plane.

$$uncertainty = \arccos angle \times \frac{VisualisedArea}{Distance} \tag{4.12}$$

Marker Uncertainty - Extended Kalman Filter

Previously, an Extended Kalman Filter was verified to be extremely valuable for improving the camera pose estimation as it has information of the past, can predict future poses and filter erroneous data inputs. Besides the improvement in the pose estimation, we can also extract how much it trusts each marker by extracting each filter prediction/correction uncertainty and applying it as the uncertainty of the markers in the mapping process. This value can be extracted from the matrix P of Eq. 3.43. Its diagonal represents the uncertainty of the tracked 2D points.
With this knowledge, by running an Extended Kalman Filter at each detected marker, there is a considerable reduction of the impact of erroneous detections and high stabilisation of the individual estimations over time. Furthermore, as the filter is predictive, it can still use the information of lost markers for a few frames (until their uncertainty becomes too high as no observations feed the filter to reduce it). For the same reason, in-between frames, we can use the prediction of the filter in order to create more data, which means that the visualisation can have a much higher frame rate than the camera provides to the algorithm and providing a continuous and smoother experience.
We can now better estimate the uncertainty of the markers individually. However, to create a map of markers, it is necessary to calculate uncertainty dependencies, which can be obtained by the product of the individual uncertainty of the two markers in analysis. If multiple markers are detected, the dependencies of all the detected markers are calculated to every other detected one, except for the markers where the marker itself is somehow parent of the second or both are not included previously in the map. Once the dependencies are calculated, the lowest dependency available for every marker proceeds to the next step, where, in the case of the first marker is not included previously, it is added relative to the best match possible (lowest uncertainty). If it is included, the uncertainty is compared with the previous one, and if lower, the map is updated.

Fig. 4.6, shows two markers in the frame with the ID's 249 and 250 as an example of their uncertainties in a specific frame (middle value), where both of their poses are being estimated individually by EKFPnP. Once estimated, a square is shown in the virtual environment, creating an augmented reality squared texture which is updated every frame displaying the ID and uncertainty of the estimation on top of each marker.
Assume that both markers are below the maximum uncertainty threshold and have been seen for enough frames for the uncertainty to stabilize. If we assume that the origin is the marker with ID 250 with the current uncertainty of 1.33 and already mapped, the logical thing to happen is map-

ping the marker with ID 249 and uncertainty 1.41 relative to the marker with ID 250. If we follow this logic, in the current frame, the marker with ID 249 can be mapped to 250 as they are both stable and under the maximum uncertainty threshold. The algorithm will then add marker 249 to the map, and its uncertainty will be the product of the previous two ($1.33 \times 1.41 = 1.8753$).

Assume three markers on the scene. "A" and "B" were previously mapped, "C" was not. On the current frame, the relative pose of "C" is calculated relative to both "A" and "B", but it would only be mapped relative to the marker where the relative uncertainty is the lowest.
Assume now a different scenario, and all three markers are already mapped. As "C" is already on the map, the algorithm would only estimate the relative uncertainties with other mapped markers, and the algorithm would try to update it if at least one of the new uncertainties calculated is lower than its uncertainty when it was mapped. The algorithm repeats this step to all markers trying to reduce the overall uncertainty of the map, keeping in mind that possible dependencies of markers may interfere with one another if both are updated in the current frame. Therefore, there are possible updates that may be skipped.

4.4 Final Map Structure

The map is composed of a list of a tree-like nodes, where each node corresponds to one marker. Each node stores:

- Self ID;
- Parent ID;
- Own uncertainty;
- Pose relative to the father;
- pose relative to the map origin - (useful to reduces the number of calculations when building the 3D point cloud),

4.5 Visualization - 3D point cloud

In each frame, the ArUco detector returns all detected markers ID and the position of the corners of the markers in the image plane. Once completed, a new list of 2D to 3D correspondences is created empty, and the algorithms search on the tree-like structure if the detected markers exist already in the map.
For each detected marker, if it exists in the data structure, four 3D-2D correspondences are added to the two correspondences list (four corners of the marker) based on the pose of the marker relative to the origin. Once all the detected markers are tested, these two lists carry on to the PnP solver unless there are lower than four correspondences. If that is the case, the algorithm skips to the next iteration. If the algorithm did not skip to the next iteration, they pass through the RANSAC PnP solver to filter the outliers before passing the correspondences to the EKFPnP. If there are not

enough inliers, the algorithm skips to the next iteration similarly to before and does not enter the EKFPnP step. If it can enter EKFPnP and an instance of it exists, the correspondences are passed to update the filter. If, on the other hand, it does not exist, a new instance is created. There is also the case that a filter instance exists, but the current frame does not have enough correspondences to run the filter update method, either because of RANSAC deleting much of the correspondences or simply by the markers not being detected. If this is the case, the filter only runs the prediction method instead of the regular update (prediction and correction), increasing its uncertainty to be tested to keep it under a maximum threshold. If the threshold is surpassed, the instance of the current filter is deleted, commanding the creation of a new filter when possible.

4.6 Tests and Results

This section presents experiments realised with the above methodology to provide information about the mapping process using solely squared fiducial markers.

4.6.1 Experiment Setup:

The markers were freely placed in the real world (non-rigid configuration), meaning that the user only has to be accountable for having the markers spaced so that, ideally, at least two markers can be seen per frame of the video feed and know each other marker size. Besides this, even though not a requirement, it is recommended to provide all the ID and sizes a priori to ensure they are correct by creating a JSON file using a simple side application, also developed in this work, providing ID and sizes of all the used markers. If an ID is not present in the file, a default marker size is used. Of interest is the analysis of the individual markers uncertainty and how the relationship between markers evolves over the period a video feed is used as the mapping algorithm input.
The videos were recorded after the marker placement using a calibrated ”Xiaomi Poco X3 Pro” smartphone camera. Even though it has a CMOS camera sensor, it offers decent video stabilisation and a reduced Rolling Effect, taken at 1080p@30 for real-time mode and 1080p@60 when in offline mode to acquire more information about the scene to the mapping process.

Firstly, it is important to visualise a reduction in the uncertainty of each marker over time as the EKFPnP approximates the correct value. Once it is close to the actual value, it is prone to be stable, and the estimation of the marker pose to be at its peak in accuracy and error rejection as the predictive part of the filter is weighing the most compared to the observation.

Secondly, it is interesting to analyse how the relationship between markers evolves using the previously mentioned individual uncertainties to estimate it. As the individual uncertainties get lower and lower, the goal is to decrease the related uncertainties and, therefore, the error of such relations.

Thirdly, a complete map is visualised, including the positions of the markers and their relationship uncertainties.

Fourthly, the previously map created is used in a visualisation algorithm where all the corners of the detected markers are provided to an EKFPnP where it is proven that occluding some markers

does not affect the visualisation. It may only reduce its robustness.

4.6.2 Results:

The individual uncertainties of the markers associated with the ID 249 and 250 were taken from a video sequence containing both Fig. 4.5 and Fig. 4.6 frames. The markers in the figures have three values on top of each representing the ID, individual uncertainty and the number of frames the markers are continuously being seen while below the maximum threshold (requirement to be classified as stable and able to enter the mapping process, in this case, the configurations were the maximum threshold of 2.0 and minimum number on continuous frames of 150).

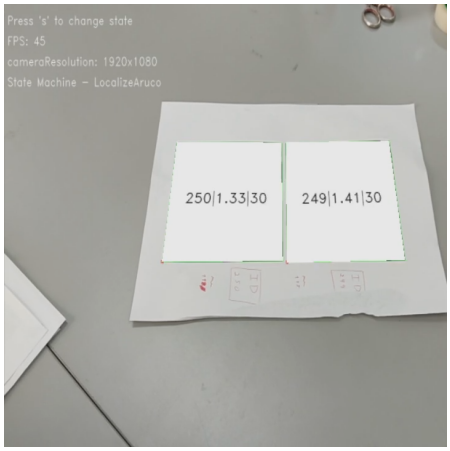

Figure 4.5: Individual Uncertainty 1

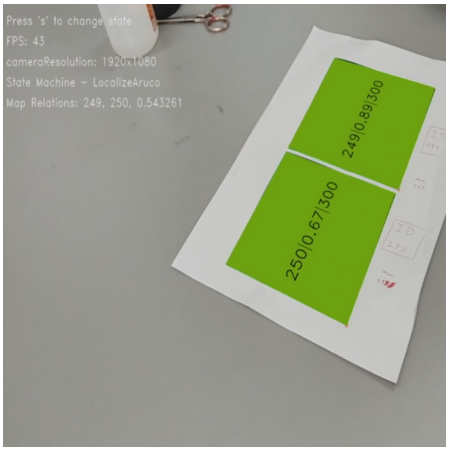

Figure 4.6: Individual Uncertainty 2

As expected, because the environment and camera did not change by much (the camera had a similar angle relative to the markers throughout the video, no change in lighting, ...), the uncertainties of the individual markers saw a decrease and stabilisation over time. The initial spike on both cases is expected as the filters initialise with zero uncertainty, giving the observations absolute weight compared with the filter predictions. As the predictions get better and better, the filter can provide more and more weight to them. Therefore the uncertainties improve over time, eventually stabilising. This can be observed in the graph of Fig. 4.7.

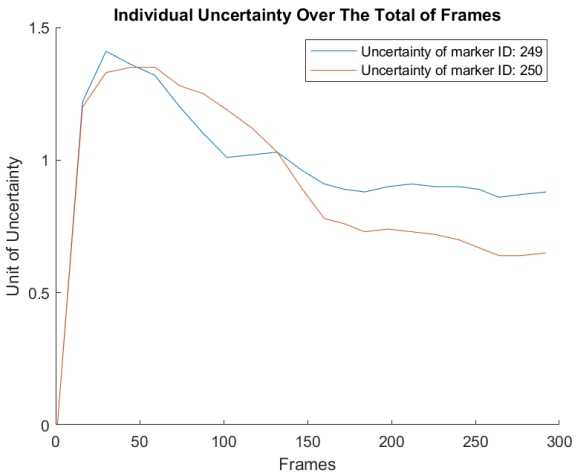

Figure 4.7: Individual Uncertainties Over The Total of Frames

Once the markers are considered stable, meaning that they are below the maximum uncertainty threshold for a continuous amount of frames, the mapping algorithm looks for markers in the same frame that can be related using the pose and uncertainties provided by each EKFPnP. The next set of data is provided by a new video sequence with four markers in the same plane to more easily know where each marker is in the real world and provide quantitative information to the reader as represented in Fig. 4.8 with respective IDs shown in Fig. 4.9 and final map in Fig. 4.10. The sizes of the markers with the IDs 247, 248, 249, and 250 are, respectively, 130mm, 129mm, 118mm, and 118mm.

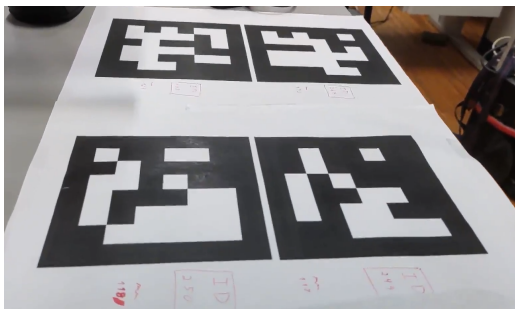

Figure 4.8: Four Markers Test

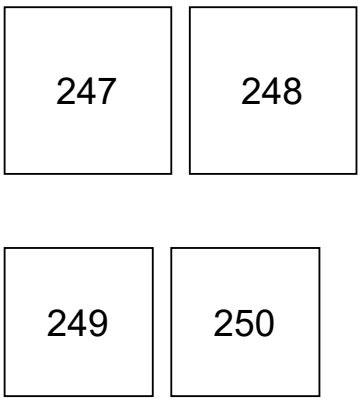

Figure 4.9: Marker IDs

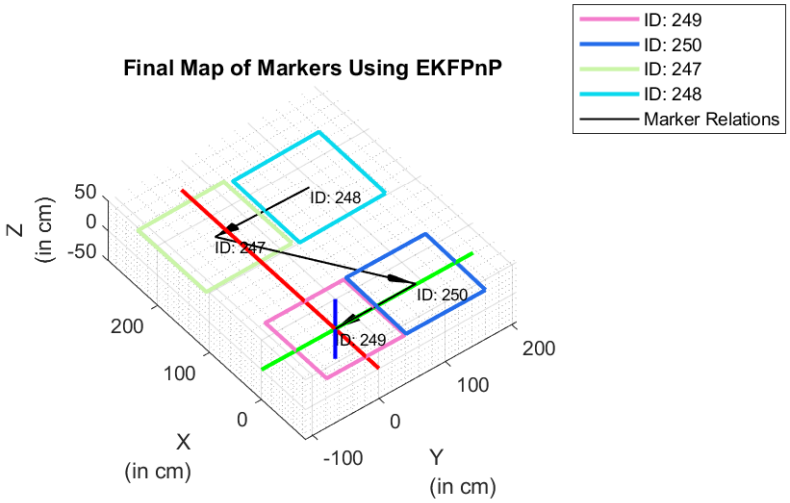

Figure 4.10: Final Map of Markers Using EKFPnP

In this scenario, the uncertainty of the relations varies throughout the video, as shown in Fig. 4.11, where it is possible to visualise the frame's index when a relationship was first created and its evolution over the subsequent frames.

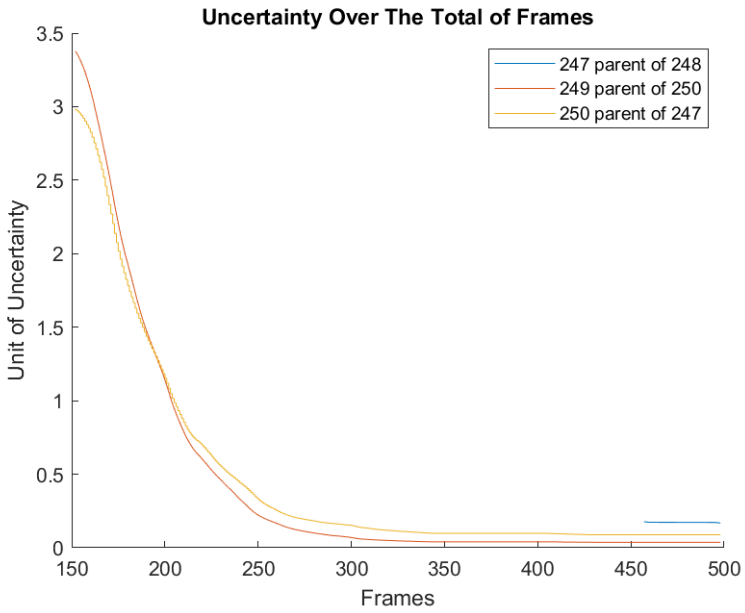

Figure 4.11: Uncertainty Over The Total of Frames

Continuing the analysis of the mapped markers, the angle between the markers is provided in Fig. 4.12, where, as they are in the same plane, as expected, the angle is consistently close to zero, and finally in Fig. 4.13 it is shown the distance error between the estimated relations and the ground

truth measured.

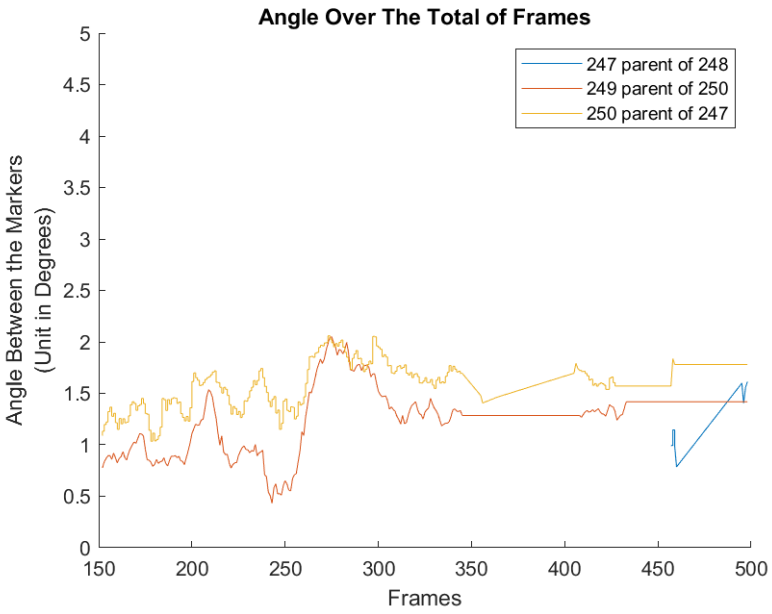

Figure 4.12: Angle Over The Total of Frames

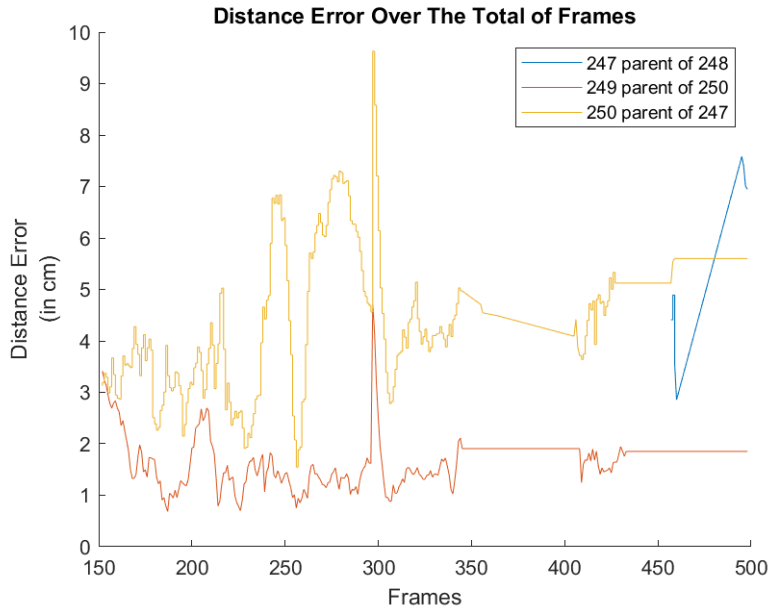

Figure 4.13: Distance Error Over The Total of Frames

To conclude this section, a new map of the markers in Fig. 4.14 was created, where it is possible to play a game of chess in Augmented Reality also created in the development of this project, which can be visualised in Fig. 4.15. As expected, the user Augmented Reality experience was stable, robust and smooth.

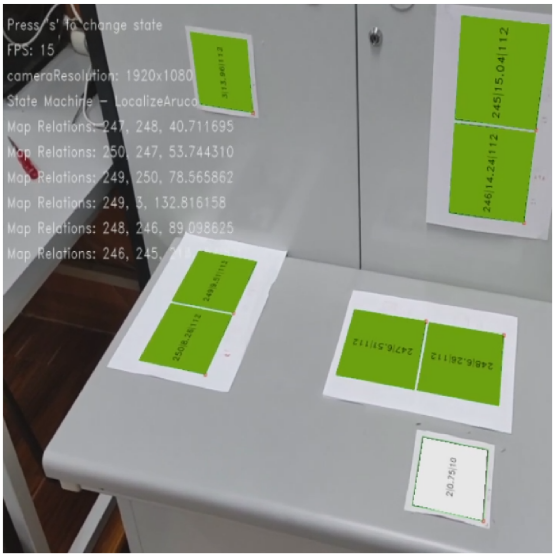

Figure 4.14: Chess Game - Markers

Figure 4.15: Chess Game - Visualisation

5

UcoSLAM for Augmented Reality Applications

In this chapter, a second methodology integrating UcoSLAM [39] to our project was developed. This method was integrated to localise the camera in the real world to be able to use it in Augmented Reality applications, but now with the aid of both marker-based and structural keypoints.

As UcoSLAM is still a work in progress, multiple topics may suffer possible changes. However, the project's core is already publicly available for academic use. Following this idea, because the documentation of UcoSLAM mentions that multiple sizes of markers could eventually be applied, much time was put into developing the algorithm to achieve this goal. However, without success, as it is not yet implemented and multiple essential files of the library are encrypted, this change was not possible in the time frame of this project. The problems found while expanding UcoSLAM led to communication via e-mail to the authors. In response, they confirmed that the project was not finished and that multiple marker sizes were not yet implemented. Furthermore, multiple attempts to upgrade the work were made, but sections of the library are encrypted, not allowing for necessary changes. Even though this is not the ideal scenario, it is still an outstanding work and is believed to be a solid choice to implement in a wide variety of Augmented Reality applications in the industry, including this project, if the user considers the marker size limitations.

5.1 Important limitations to have in mind

Given the ability to save and load the maps, it is possible to do the mapping process only once, following the localisation standalone. The mapping process in industrial scenarios must be as precise and deterministic as possible. Even though UcoSLAM can run in parallel mode, the authors suggest running the mapping process sequentially when accuracy and precision are critical. Even though it takes more time to finish, the result is more precise as there is a possibility of skipping frames, which can occur in the mapping process while using multithreading mode, where a thread assigned to the map manager workload did not process the current frame iteration when another thread has already finished the subsequent frames feature extraction. Fig. 5.1 helps to visualise this phenomenon.

UcoSLAM can simultaneously map and localise the camera in the world. However, considering

this application as critical, it is advised to record a long video sequence of all the areas to be mapped with an external camera (e.g., smartphone) and then run the mapping process sourcing the video/videos. Furthermore, multiple video sequences can be used in the mapping process. This way, even if it takes longer, it is possible to run the program offline and get a much better result with a lower error. Once the map is created and saved, it can be safely used in camera localisation standalone taking advantage of multiple threads. Taking this advantage, a very useful speedup is obtained, and there are no risks of damage of skipping frames, as the program does not run the heavy tasks such as global map optimisations and loop closure verification. If map expansion is required, UcoSLAM allows to load of the previous map, and once the camera is well localised, the mapping process can continue.

As the project is still in progress, not all the features are yet applied or completed. In this case, it means that we can only insert a single value to all the marker's sizes. This is an important limitation to our project because, as we discussed previously, industrial applications can take advantage of multiple marker sizes, depending on the user's location. As said before, in the user workspace, it is essential to have multiple small markers to improve the robustness and larger ones on corridors. For example, in the user workspace (e.g., Desk), it may not be possible to insert significant sized markers and, on the other hand, on huge surfaces (e.g., corridors), small markers represent can be hard to detect correctly, which introduces error, leading to both wrong mappings and erroneous pose estimations.

5.2 UcoSLAM architecture

In the Fig. 5.1, a visualization of the UcoSLAM pipeline is shown, followed by a overview of the system. However it does absolutely not replace the read of the original paper in order to understand the equations and methodologies used.

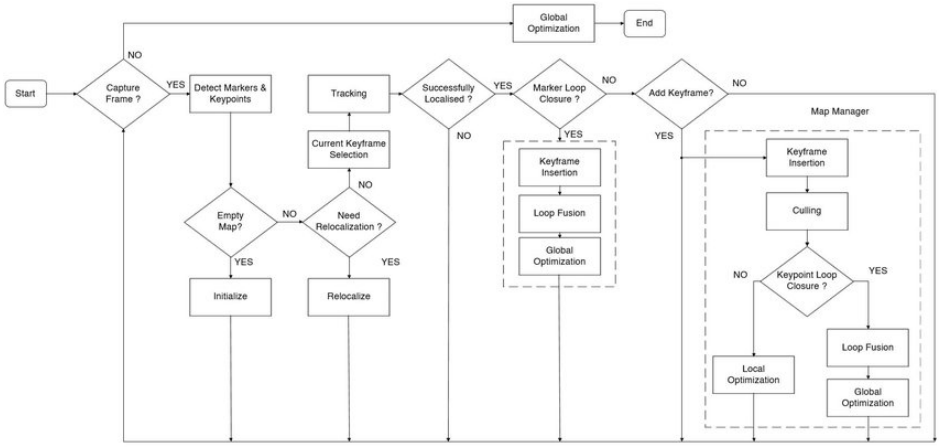

Figure 5.1: UcoSLAM architecture (from [39])

The following five sets of information are used to create the environment map:

- **Set of keyframes**

  This set stores the minimal ammount of required frames for the keypoints to be referenced to. This set is contained in the set of frames, where each frame stores the time of capture, pose that transforms the points from the global reference system to the camera reference system, and the intrinsic parameters of the camera used.

- **Set of Map Points**

  This set stores all the points that must store its 3D position based on the triangulation of the same keypoints on multiple keyframes and the viewing direction (average of all the key points that share the point in analysis) and its descriptor.
  The keypoints, detected through an image pyramid technique, must have their own set, which stores the level of the image it was taken from, the 2D pixel coordinates upsampled to the first level of the pyramid (level 0), and a descriptor.

- **Set of fiducial markers**

  The set of markers can easily be described by the marker's scale, pose, and corners default position on the plane Z=0.

- **Set of connection graphs**

  A connection graph is defined by the weight of each keyframe with his neighbors. It is based on the number of points and markers shared between the keyframes.

- **Set of keyframe features**

  For relocalization purposes, if no markers are detected, the system uses a keyframe recognition method based on a BoW with the keyframes' features in the map taking advantage of vectorial operations.

5.3 Ucoslam Pipeline

Assuming the intrinsic parameters are estimated before the use of the system, the combined use of keypoints and ArUco markers for tracking and relocalization is the primary distinction in the UcoSLAM method compared with typical visual SLAM alternatives.
Every time a new frame is presented to the system, the UcoSLAM system creates and updates a map of the environment. It employs homography, a fundamental matrix (using keypoints), and one or more markers for map initialization. The system begins tracking or relocalization when the map has been initialized. When the system identifies the camera pose in the previous frame, it attempts to estimate the current location using the previous frame as a starting point. The tracking in the system is based on keypoint and marker corner reprojection errors. Before tracking, a reference frame is chosen as the map keyframe, including frequent matches to the frame examined in the previous instant. The system looks for loop closures produced by ArUco markers once

the tracking procedure is complete. The system proceeds to keyframe insertion, loop fusion, and global optimization if a loop closure is spotted. If the loop closure is not spotted, the system falls back on the map manager block, which initiates the culling process whenever a new keyframe is entered.
By eliminating duplicate information from the map, the culling process helps to keep the map size reasonable by checking for keypoint loop closure after the culling process. The system conducts local optimization to incorporate the new information if the keypoint loop closure is not recognized. If the system detects the keypoint loop closure, it proceeds to the loop fusion and global optimization phases. If the system's tracking procedure failed in the previous frame, the system switches to relocalization mode. It verifies the markers already recorded on the map in the relocalization mode. The bag-of-words (BoW) technique is used if the relocalization mode cannot recognize the known markers.

5.4 Transfer of Estimated Poses Between Programs

The process of incorporating UcoSLAM in our project would trigger inevitable slowdowns in this project. To face such a problem and rapidly have a working prototype, every new camera poses estimated by UcoSLAM is sent via UDP connection to our main program, responsible for displaying the virtual elements to the user. The UDP connection is, for now, the bridge connecting both programs allowing the merge in order to create the Augmented Reality experience we are trying to provide to the user. This connection avoids time lost in dependencies conflicts and redoing much of the work already implemented. Furthermore, this approach allows the frame to be taken and sent wirelessly from the camera to a powerful computer running UcoSLAM, which sends the poses back to the display. Once the main program receives the pose estimated by UcoSLAM, the camera's pose will be filtered by a Constant Acceleration Model of a Linear Kalman Filter, based on [51], to obtain a more stable and consistent estimation over time and estimate the poses in between transfers. Even though the Extended Kalman Filter implemented in [36] is the final objective because of the non-linear rotations, it requires a modifications in the approach. The overview of this architecture can be found in the Fig. 5.2

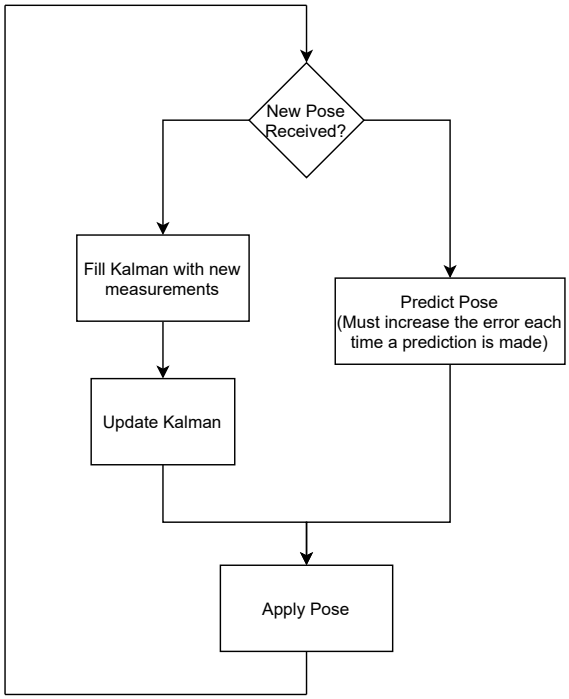

Figure 5.2: LKF Architecture applied after the pose estimation

5.4.1 Immediate Improvements in this Application

1. **Stability and smoothness of the camera pose**: As discussed previously, by predicting the next frame pose and correcting it based on the observations, we nearly eliminate the tremble in the camera pose over time and minimize single erroneous estimations.

2. **Frame rate**: If the Augmented Reality application frame rate is linear to the number of poses estimated by UcoSLAM, it would have a continuously variable frame rate and the possibility of falling to small values because the pose estimation from UcoSLAM tested provided roughly 15 to 45 pose estimations per second.
   Because an ideal Augmented Reality experience for the user requires high and very stable frame rates, the filter provides a virtually invariable frame rate experience as long as a decent estimation rate is provided by predicting the camera's pose in between transmissions of poses from the UcoSLAM to our program.

3. **Possibility of including multiple sensors:** such as IMU to further improve the pose estimation in the UcoSLAM project or provide better predictions in-between pose transfers (as future work).

5.4.2 Linear Kalman Filter Design

The implementation of the Kalman Filter from OpenCV [6] is integrated in this project, it takes care of behind-the-scenes calculations discussed in Section 3.4, which follows the general diagram of a linear kalman filter as shown in fig 5.3.

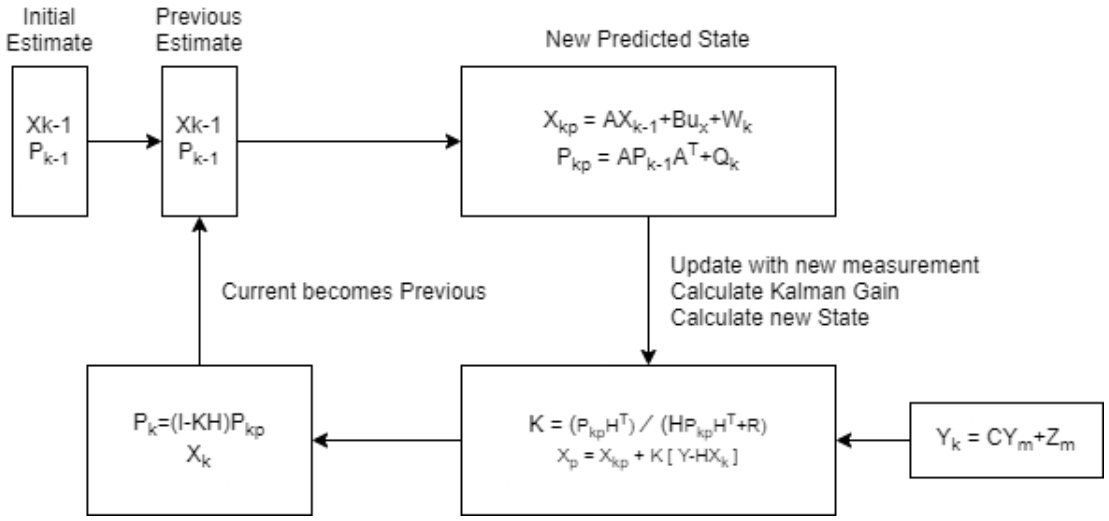

Figure 5.3: Kalman Diagram

Kalman State Matrix:
One of the main objectives of the referenced work, is the filter both translation and rotation of the camera pose. Thus, assuming a constant acceleration filter, it requires the translation, rotation, and the first and second derivatives of both. Each of the mentioned requirements can be represented as a set of three variables. Accordingly, there are six sets of three variables, each representing the position, velocity, acceleration, Euler rotation (roll, pitch, and yaw), angular velocity, and angular acceleration. Merging it, the Eq. 5.1 arises for a system with the following state vector of 18 states, represented in the state matrix Eq. 5.2.

$$X = (x,y,z,\dot{x},\dot{y},\dot{z},\ddot{x},\ddot{y},\ddot{z},\psi,\theta,\phi,\dot{\psi},\dot{\theta},\dot{\phi},\ddot{\psi},\ddot{\theta},\ddot{\phi}) \tag{5.1}$$

$$A = \begin{bmatrix} A_1 & 0_{3\times3} & \cdots & \cdots & \cdots & 0_{3\times3} \\ 0_{3\times3} & A_2 & \ddots & & & \vdots \\ \vdots & \ddots & I_{3*3} & \ddots & & \vdots \\ \vdots & & \ddots & A_1 & \ddots & \vdots \\ \vdots & & & \ddots & A_2 & 0_{3\times3} \\ 0_{3\times3} & \cdots & \cdots & \cdots & 0_{3\times3} & I_{3*3} \end{bmatrix}_{(18\times18)} \tag{5.2}$$

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 & dt^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 & 0 & dt^2 & 0 \\ 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 & dt^2 \end{bmatrix}_{3*9} \tag{5.3}$$

$$A_2 = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \end{bmatrix}_{3*6} \tag{5.4}$$

The B matrix will be ignored for now as no known control variables exist. The delta time here could be provided in it, but this simple implementation works well in our results. It is possible to write a simplified version of the current state as:

$$X_{kp} = AX_{k-1} + W_k \tag{5.5}$$

Kalman Measurement Matrix: Extrapolating the information from the state vector, the measurement matrix **Y**, to account for the position (**x**,**y**,**z**) and rotation (ψ,δ,ϕ), must be conceived as follows.

$$Y_k = \begin{bmatrix} 1 & 0 & 0 & 0_{3\times6} & 0 & 0 & 0 & 0_{3\times6} \\ 0 & 1 & 0 & 0_{3\times6} & 0 & 0 & 0 & 0_{3\times6} \\ 0 & 0 & 1 & 0_{3\times6} & 0 & 0 & 0 & 0_{3\times6} \\ 0 & 0 & 0 & 0_{3\times6} & 1 & 0 & 0 & 0_{3\times6} \\ 0 & 0 & 0 & 0_{3\times6} & 0 & 1 & 0 & 0_{3\times6} \\ 0 & 0 & 0 & 0_{3\times6} & 0 & 0 & 1 & 0_{3\times6} \end{bmatrix}_{3*18} \tag{5.6}$$

Kalman Noise Covariances: The noises for the process and measurements are fixed and equal to the default OpenCV values because, based on our observation, they work well for our application. Based on how much we trust each sensor observation on each frame, we could vary their value, but the slight quality improvement would require additional time and would not be worth it.

Initial Process Covariance Matrix The initial estimate is not particularly important because the Linear Kalman filter has the advantage of quickly converge to the correct value. Because of this, the default OpenCV values are used (Identity Matrix). In other words, the Kalman Gain, which varies from 0 to 1, will be very high in the initial iteration of the filter. This high value represents low stability in the estimates, making the Kalman Gain weight choose the measurements over the prediction as they are considered accurate. It is expected for the Kalman Gain to decrease very quickly, particularly in the initial estimations with observation.

5.5 Tests and Results

This section presents experiments realised with the above methodology to provide information about UcoSLAM monocular camera pose estimation.

5.5.1 Experiment Setup:

Similar to section 4.6.1, the same calibrated smartphone camera was used in these tests to record the videos. Here, the camera settings were 1080@60 in the offline mapping process and 1080@30 while running localisation standalone. In this case, the markers are required to have the same size, and their placement over the laboratory and surrounding corridors was done to help map complex or repetitive textured areas such as white walls and the corridors themselves. Furthermore, multiple were placed around the user workspace to aid the localisation algorithm and improve robustness.

There were two tests realised. The first test aims to map the user workspace for later localisation of the camera, which is sent via a connection to the Augmented Reality application that filters the received camera poses via a Linear Kalman Filter and displays the corresponding virtual elements. Of interest are the percentage of tracking over the video sequence, the median number of poses transfered, the frame rate perceived by the user, and the amount of time taken to map such environment. The second test aims to test the mapping over an extended area. Here, the markers were placed in the problematic areas of keypoints extraction and comparison around the laboratory and large loop closure was tested.

Take into consideration that quantitative information about UcoSLAM is already provided in their publication. Here, the focus is demonstrating the integration of this work in Augmented Reality.

5.5.2 Results:

First Test - Workspace

Multiple markers were placed around the workspace, and the origin axis of the created map was set to correspond to one of the markers. Multiple attempts to map the workspace taking advantage of multiple threads were made, but the results were inconsistent, leading multiple times to a worthless map. Therefore, the only attentions taken while running the mapping process were running it in sequential and offline modes, which took more time but provided a decent map of the environment as expected. In Fig. 5.4, a sequence of some frames taken from the mapping process is shown, followed by a brief discussion.

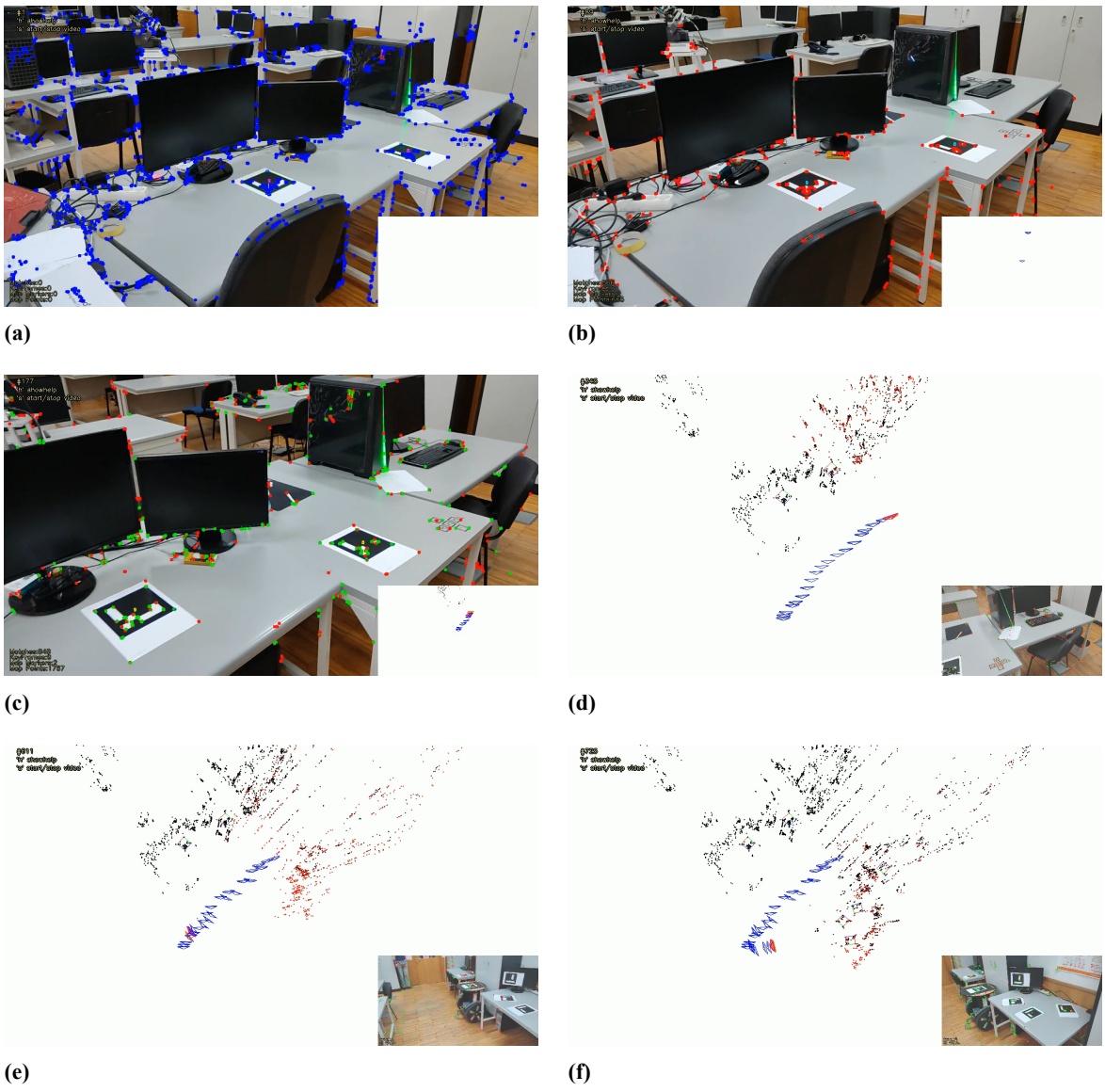

Figure 5.4: UcoSLAM Mapping Demo 1 - Mapping Process

In Fig. 5.4, it is possible to observe the detection of the structural keypoints in “a” (blue points),

tracks the ones that exist among continuous frames in “b” (red points), some of which, if consistent over time, are part of the green keypoints in “c”, meaning they are stored in the map. In the meantime, it also found and mapped some markers, which enabled the application of the map scale. These three figures represent the left side of the area to be mapped.
In “d”, “e” and “f”, it is possible to see how the map grows over time as the right side of the area is mapped, and the choice between keeping or not the temporal keypoints represented in red is made. The blue objects in these last three figures are the resulting keyframes, the orange object is the current camera pose estimation related to the red keyframe, and is the one that the current keypoints detected are being related to until the coming culling process. In Fig. 5.5, the keypoints are turned off visually to provide the reader with better visualisation of the markers already mapped in the mentioned process.

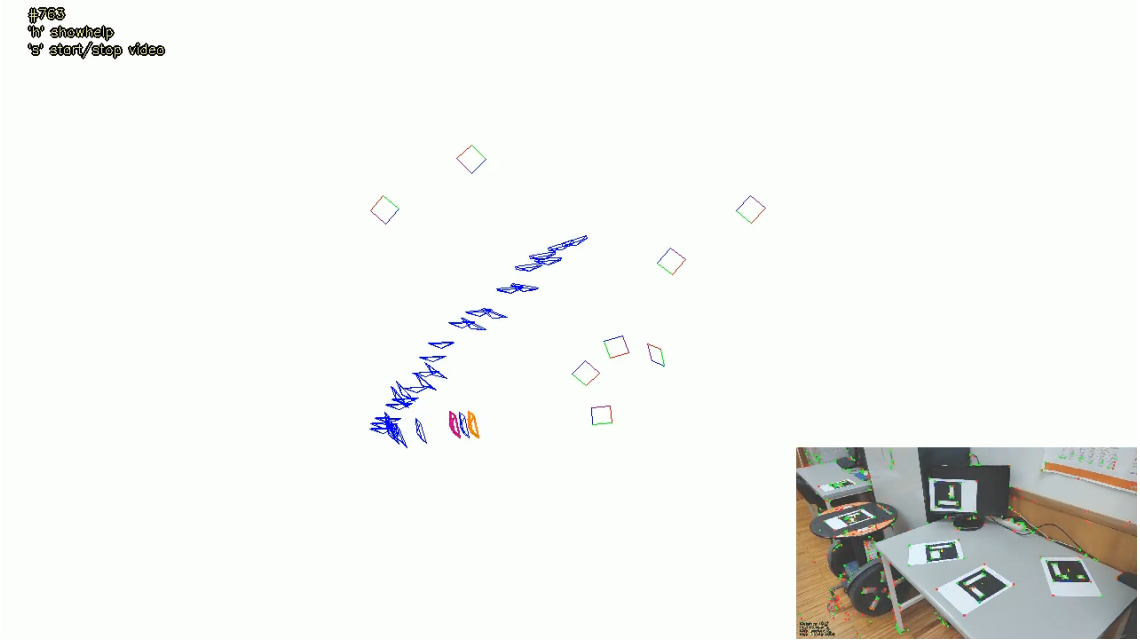

Figure 5.5: UcoSLAM Mapping Demo 1 - Markers Only

In table 5.1, it is possible to understand how UcoSLAM estimations vary a lot, being uncomfortable if used directly to estimate the virtual object pose, mainly when its performance is low. The Linear Kalman Filter estimates in-between pose transfer to provide a much higher FPS to the user.

Results							
Local	% Tracking	Transfers			FPS		
		min	median	max	min	median	max
Workspace	100	17.76	21.03	34.59	362	415	1166

Table 5.1: UcoSLAM - Workspace Mapping

Note: The programs were running on the same machine concurrently. The machine has an Intel I5

7300HQ and an NVIDIA GTX 1050. Typical values of our application when running standalone go beyond the 700 FPS.

Finally, using the previous map, Fig. 5.6 shows some virtual elements in an Augmented Reality Application as proof of concept in its top half part, while showing the structural keypoints and markers of the particular frame the pose estimation happened. The top half of the figure has a black background because it corresponds to the transparent part of the displayed image on the Smart Glasses device, providing the superimposed imagery. The bottom half of the figure shows the features used to localise the camera.

Figure 5.6: UcoSLAM Visualisation Demo

Second Test - Loop Closure While Navigating Around The Laboratory

In this test, it is interesting to see how the algorithm succeeds in mapping a bigger area and how it handles a loop closure. In Fig. 5.7 it is possible to visualise the final map where the circled area represents the laboratory, where the mapping process took more than three hours. In Fig. 5.8, it is possible to visualise the map construction overtime where in "a" the blue circled area is the main entrance of the laboratory floor, in "b" the blue circled area is closest stairs and elevator retative to "a", in "c" the blue circled area is the middle outside area and in red the laboratory workspace of the previous demo. Finally, in "d", the blue circled area is where the loop closure was detected. Note how the areas circled in "c" were corrected, drifting to the right and down.

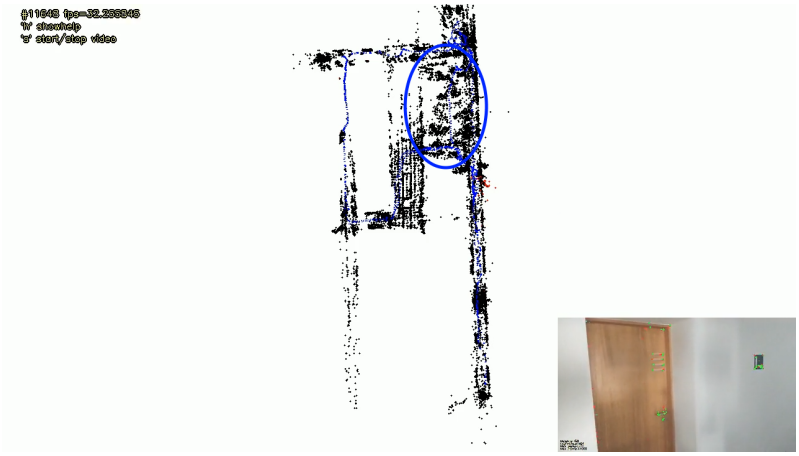

Figure 5.7: UcoSLAM Mapping Demo 2 - Final Map

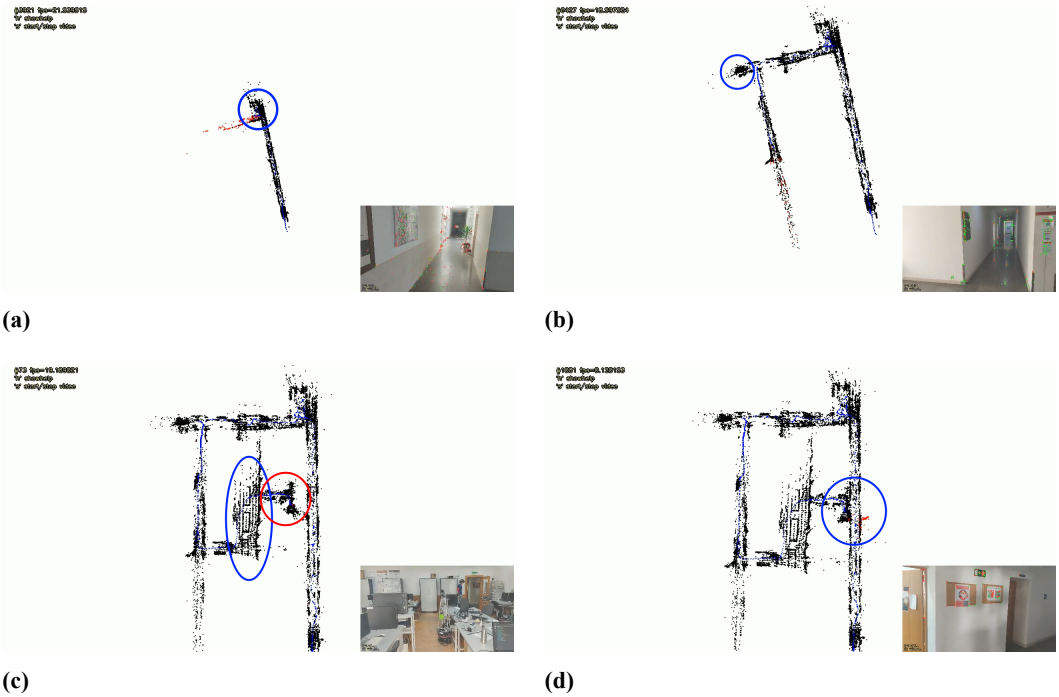

Figure 5.8: UcoSLAM Mapping Demo 2 - Mapping Process

6

Conclusion and Future Work

This work goal was to enable Augmented Reality in the industry, particularly in the assembly line of the mould industry. It requires the estimation in real-time of where the worker is looking to be able to superimpose the information with the real world.

Therefore, a robust way to estimate where the user is looking was developed using fiducial markers and Smart Glasses, which have a monocular camera system mounted on the user's forehead. The Smart Glasses enable the record of a user field of view to detect the markers placed freely on the environment while concurrently displaying the virtual elements. At the same time, the fiducial markers provide a way to ensure that keypoints exist while providing robustness, absolute pose and scale. Having the video feed of the user field of view, an algorithm that enables the simultaneous mapping and localisation of the camera was developed by relating the markers detected in the same frame, which might have different sizes depending on the area to be mapped. A supplemental application was built to enable an easy way to establish the sizes of the markers, which was a requirement as smaller ones are ideal for mapping the workspace, and bigger ones are better used for complex areas such as corridors. This relation was then used to create a tree-like data structure where each node is a marker, and the connections between a node and its parent represent a map relation.

As the relationship algorithm was being developed, the biggest problem of this project emerged, the ambiguity in the pose estimation of the markers, which was reduced with the implementation of EKFPnP but not entirely removed, particularly as they get farther from the camera. Once the filter implementation was finished, the uncertainty resulting from the marker poses substituted the naive implemented in the initial mapping algorithm, which hugely improved its robustness and reduced errors. The same EKFPnP was then used in the Augmented Reality visualisation process. However, in the first case, it was fed with only the four corners of each marker individually, and in this scenario, it is fed with all the detected marker corners to maximise the number of 2D-3D correspondences used in the estimation of the PnP problem. The finished product works appropriately, but future work is favoured to include more sensors to the EKFPnP, providing more data that can further reduce the errors and provide estimation even without detecting markers. The ambiguity problem mentioned above must be considered, as UcoSLAM authors also observed and even mentioned that they believe it is why markers-based maps were not developed in the past. The introduction of Speeded Up ArUco would also be great in this application, as it enables the

input of videos with higher resolution, which helps in the pose estimation problem as the solvers are fed with more precise data while detecting the markers very efficiently. The algorithm would also benefit immensely from the integration of loop closure, local and global optimisers.
To provide a real world scenario, a multiplayer chess game was developed, and it is possible to conclude that the virtual elements align very well with the real world and that the experience is very smooth and robust, even if the before mentioned suggestions are not yet implemented.

On the other hand, even though UcoSLAM does not fill all the required criteria in our list, such as having multiple sized markers, this work has proven to be very precise and reliable to map a larger area than the user workspace using both structural and artificial keypoints. Its implementation to our project was done where the video feed and pose estimation were sent wirelessly, except for the offline mapping process. As the pose is received in the Augmented Reality visualisation program by a process running in a separate thread with this specific task, the primary program filters the received poses in a Linear Kalman Filter smoothing the estimations and providing a much higher frame rate to the user by predicting in between transfers.
As the first implementation was concluded, multiple essential parts of UcoSLAM can be improved by our approach by integrating the multiple marker size solution and EKFPnP, which would not only improve the pose estimation but also enable multiple sensors to be used, as this project already offers multiple of the challenging suggestions provided to the first method.
The tests realised with the integration of this method provide a genuinely satisfactory result when mapping more expansive areas than the previous method. On the other hand, it is also much slower but has the functionality of offline mapping and, while in visualisation mode, the Kalman filter predictions compensate for it.

Finally, it is now encouraged for future work to be developed on top of this approach to develop a true Augmented Reality application and its test in a wider variety of use cases that were not tested, such as mobile robot localisation, video games, and drones.

Bibliography

[1] *Analisis De Eventos Deportivos Mediante Vision Artificial*. URL: https://drive.google.com/file/d/1CNRmlmaoT-_PZBlRO9ZpJfQuvus8Oknk/view (visited on 10/15/2021).

[2] *AVA*. URL: http://www.uco.es/investiga/grupos/ava/ (visited on 10/14/2021).

[3] Ronald T. Azuma. "A Survey of Augmented Reality". In: *Presence: Teleoperators and Virtual Environments* 6.4 (Aug. 1997), pp. 355–385. DOI: 10.1162/pres.1997.6.4.355. URL: https://doi.org/10.1162/pres.1997.6.4.355.

[4] Donna R. Berryman. "Augmented Reality: A Review". In: *Medical Reference Services Quarterly* 31.2 (2012). PMID: 22559183, pp. 212–218. DOI: 10.1080/02763869.2012.670604. eprint: https://doi.org/10.1080/02763869.2012.670604. URL: https://doi.org/10.1080/02763869.2012.670604.

[5] Bimber et al. *Spatial Augmented Reality Merging Real and Virtual Worlds*. Aug. 2005. ISBN: 9780429108501. DOI: 10.1201/b10624.

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

[7] Dean Brown. "Decentering distortion of lenses". In: 1966.

[8] Wilhelm Burger. *Zhang's Camera Calibration Algorithm: In-Depth Tutorial and Implementation*. May 2016. DOI: 10.13140/RG.2.1.1166.1688/1.

[9] Michael Calonder et al. "BRIEF: Binary Robust Independent Elementary Features". In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792. ISBN: 978-3-642-15561-1.

[10] *Camera intrinsics: Axis skew*. URL: https://blog.immenselyhappy.com/post/camera-axis-skew/ (visited on 10/31/2021).

[11] Julie Carmigniani et al. "Augmented reality technologies, systems and applications". In: *Multimedia tools and applications* 51.1 (2011), pp. 341–377.

[12] Thomas Caudell and David Mizell. "Augmented reality: An application of heads-up display technology to manual manufacturing processes". In: vol. 2. Feb. 1992, 659 –669 vol.2. ISBN: 0-8186-2420-5. DOI: 10.1109/HICSS.1992.183317.

[13] Javier Civera, Andrew J. Davison, and J. M. M. Montiel. "Structure from Motion using the Extended Kalman Filter". In: *Springer Tracts in Advanced Robotics*. 2012.

[14] A. E. Conrady. "Decentred Lens-Systems". In: *Monthly Notices of the Royal Astronomical Society* 79.5 (Mar. 1919), pp. 384–390. ISSN: 0035-8711. DOI: 10.1093/mnras/79.

5.384. eprint: https://academic.oup.com/mnras/article-pdf/79/5/384/18250798/mnras79-0384.pdf. URL: https://doi.org/10.1093/mnras/79.5.384.

[15] Mark Fiala. "Designing Highly Reliable Fiducial Markers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.7 (2010), pp. 1317–1324. DOI: 10.1109/TPAMI.2009.146.

[16] D. Flohr and J. Fischer. "A Lightweight ID-Based Extension for Marker Tracking Systems". In: *Eurographics Symposium on Virtual Environments (EGVE) Short Paper Proceedings*. Weimar, 2007, pp. 59–64.

[17] Dorian Galvez-López and Juan D. Tardos. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197. DOI: 10.1109/TRO.2012.2197158.

[18] S. Garrido-Jurado et al. "Generation of fiducial marker dictionaries using Mixed Integer Linear Programming". In: *Pattern Recognition* 51 (2016), pp. 481–491. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2015.09.023. URL: https://www.sciencedirect.com/science/article/pii/S0031320315003544.

[19] Sergio Garrido-Jurado et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion". In: *Pattern Recognit.* 47 (2014), pp. 2280–2292.

[20] *Geometric Camera Models and Calibration*. URL: https://slideplayer.com/slide/14241077/ (visited on 10/31/2021).

[21] M. Heilig. "Beginnings: Sensorama and the Telesphere Mask". In: 1998.

[22] M Heilig. "US Patent# 3,050,870". In: *Sensorama Simulator* (1962).

[23] S Karthika, P Praveena, and M GokilaMani. "Hololens". In: *International Journal of Computer Science and Mobile Computing* 6.2 (2017), pp. 41–50.

[24] H. Kato and M. Billinghurst. "Marker tracking and HMD calibration for a video-based augmented reality conferencing system". In: *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*. 1999, pp. 85–94. DOI: 10.1109/IWAR.1999.803809.

[25] Youngjoo Kim and Hyochoong Bang. "Introduction to Kalman Filter and Its Applications". In: Nov. 2018. ISBN: 978-1-83880-536-4. DOI: 10.5772/intechopen.80600.

[26] Georg Klein and David Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. 2007, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852.

[27] Dieter Koller et al. "Real-time Vision-Based Camera Tracking for Augmented Reality Applications". In: (July 1998). DOI: 10.1145/261135.261152.

[28] Kore. *Understanding the different types of AR devices*. 2018. URL: https://uxdesign.cc/augmented-reality-device-types-a7668b15bf7a (visited on 10/06/2021).

[29] Maxim Kuzmin. "Review. Classification and Comparison of the Existing SLAM Methods for Groups of Robots". In: *2018 22nd Conference of Open Innovations Association (FRUCT)*. 2018, pp. 115–120. DOI: 10.23919/FRUCT.2018.8468281.

[30] Heiner Lasi et al. "Industry 4.0". In: *Business Information Systems Engineering* 6 (Aug. 2014), pp. 239–242. DOI: 10.1007/s12599-014-0334-4.

[31] Vincent Lepetit and Pascal Fua. “Monocular Model-Based 3D Tracking of Rigid Objects: A Survey”. In: *Foundations and Trends in Computer Graphics and Vision* 1 (Jan. 2005). DOI: 10.1561/0600000001.

[32] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. “EPnP: An accurate O(n) solution to the PnP problem”. In: *International Journal of Computer Vision* 81 (Feb. 2009). DOI: 10.1007/s11263-008-0152-6.

[33] Hyon Lim and Young Sam Lee. “Real-time single camera SLAM using fiducial markers”. In: *2009 ICCAS-SICE*. 2009, pp. 177–182.

[34] D. Litwiller. “CCD vs. CMOS: facts and fiction”. In: *Photonics Spectra* 35 (Jan. 2001), pp. 154–158.

[35] David G. Lowe. “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: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[36] Mohammad Amin Mehralian and Mohsen Soryani. “EKFPnP: Extended Kalman Filter for Camera Pose Estimation in a Sequence of Images”. In: *CoRR* abs/1906.10324 (2019). arXiv: 1906.10324. URL: http://arxiv.org/abs/1906.10324.

[37] Paulo Menezes. “An Augmented Reality U-Academy Module: From Basic Principles to Connected Subjects”. In: *International Journal of Interactive Mobile Technologies (iJIM)* 11 (July 2017), p. 105. DOI: 10.3991/ijim.v11i5.7074.

[38] E. Mouragnon et al. “Real Time Localization and 3D Reconstruction”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 1. 2006, pp. 363–370. DOI: 10.1109/CVPR.2006.236.

[39] Rafael Muñoz-Salinas and Rafael Medina Carnicer. “UcoSLAM: Simultaneous Localization and Mapping by Fusion of KeyPoints and Squared Planar Markers”. In: *CoRR* abs/1902.03729 (2019). arXiv: 1902.03729. URL: http://arxiv.org/abs/1902.03729.

[40] Raul Mur-Artal and Juan D. Tardos. “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras”. In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262. ISSN: 1941-0468. DOI: 10.1109/tro.2017.2705103. URL: http://dx.doi.org/10.1109/TRO.2017.2705103.

[41] Raul Mur-Artal and Juan D. Tardós. “Fast relocalisation and loop closing in keyframe-based SLAM”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 846–853. DOI: 10.1109/ICRA.2014.6906953.

[42] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[43] Rafael Muñoz-Salinas, Manuel Marín-Jiménez, and Rafael Medina-Carnicer. “SPM-SLAM: Simultaneous Localization and Mapping with Squared Planar Markers”. In: *Pattern Recognition* 86 (Sept. 2018). DOI: 10.1016/j.patcog.2018.09.003.

[44] Rafael Muñoz-Salinas et al. *Mapping and Localization from Planar Markers*. 2017. arXiv: 1606.00151 [cs.CV].

[45] Michael Neunert, Michael Bloesch, and Jonas Buchli. “An open source, fiducial based, visual-inertial motion capture system”. In: *2016 19th International Conference on Information Fusion (FUSION)*. 2016, pp. 1523–1530.

[46] D. Nister and H. Stewenius. “Scalable Recognition with a Vocabulary Tree”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 2. 2006, pp. 2161–2168. DOI: 10.1109/CVPR.2006.264.

[47] Edwin Olson. “AprilTag: A robust and flexible visual fiducial system”. In: *2011 IEEE International Conference on Robotics and Automation* (2011), pp. 3400–3407.

[48] Volker Paelke. “Augmented reality in the smart factory: Supporting workers in an industry 4.0. environment”. In: *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. 2014, pp. 1–4. DOI: 10.1109/ETFA.2014.7005252.

[49] Namrata S. Pathkar and Neha S. Joshi. “GOOGLE GLASS: PROJECT GLASS”. In: 2014.

[50] Philipp Rauschnabel, Alexander Brem, and Young Ro. “Augmented Reality Smart Glasses: Definition, Conceptual Insights, and Managerial Importance”. In: *Working Paper, The University of Michigan-Dearborn* (July 2015).

[51] *Real Time pose estimation of a textured object*. URL: https://docs.opencv.org/master/dc/d2c/tutorial_real_time_pose.html (visited on 10/28/2021).

[52] *Rolling Effect*. URL: https://upload.wikimedia.org/wikipedia/commons/4/49/Rolling_shutter_SMIL.svg (visited on 10/31/2021).

[53] Francisco Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. “Speeded Up Detection of Squared Fiducial Markers”. In: *Image and Vision Computing* 76 (June 2018). DOI: 10.1016/j.imavis.2018.05.004.

[54] A. Rosenfeld. “Computer vision: basic principles”. In: *Proceedings of the IEEE* 76.8 (1988), pp. 863–868. DOI: 10.1109/5.5961.

[55] Edward Rosten and Tom Drummond. “Machine Learning for High-Speed Corner Detection”. In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443. ISBN: 978-3-540-33833-8.

[56] Edward Rosten, Reid Porter, and Tom Drummond. “Faster and Better: A Machine Learning Approach to Corner Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (2010), pp. 105–119. DOI: 10.1109/TPAMI.2008.275.

[57] Ethan Rublee et al. “ORB: an efficient alternative to SIFT or SURF”. In: Nov. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[58] Leonardo dos Santos et al. “Industry 4.0 collaborative networks for industrial performance”. In: *Journal of Manufacturing Technology Management* ahead-of-print (Nov. 2020). DOI: 10.1108/JMTM-04-2020-0156.

[59] Aron Sheldon et al. “Putting the AR in (AR) chitecture-Integrating voice recognition and gesture control for Augmented Reality interaction to enhance design practice”. In: (2019).

[60] Anil Kumar Singh. “Error detection and correction by hamming code”. In: *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. 2016, pp. 35–37. DOI: 10.1109/ICGTSPICC.2016.7955265.

[61] Breannan Smith et al. “Constraining Dense Hand Surface Tracking with Elasticity”. In: *ACM Trans. Graph.* 39.6 (Nov. 2020). ISSN: 0730-0301. DOI: 10.1145/3414685.3417768. URL: https://doi.org/10.1145/3414685.3417768.

[62] Anna Syberfeldt, Oscar Danielsson, and Patrik Gustavsson. “Augmented Reality Smart Glasses in the Smart Factory: Product Evaluation Guidelines and Review of Available Products”. In: *IEEE Access* 5 (2017), pp. 9118–9130. DOI: 10.1109/ACCESS.2017.2703952.

[63] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. “Visual SLAM algorithms: a survey from 2010 to 2016”. In: *IPSJ Transactions on Computer Vision and Applications* 9 (Dec. 2017). DOI: 10.1186/s41074-017-0027-2.

[64] *Tooling4G*. URL: https://tooling4g.toolingportugal.com/ (visited on 10/10/2021).

[65] Daniel Wagner and Dieter Schmalstieg. “ARToolKitPlus for Pose Tracking on Mobile Devices”. In: Jan. 2007.

[66] Michaela S. Wagner. “Google Glass: A Preemptive Look at Privacy Concerns”. In: *J. Telecommun. High Technol. Law* 11 (2013), pp. 475–.

[67] Ramy Zeineldin and Nawal El-Fishawy. “A Survey of RANSAC enhancements for Plane Detection in 3D Point Clouds”. In: 26 (July 2017), pp. 519–537. DOI: 10.21608/mjeer.2017.63627.

[68] Zhengyou Zhang. “A Flexible New Technique for Camera Calibration”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000). MSR-TR-98-71, Updated March 25, 1999, pp. 1330–1334. URL: https://www.microsoft.com/en-us/research/publication/a-flexible-new-technique-for-camera-calibration/.