An Augmented Reality System for Urban Environments using a Planar Building Façade Model

Eric McClean

Masters of Science

National University of Ireland Maynooth
Department of Computer Science
Faculty of Science and Engineering

February 2013

Head of Department: Dr. Adam Winstanley
Supervisor: John McDonald

Contents

List of Figures

Abstract

In recent years there has been a widespread adoption of smartphone technology. According to research by Google [Pha12] the U.S., New Zealand, Denmark, Ireland, Netherlands, Spain and Switzerland have a smartphone penetration greater than 40 percent. These phones are frequently equipped with with reasonable processing capabilities, internet access, high resolution cameras, and positioning sensors. This widespread adoption coupled with the hardware capability's of current generation smartphones present a great opportunity to develop mobile augmented reality (AR) applications for visualising geospatial information.

While smartphones are able to provide geolocalisation through GPS, the lack of accuracy does not allow for precise and robust alignment of the augmented data with the camera image. Computer vision techniques can be used to gain accurate alignment. However, these methods often require the creation a comprehensive 3D model of a scene. This can be intensive to compute and the storage of this data can become a problem when the system is required to work over the scale of a citywide area.

This thesis presents a mobile AR navigation system that uses a planar based representation in order to augment 3D data into an urban scene. By performing planar extraction the system can use the façades of building as reference for accurately augmenting content. This is turn removes the need for building a comprehensive 3D model for the scene. The system uses a client server architecture where a user takes an image within an urban setting with their smartphone which is forwarded to the server for processing. This processing includes extraction of the planar façades in the scene, camera pose estimation and, matching of the façades to a topological map of the environment. Once matched each of the the associated camera pose parameters in conjunction with relevant AR content can then be sent back to the mobile device. These parameters allow the system to integrate the augmented content into the view which can highlight and visualize geographically or contextually meaningful information about the scene.

Results presented demonstrate that the system provides an accurate and robust approach to AR in urban environments without associated complexity of creating metric 3D reconstructions of the environment.

Acknowledgements

I would like to thank my supervisor John McDonald for his continual help, guidance and understanding throughout this project. I am also very grateful to Dr. Guillaume Gales for his help and assistance in answering any queries that I had. I would also like to thank my parents and friends for their continued love and support.
Research presented in this thesis was funded by a Strategic Research Cluster grant (07/SRC/I1169) by under the National Development Plan. I would like to gratefully acknowledge this support.

1 Introduction

1.1 Augmented reality

Augmented Reality(AR) provides the ability to integrate computer-generated virtual data into real world images or video streams. Virtual Reality(VR) presents the user with a completely virtual world. AR in contrast seeks to enhance the user's view of the physical world with extra information relative to the real world objects within the scene. A familiar example of AR can be seen during sport broadcasts on TV. Here a white circle is place under a player as he moves about the pitch in real-time. This circle is used to indicate who the commentators are talking about. The goal of AR techniques is to allow for the integration of 3D data in a perceptually transparent manner that is contextually relevant to the user (i.e. as if it were a part of the scene). This is achieved by reprojecting 3D virtual objects to appear as if they were part of the scene and then superimposing them over the original image. An example of this process is illustrated in Fig. 1.1.

AR has been shown to have many possible applications in a wide variety of fields with its initial roots in military, industrial, and medical applications. The ability to intuitively highlight or integrate computer generated information into real world surroundings has a wide variety of applications with examples now being seen in many other commercial areas. Some examples include Tourism[FSL05], Marketing[Leg11],

Figure 1.1: *Augmented Reality Example – The figure shows the AR game Invizimals on the PSP Vita [Son12]. In this a player can fight 3*D *monsters in real world settings*

Navigation[Lay11], Medical[FNFB04], Entertainment and Gaming applications[Son12].

1.1.1 Camera pose estimation

In order to create effective AR systems it is important to have consistent alignment of the 3D virtual objects with the real world image. This is achieved by estimating the position and orientation of the camera which has taken the image with respect to the scene or object that we wish to augment. This is known as the *camera pose estimation* problem[HZ04] or *registration* problem. The camera pose information is then used to match the real camera with the virtual camera that displays the 3D objects. Once this is done the virtual data can be reprojected and overlaid on the real image. Camera pose estimation can be computationally expensive to solve with the most accurate methods requiring iterative non-linear optimisation techniques. In recent years, as AR applications have begun to appear on mobile devices this computational requirement has been

seen as a significant challenge. In particular, given the limited computational power of these devices, approaches that reduce this complexity are required.

1.1.2 Pose estimation approaches

Within AR there are various different strategies employed to estimate the camera pose. We can categorise these approaches along the following lines:

1. ***Proprioceptive sensors*** – Making use of sensors (e.g. accelerometers, GPS, and digital compasses) for obtaining the camera position and orientation.

2. ***Computer vision*** – Taking advantage of features within the image itself in order to estimate and extract the camera pose from the images taken by the camera.

3. ***Hybrid approach*** – Combining both computer vision and proprioceptive sensors in order to gain camera registration.

Full details of each of these approaches is provided in Chapter 2, however in order to motivate the problem addressed in this thesis it is necessary to understand the advantages and disadvantages of each of these approaches. The advantage of using proprioceptive sensors is that they provide a direct measurement of the camera pose thereby avoiding the high computational complexity associated with other techniques. However due to the inherent inaccuracy of these sensors, their use in mobile AR applications has been restricted to situations where pixel level positioning accuracy is not possible (e.g. roughly displaying the location of a bus stop at a distance).
The use of computer vision techniques on the other hand allow for highly accurate camera registration and robust alignment of the data with the camera image. In general, computer vision approaches for acquiring camera registration can be split into two categories: marker based tracking and markerless tracking.
In marker based tracking the calculations for the camera pose are simplified by inserting artificial markers(fiducial markers) into the scene. These

fiducial markers are purposely designed to be distinctive and easy to detect, segment and track within the image. An example of marker based AR can be see in Fig 1.2. While this method works well over small to medium scale controlled environments (e.g. a building or even a campus) it is not suitable for use within a large uncontrolled environment such as a city since it is simply impractical to place makers everywhere within the environment that the system may need. This impracticality demonstrates the need for markerless or natural feature tracking methods for AR.

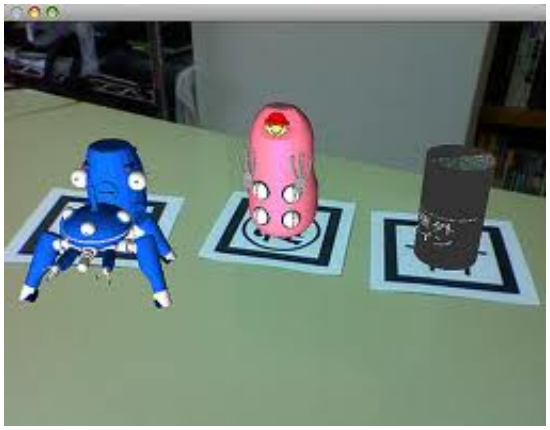

Figure 1.2: *Marker based tracking for AR [KB99]*

Markerless methods calculate the pose information of the camera from naturally occurring features within the scene such as planes, edges, or corner points. Hence, the markerless approach does not require artificial elements to be placed within the scene or for the scene to be prepared in any way. Markerless tracking usually relies on the processing of multiple images in order to acquire accurate camera pose information. This method is the more computationally costly of the two computer vision based approaches since it means that an estimation of the 3D structure of the scene is normally required. This 3D model of the scene can then be used as a reference for both pose estimation and to artificially add elements into the scene. An example of a markerless based AR system can be seen in Fig. 1.3

Figure 1.3: *The figure shows Markerless tracking [RD06a]*

Finally hybrid approaches combine both the proprioceptive sensor and computer vision approaches. This technique can take advantage of the accuracy of computer vision but is able to offset some of the computational costs associated by using hardware sensors. For example, proprioceptive sensors can be used to give a seed value to the vision based techniques. This allows us to cut down on the search space and apply vision to a sub region. This can give a gain in effeciency or reduction in computational complexity. The main difficulty with the hybrid approach can be in finding the balance between the computer vision and the hardware based method. This is dependant on the level of alignment accuracy required and the hardware capabilities available to the application.

1.2 Problem statement

The aim of this thesis is the development of a mobile AR system for positioning and navigation which works over a large urban setting. This thesis builds on previous work of the Computer Vision and Imaging Lab at NUI Maynooth on planar extraction algorithms described in [CM12] and focusses on the problem of applying this algorithm to create a mobile augmented reality navigation system. The contributions of this thesis are:

- *A method for augmenting* 3D *content using a planar building façade model of the environment.*

- *A intuitive method for authoring* 3D *AR targeted at non-expert users.*
- *Development of a mobile AR system for user based navigation within an urban environment, demonstrating the effectiveness of the two contributions above.*

This system must be able to author, augment and share relevant navigation and point-of-interest (POI) information using single arbitrary images of urban scenes. The principal requirements for the resulting system are:

1. *The system must first be able to augment* 3D *POI data at a pixel level accuracy within an urban scene.* As mentioned above a computer vision approach is the only option for gaining alignment of the 3D augmented data at pixel level accuracy relative to a real-world scene. Furthermore, in order to facilitate operation over a large scale and in an uncontrolled environment the approach must also be markerless. Typically this leads to computationally expensive approaches which can be beyond the capabilities of current mobile devices. However, since the aim is to create a navigation system for an urban setting we can exploit a number of assumptions about such environments which allows for a more simplified approach to markerless AR. In particular, the application of standard markerless computer vision approaches involve the process of computing a comprehensive 3D model. Within an urban scene however this can be seen as unnecessary since the façades in such environments are frequently the main reference point for the user. To exploit the building façades we use the fact that they are often planar and will contain a large number of rectilinear structures (i.e. containing parallel lines in orthogonal directions). The system can therefore make use of the projection of parallel lines thereby permitting the recovery of the original 3D geometry of the building façades from single input 2D images. These extracted planes can then be used to represent the 3D spatial layout of a scene. Although the underlying representation is not a complete 3D representation of the environment, as will be demonstrated in later chapters it does allow for the creation of a comprehensive 3D AR system with significantly reduced complexity in the approach.
2. *The system should support non-expert authoring of AR content and the sharing of that content between users.* This objective is achieved

by using the simplicity of the underlying façade based model to provide an intuitive approach to AR content authoring and sharing. In order for the user to add new content, an authoring client has been developed where a user can insert new 3D items relative to the extracted façades. This permits a straightforward method for integrating augmented content into the view which can highlight and visualize geographically or contextually meaningful information about the scene. This is in contrast to other systems whereby the authoring involves complex 3D modelling tools typically requiring specialised knowledge on the part of the author. Furthermore the use of a centralised global model facilitates sharing of content between users thereby allowing the AR system to be built around user generated content (UGC).

3. *The system should be scalable to a level that it can operate over a large(citywide) urban area and should support mobile clients.* This requirement is principally satisfied through the use of a client server based approach. Here, the data needed for augmentation is stored on the server side so the client only accesses local regions of the model and hence does not require the entire model to be stored on the device. In terms of the processing requirements on the server side, the computational complexity is reduced through the use of the façade based model, which essentially models the environment as a set of local maps which are combined globally through a set of topological constraints. The implication of this representation is that the complexity of adding new images to the system is bounded by the complexity of the local map, since we avoid the complex issue of global consistency associated with metric 3D reconstruction techniques. Furthermore the computational complexity of the image matching procedure required for the localisation step (i.e. which uses the input image to index into the global map) is reduced by limiting the scope of the search by including location data from the GPS sensor of the mobile device.

1.3 Mathematical notation

Given the importance of the camera pose estimation problem in this work, this section provides an explanation of the mathematical notation used throughout the remainder of the thesis.

1.3.1 Representing cameras

The perspective camera model is used to denote a projective mapping of a three dimensional point in the real world to the corresponding two dimensional point in the image plane. The 3D homogeneous world coordinate point $\mathbf{X} \in \mathbb{R}^4$ is mapped by the camera to the point $x \in \mathbb{R}^3$ by the transformation:

$$\mathbf{x} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X} \tag{1.1}$$

Equation 1.1 shows the factorisation of the perspective camera model into the *intrinsic* camera matrix $\mathbf{K}$ and *extrinsic* camera matrix, $[\mathbf{R}|\mathbf{t}]$. Here $\mathbf{R}$ and $\mathbf{t}$ represent the rotation and translation, respectively, between the world and camera coordinate systems. The elements of the intrinsic matrix are shown in Equation 1.2.

$$\mathbf{K} = \begin{bmatrix} \alpha_x & 0 & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

where, the parameters α_x and α_y represent focal length in terms of pixels. $\alpha_x = f.m_x$ and $\alpha_y = f.m_y$ where f is the focal length in millimetres, m_x and m_y are conversion factors relating millimetres to pixels and, u_0 and v_0 represent the x- and y-coordinate of the principal point. Standard camera calibration techniques[Zha00] can be used for estimation of these parameters. Throughout this thesis unless otherwise stated the camera is assumed to be a calibrated i.e. $\mathbf{K}$ are known. While real cameras exhibit radial and tangential distortion, throughout this thesis it is assumed that standard methods have been applied to remove these effects.
The extrinsic or external camera parameters determine the mapping from 3D points in the world coordinate system to their representation in the

camera coordinate system. Hence, given a calibrated cameras, it is by estimating the extrinsics of the camera that we can fully estimate the model described in Eq. 1.1 and as a consequence perform the reprojection required by the AR system.

1.4 Structure of thesis

The remainder of this thesis is structured as follows: In Chapter 2 a review of the current state of AR is presented. Chapter 3 gives details of the planar extraction method referred to in Section 1.2. Using this planar extraction technique as a starting point, Chapter 4 will describe the façade based approach for augmented reality developed during this thesis. This method creates augmented content using only a single image. Chapter 5 describes a client server based AR system that provides the user with positioning and navigation information. This system satisfies the requirements listed in Section 1.1 and demonstrates the applicability of the techniques from Chapters 4 and 5 in a real-world environment. Chapter 6 will summarise the contribution of the thesis and identify potential avenues of future research.

2 Related work

2.1 Overview

Computer Mediated Reality centres around the altering of one's perception of reality by inserting/removing elements or manipulating it in some way using a computer [MK94, Str96, Str97]. As can be seen from the taxonomy of mediated reality in Fig. 2.1 both Augmented Reality (AR) and Virtual Reality (VR) fall into a sub category of this topic.

VR coined by Jaron Lanier [KHS89] in 1989, applies to creating computer-simulated environments which aim to emulate the physical presence of a place in the real world. An example of a VR system is shown in Fig. 2.2 (a). In this system images are projected onto the floor, walls and ceiling to give the user an immersive experience of the virtual world.

In contrast to this AR seeks to alter the user's perception of their physical environment rather than completely emulate another (see in Fig. 2.2 (b)). AR allows for the blending or augmentation of real world environments with synthetic computer generated information in order to generate a composite view of both. AR also extends into the audio domain with work on audio/aural augmenting techniques having been presented in [Bed95, CAK93, MBWF97, HJT+04]. For the purpose of this thesis however, the focus will be solely on visual based AR methods.

Visual based AR techniques first began with Sutherland's [Sut68] head-mounted display in the 1960's. Although it began in the 1960's the

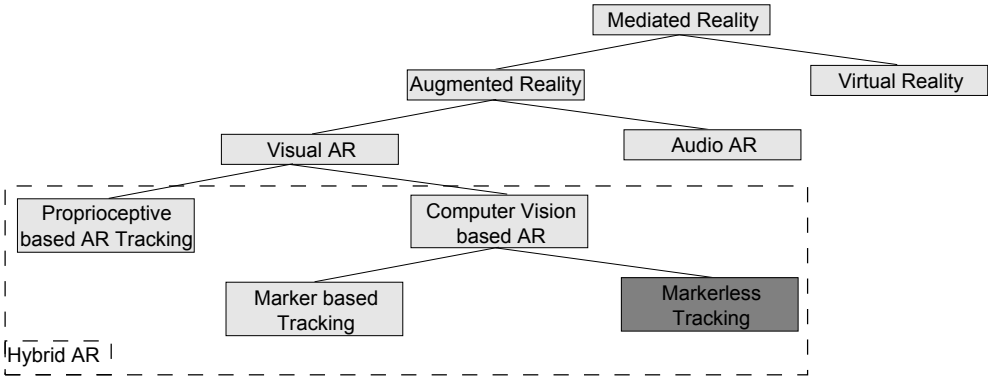

Figure 2.1: *Augmented Reality Categories - Categories of Mediated Reality*

(a)[Pap01] (b)[Wik09]

Figure 2.2: *AR and VR Reality Examples - A side by side view of AR and VR*

term AR is not believed to have been coined until the 1990s by Thomas Caudell [CM92]. A commonly accepted definition by Azuma [Azu97] states that an AR system contains the following properties:

- *It combines real and virtual objects with each other*

- *It registers (aligns) these virtual objects with the real world in three dimensions*
- *It runs interactively*

This is typically achieved by tracking the 6 Degrees of Freedom (DoF) pose of the camera or the *user's* view of the scene and using it to seamlessly overlay computer generated content in such a way that it is consistently registered within the real world image or camera feed. In fact, the accurate estimation of camera pose is one of the main challenges for the AR system. Registration between these two environments in practice presents a difficult challenge to overcome due to the complexity and processing power required. As mentioned in Chapter 1, approaches to this problem divide into 3 main categories:

1. Registration using proprioceptive sensors.
2. Registration using computer vision.
3. Hybrid approaches combining both the proprioceptive sensors and computer vision techniques.

2.2 Proprioceptive sensor based AR

Proprioceptive sensor based registration methods were the earliest method developed for creating AR. In this thesis proprioceptive techniques are defined as those that use sensors that directly measure the orientation of the camera(e.g. accelerometer and digital compass) [CM92, KO97, FHP98]. In Sutherlands [Sut68] work in the 1960s he tracked the head position of the user and presented the aligned graphics through a see-through head mounted displays (HMD). Much of the work that followed [CM92, KO97, FHP98, HFT+99] up until the late 90s required measuring the position and orientation of some part of the user's body in order to achieve registration. This evolved under the name of wearable computing and had some promising applications from assisting with complex machinery [SB97] to helping doctors with patient biopsies [FSP+96]. These

systems however required having to wear specialist headset that often needed precise calibration to work correctly.

Figure 2.3: *Layar Example [Lay11] – Augmenting a city scene with historical data using Layar.*

Magnetic, acoustic, ultrasonic, and mechanical sensors have all been used to varying degrees of success. A detailed review of these technologies can be found in [BDR01]. These systems can work well over a smaller area however the limited accuracy of the techniques result in a noticeable drift of the camera registration when generated over a larger area.
In the late 90s hardware became sufficiently small and powerful to allow mobile devices to be used for AR systems. One of the early prototypes for this includes Hollerer's mobile AR system (MARS) [HFT+99]. This system used GPS for positioning and a gyrometer and accelerometers in order to compute the 6 DoF camera orientation for the user. GPS and accelerometers have become common, even standard on most modern mobile phones, removing the need for proprietary hardware to create AR systems. Commercial systems such as Layar [Lay11], Wikitude [Wik09] and ARNav [ARN12] have all made use of such hardware on smart phones in order to create AR based navigation systems. An example of this can be seen in Fig. 2.3. Although these systems can operate at a global

scale they do suffer from the limited accuracy of the sensors and as such do not allow for accurate and robust registration of the data with the camera image. This results in a user experience where the positioning of graphical elements within the image is inaccurate. For example using the proprioceptive sensors, Fig. 2.3 shows that a dot can be augmented in a general area however in order to accurately augment onto a specific feature the use of the image data is required.

2.3 Computer vision based AR

Computer vision based tracking approaches, unlike proprioceptive based tracking, do not suffer from the limiting registration accuracy. These approaches utilise geometric and photometric information from images of the scene in order to estimate the pose of the camera. This permits significantly increased accuracy of registration. This accuracy comes at a computational cost which has given rise to two types of vision based approaches in order to cater to it:

- *Marker based tracking*
- *Markerless based tracking*

2.3.1 Marker based AR

The use of visual markers for registration is common and popular in AR. Fiducial markers are easily identifiable predefined markers that are placed into the field of view. These can then be used as a point of reference in the image allowing for easier recovery of the 3D camera position. In 1998 Rekimotos Matrix Code [Rek98] was released and has been credited as being the first marker based tracker developed for AR. It made use of 2D barcode patterns inside square planar shapes for pose estimation. Since then there has been many improvements in marker based tracking with Zhang et al. [ZFN02] comparing several of the leading approaches.

The square marker method became particularly popular with the release of the open source library ARToolkit [KB99] in 1999. This was a pose tracking library that was able to track the camera's 6 DoF pose using square fiducial markers. It found the square marker in the image by thresholding the image, performing contour extraction and then line fitting to acquire the four straight line of the square marker. The pose estimation used the extracted corner points of the square to guess an initial coarse pose estimation. It created variations on this by rotating the pose around all 3 axis and reprojecting it to select the correct hypotheses pose. This was a relatively simplistic method but it was very fast and allowed for real time augmented video (See Fig. 2.4). This gave a wide audience the ability to perform image based tracking using commercial off-the-shelf (COTS) cameras.

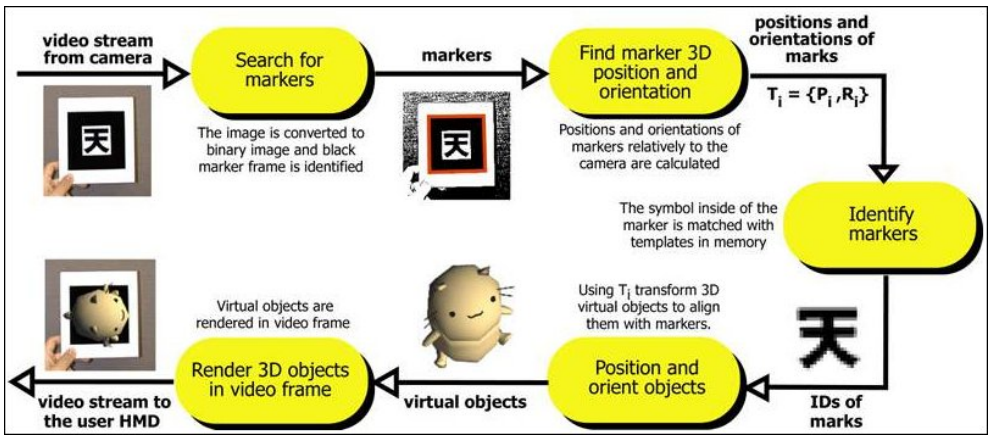

Figure 2.4: *ARtoolkit Overview [ART] – Overview of the steps for ARToolkit*

The ARTags [Fia04, Fia05a] library was inspired by ARToolkit and aimed to reduce the rate of false positives and poor detection rates that ARToolkit could be susceptible to. The original ARToolkit method for creating tags relied on user created binary images. However it became clear that visual appearance of the tags had a dramatic impact on the detectability, recognition and the pose estimation. ARTags replaced the morphology operations on a binary image that ARToolkit used with an edge detection based system. They also employed a digital encoding

method for the fiducial markers. This is a predefined digital pattern inside the tag which is not as susceptible to false positives as the user created binary images of ARToolkit. This can be seen in Fig. 2.5. This allowed for a greater robustness to lighting, occlusion and lower detection of false positives.

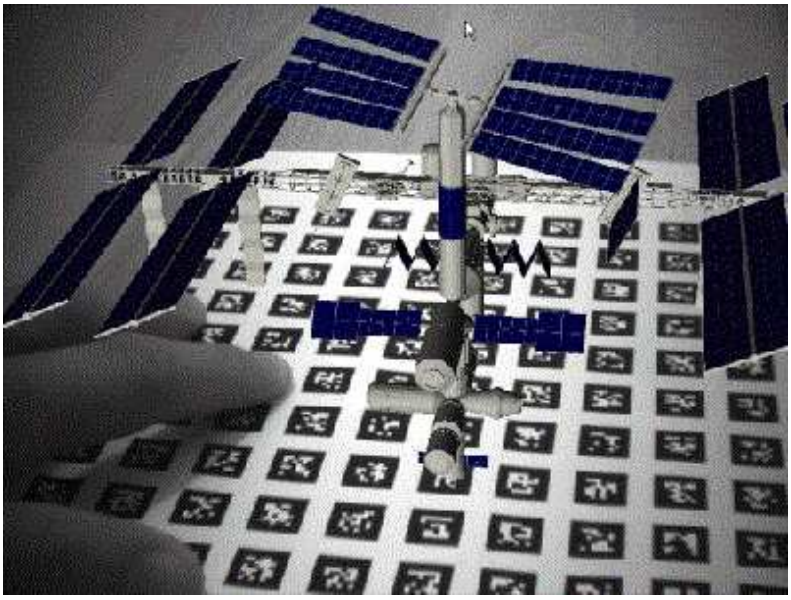

Figure 2.5: *ARTag Example [Fia04] – ARTag registering against a digitally encoded tags*

The improvements created by ARTags encouraged further enhancements to ARToolkit with ARToolkitPlus [WRM+08] being released in 2004. This added multiple improvements including making use of a digital encoding system for its markers. A comparison of these two systems can be found in [Fia05b]. ARToolkitPlus has since evolved into a full AR framework, the *Studierstube* Tracker [WS09a, WS09b]. This further added improvements by using multiple different marker types, pose estimators and thresholding algorithms. The strengths and weaknesses of these are outlined in [WLS08]. Studierstube was designed to be used on a mobile platform and as such had far better memory management as opposed to other systems which were just ports of the PC version.

One of the latest approaches to fiducial marker tracking is that of AprilTags [Ols11]. Unlike ARTags and Studierstube this is a fully open source AR library. This system demonstrated a graph based image segmentation for line estimation and a quad extraction method (see Fig. 2.6). They also added a new coding system that added greater robustness to rotation and false positives arising from natural imagery. These changes added a significant robustness to rotation, occlusion and further reduced false positive rates.

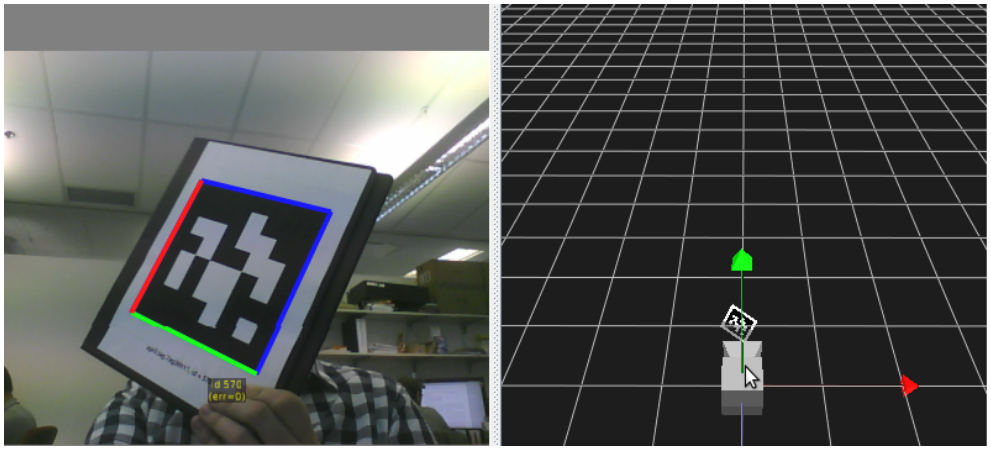

Figure 2.6: *AprilTags Example [Ols11] - AprilTags pose estimation*

Marker based technologies have a high degree of accuracy for estimation of the camera pose however the marker based approach only works under the confining circumstances that it is feasible to put artificial markers within the scene that you are trying to augment. In an uncontrolled environment (e.g. outdoors) this method does not scale and it simply becomes impractical to use marker based technologies.

2.3.2 Markerless AR

Given the restrictions of marker based systems, over the past decade considerable research effort has focused on calculation of the camera pose

information from naturally occurring features in the scene such as planes, edges, or corner points. Park [PYN98] demonstrated how to extend beyond fiducial tracking using point and region tracking. This system dynamically acquired new natural feature points so that it could still update the pose calculation after the initial fiducial had left the view. In [SL04] they described a natural feature method that matched feature points against that of a database of feature descriptors.
Many of the other natural feature techniques[KD03] rely on an *a-priori* model or map of the scene to track against. One of the advantages of this is that it increases the robustness of the pose estimation and reduces the effect of outliers. This however requires a comprehensive off-line 3D model to be created and over a large area this becomes infeasible.
One of the most popular classes of methods for solving this problem is based on Structure-from-Motion (SfM) or the related online approach of Visual Simultaneous Localization and Mapping (SLAM) [DWB06]. Such methods simultaneously estimate camera position as well as the 3D structure of the imaged scene. These systems can deal with uncalibrated image sequences acquired with a hand held camera. Broadly speaking the processing involved in these techniques can be divided into two components:

- Feature detection and matching (cf. [MTS$^+$05] for a detailed review of recent approaches).

- Structure and motion estimation. These SfM-based techniques permit accurate 3D registration and pose estimation in unstructured environments.

In 2007 Klein et. al [KM07] demonstrated Parallel Tracking and Mapping (PTAM) shown in Fig. 2.7. PTAM separated the tracking and mapping onto different threads. In this map is initialised from a stereo pair of images using the 5-Point Algorithm. Mapping is then based on key frames which are batch processed using Bundle Adjustment. An epipolar search is used to initialise and track new points. This gives PTAM the ability to augment a scene in real time while concurrently generating a map for that scene to track against. This allows PTAM to operate without having any prior knowledge of the users environment. This had the issue of only working over small AR workspaces but this issue was dealt with by the release of Parallel Tracking and Multiple Mapping (PTAMM) [CKM08]. PTAMM had the same approach as the original PTAM system however

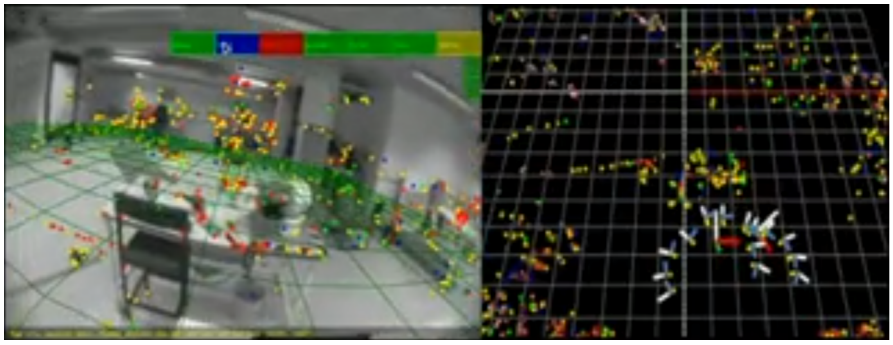

Figure 2.7: *PTAM Example [KM07] – PTAM registering against the natural features. The map generated by the scene is on the right.*

it worked over a larger area as it was able to swap between multiple generated maps of an area.
One of the main shortcomings of such methods is that they need a set of related images of a scene taken at similar viewpoints to establish frame-to-frame correspondences. Furthermore these methods require processing a set of keyframes in a bundle adjustment operation [HZ04] to achieve global optimum. This procedure is computationally expensive for interactive applications. The solution proposed in this thesis is that of a single image planar based representation in order to augment 3D data into an urban scene. By using planar extraction the system can use the façades of buildings as reference for accurately augmenting content. This is turn removes the need for building a comprehensive 3D model for the scene. This is discussed in Chapter 3.

2.4 AR authoring

One of the main factors in the usability and utility of AR systems is the ease with which users can create new AR content. Many solutions and applications [WW11, Led04, PTB+01] have been suggested for performing this task with the majority of current approaches involv-

ing the creation of a full three-dimensional representation of the environment, image-based representations of the scene or panoramic based solutions [LWMS12, KCSC10]. 3D data can then be positioned relative to these representations.

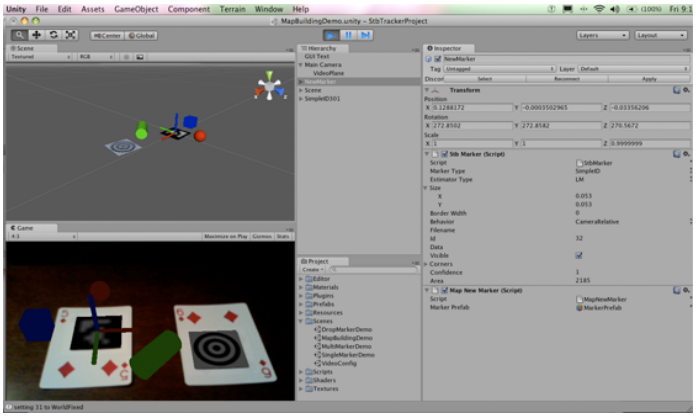

Figure 2.8: *UART Example [GAT10] – Authoring pluggin for ARToolkit*

Authoring content therefore involves working in these representations, which requires complicated user interfaces for the non-expert user. This approach has been shown to be more difficult to manipulate for content developers and as a result authoring systems have often been created as plugin's for existing well known 3D rendering systems [Aut13, GAT10, Uni09] (see Fig. 2.8). This has also encouraged authoring tools to move away from the traditional methods of input [LNBK04].
These solutions entail the user to author content relative to a 3D model which requires complicated user interfaces for the non-expert user. The authoring solution proposed in this thesis aims to overcome many of these issues by using a method of authoring AR content based on planar façade extraction. With this method the extracted plane can be used to create a front parallel view which in turn can be used as a frame of reference for authoring 3D AR content. Authoring then consists of positioning 3D content relative only to the front parallel view of the façade. This approach permits a simplified method for integrating augmented content into a view. Full details of approach are discussed in detail in Chapter 4 and 5

3 Façade extraction

3.1 Overview

The overall goal of this thesis is to create a mobile AR system for use in urban environments. Over the two previous chapters the various different methods used to augment a view with 3D data have been introduced. Most of these methods do this by computing a full 3D reconstruction of the scene. This however can be computationally expensive and as such is unsuitable for AR on a mobile platform. In addition, these approaches generally require multiple images and precise calibration procedures.

The solution proposed in this thesis is a façade based AR system. This allows for a single image from an urban scene to be augmented without the need for a full 3D reconstruction of the scene. The novelty of the approach lies in extracting the planar building façades to compute a simplified representation of the scene. This simplified representation avoids the computational complexity of computing the full 3D structure of the scene whilst providing adequate information to augment those façades. Another advantage of using this approach is that the underlying planar façade extraction is both robust to perspective distortions, viewpoint changes and allows augmenting using a only a single image as reference.

The work presented in this chapter builds on a previously developed planar extraction algorithm detailed in [CM12]. A major part of this thesis involved understanding this technique, applying it and then extending it

to develop the AR system described in Chapters 4 and 5. Given that this technique is at the core of our system this chapter provides a review of the algorithm. Throughout this chapter, results are shown on data sets that have been collected during the implementation of this thesis. These are shown in order to assist the reader's understanding of the core principles of the technique.

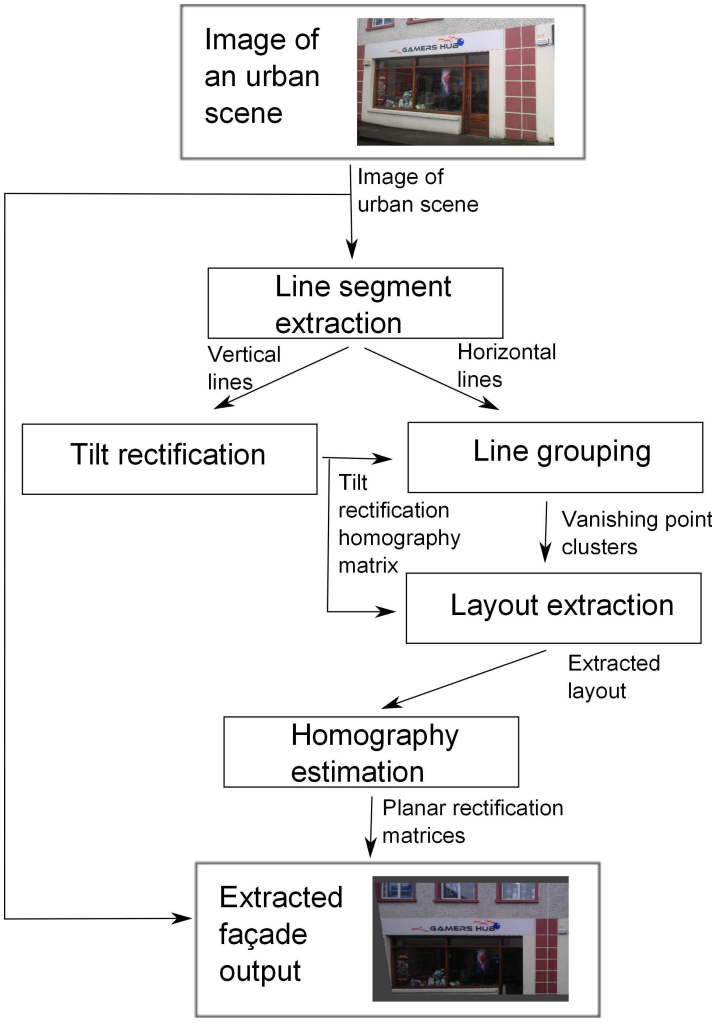

Figure 3.1: *Façade extraction flow diagram – The figure shows the flow of each of the steps in the façade extraction process.*

The façade extraction algorithm consists of finding the homography between each of the buildings façades and the input image. This technique is based on detecting vanishing points in order to estimate the direction of each plane. The façade extraction contains five major steps:

1. ***Line segment extraction*** – This step extracts line segments that are identified within the input image.
2. ***Tilt rectification*** – Tilt rectification rectifies the original image such that vertical boundaries of a building in 3D world become vertical in the rectified image. This compensates for the pitch and roll of the camera and thus removes two degrees of freedom from the subsequent processing.
3. ***Parallel line grouping*** – The line grouping is used to identify horizontal vanishing directions and assign each line within the tilt rectified image to one of the directions.
4. ***Layout Extraction*** – This step assigns a direction to a subset of the image which can then be used to estimate the layout of the scene (i.e. the number of and position of the façades in the image)
5. ***Homography estimation*** – For each extracted façade this step estimates the homography between the buildings façade and the original image.

The details of each step are presented in following subsections. An overview of this complete algorithm, including each of these steps and their interaction with the other steps can be seen in Fig. 3.1. Once complete, the extracted planes (i.e. the associated homographies) can be then used to graphically augment the original image of an urban scene. This AR step will be discussed in Chapter 4

3.2 Line segment extraction

The line segment extraction is the first step in the façade extraction process. A line segment is a section of a line that is bounded by two

definite end points. This step is described in [KZ02] and extracts both a set of vertical line segments and horizontal line segments from the input image.
As can be seen in Fig. 3.2 the input for this algorithm is an image of a street-scape taken in an urban setting. The principal assumption made is that the image includes building façades, where the façades includes some dominant rectilinear structure (e.g. due to window and door structures, building boundaries, etc.). We also assume that the image is taken from an approximately upright orientation. We note that mobile devices commonly incorporate sensors to allow them to measure the vertical orientation of the image and therefore this later assumption is not considered restrictive.

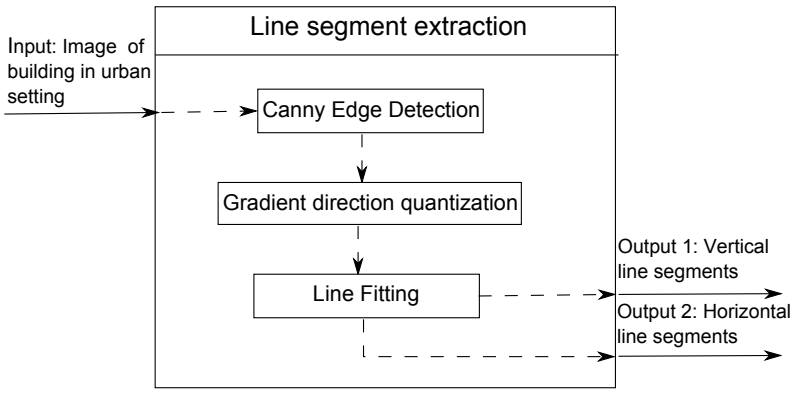

Figure 3.2: *Line segment extraction – The figure shows the flow of each of the steps in the line segment extraction*

The first part of the line segment extraction is performing edge detection using the Canny edge detector. For each resulting edge point the image derivatives are then computed. The orientations θ are given by:

$$\theta = \tan^{-1}\left(\frac{\frac{\partial I}{\partial x}}{\frac{\partial I}{\partial y}}\right) \tag{3.1}$$

The gradient direction is quantized into k ranges where $k = 8$ for our purposes. These are grouped into bins where their label corresponds to

the orientation of the edge pixels stored within it. This is given by:

$$bin = \frac{\theta}{\pi k} \mod k \tag{3.2}$$

where *bin* is the corresponding bin label for the orientation.

The line fitting stage then follows in order to generate the straight lines within the image. When considering line fitting, the pixels that fall into bins whose gradient orientation is $\frac{\pm\pi}{8}$ are also considered. This aggregates the pixels belonging to the same line segment that end up falling into different bins. When line fitting, larger segments are considered as stronger candidates for actual façade edges in the image therefore lines that are longer than a certain threshold are kept. Empirically we have found that lines with length 30 pixels and above provide a satisfactory threshold. Finally, the lines that are extracted are separated into vertical and horizontal lines. Lines that are within $\frac{\pm\pi}{6}$ radians of the vertical image column are assumed to be approximately vertical lines and lines that are within $\frac{\pm\pi}{6}$ radians of the horizontal image column are assumed to be approximately horizontal lines.

Figure 3.3: *Segment Extraction Result – The figure shows a segment extraction from multiple building façades.*

Examples of the line segment extraction are shown in Fig. 3.3. As can be seen, these lines are classified into horizontal lines (pink) and vertical lines (green). The line segments extracted which are approximately

vertical are input for the tilt rectification step. Those lines which are corresponding to horizontal parallel lines are used as input for the line grouping stage.
Although it is assumed throughout that the lines being extracted correspond to lines on the building façade this is of course not always the case within natural scenes. Natural objects such as cars, trees and people can occlude the view of the building façade. These objects however lead to short intermittent lines. The line length threshold helps to prevent many of these lines from getting through. The lines being filtered into those within a threshold of the vertical and horizontal image columns also helps decrease the effects of occlusions of lines that are extracted

3.3 Camera tilt rectification

When a user takes the original input image this will generally be done from ground level with the principal axis directed up towards the building façade. This adds a tilt effect to the building façade in the image. The purpose of the tilt rectification step is to remove this effect and output an image that shows the building façade as if the camera was parallel to the ground plane. This is achieved by identifying the two angles that compensate the pitch and roll of the camera. The result of this process will be to remove two degrees of freedom from the camera orientation and therefore simplify subsequent processing steps (see Fig. 3.4).
The input for the tilt rectification process is the approximately vertical lines that are the output from the line segment extraction. A flow diagram of this step can be seen in Fig. 3.5.
The output of the tilt rectification step is a 3×3 homography $\mathbf{H}_{tilt}$ which removes the effects of of the pitch (θ) and roll (ϕ) and is given by:

$$\begin{aligned} \begin{bmatrix} i' \\ j' \\ k \end{bmatrix} &= \mathbf{H}_{tilt} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \\ &= \mathbf{K}\mathbf{R}_{\theta}^{-1}\mathbf{R}_{\phi}^{-1}\mathbf{K}^{-1} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \end{aligned} \tag{3.3}$$

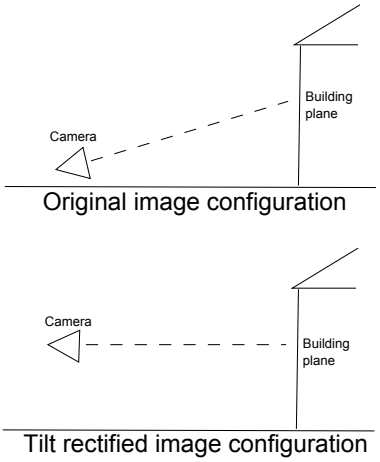

Figure 3.4: *Tilt rectification camera positions – The figure shows the difference in camera viewing angle when an image is rectified for tilt.*

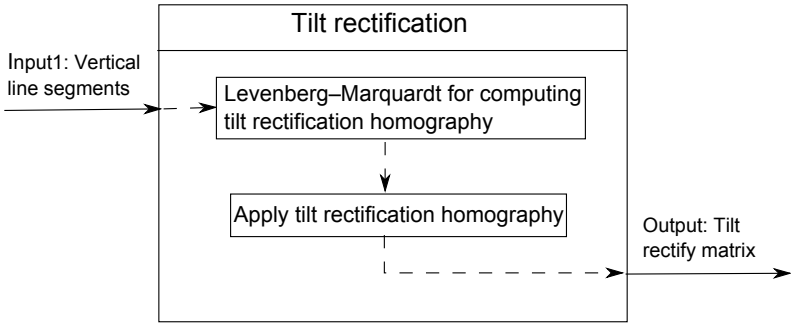

Figure 3.5: *Tilt rectification flow diagram – The figure shows the flow of each of the steps in the camera tilt rectification step.*

where $\mathbf{K}$ is the known calibrated intrinsics of the camera, $\mathbf{R}_{\theta}^{-1}$ is the inverse rotation of the camera pitch and $\mathbf{R}_{\phi}^{-1}$ is the inverse rotation of the camera roll. Expanding eq. 3.3 the transformation matrix that gives the tilt rectification can be rewritten as follows:

$$
\begin{aligned}
\begin{bmatrix} i' \\ j' \\ k \end{bmatrix} &= \mathbf{K}\mathbf{R}_{\theta}^{-1}\mathbf{R}_{\phi}^{-1} \begin{bmatrix} \bar{i} \\ \bar{j} \\ 1 \end{bmatrix} \\
&= \mathbf{K} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \bar{i} \\ \bar{j} \\ 1 \end{bmatrix} \\
&= \mathbf{K} \begin{bmatrix} \cos\theta & \sin\theta\sin\phi & -\sin\theta\cos\phi \\ 0 & \cos\phi & \sin\phi \\ \sin\theta & -\cos\theta\sin\phi & \cos\theta\sin\phi \end{bmatrix} \begin{bmatrix} \bar{i} \\ \bar{j} \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta\sin\phi & -\sin\theta\cos\phi \\ 0 & \cos\phi & \sin\phi \\ \sin\theta & -\cos\theta\sin\phi & \cos\theta\sin\phi \end{bmatrix} \begin{bmatrix} \bar{i} \\ \bar{j} \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} f\bar{i}\cos\theta + (f\bar{j}\sin\phi - f\cos\phi)\sin\theta \\ f\bar{j}\cos\phi + f\sin\phi \\ -(\bar{j}\sin\phi - \cos\phi)\cos\theta + \bar{i}\sin\theta \end{bmatrix}
\end{aligned}
\tag{3.4}
$$

where $(\bar{i}, \bar{j})$ are the pre-normalized coordinates. To calculate θ and ϕ we apply Levenberg-Marquardt (LM) optimisation to minimizes the difference of the end points' column coordinates of the vertical input lines.
LM is an optimization algorithm that is used to locate the minimum of a function,

$$S(\beta) = \sum_{i=1}^{m} f(x_i, \beta)^2 \tag{3.5}$$

where, $\boldsymbol{\beta}$ is the vector of parameters(unknown), f is the model function and $x_i \in \mathbf{x}$ vector of input data. It does this by iteratively adjusts the vector of parameters of the model function β by $\boldsymbol{\beta} + \boldsymbol{\delta}$.This is done until the found values converge on a solution that minimises the function. The δ values are computed using the first order Taylor approximation.
To minimise ϕ and θ the algorithm endeavours to minimizes the difference between the end points of the vertical line segments. The input to the LM procedure is both the set of vertical lines defined by their two endpoints A, of coordinates (i_A, j_A), and B, of coordinates (i_B, j_B) given by:

$$\mathbf{X} = \begin{bmatrix} i_A & j_A & i_B & j_B \end{bmatrix} \tag{3.6}$$

The model function f represents the column differences of the 2 endpoints

for each of the vertical lines. This is given by:

$$\begin{aligned} f(x_i, \beta) &= ((j_A)'_i - (j_B)'_i)^2 \\ &= \frac{f(j_A)_i \cos\phi + f \sin\phi}{((j_A)_i \sin\phi - \cos\phi)\cos\theta - (i_A)_i \sin\theta} \\ &- \frac{f(j_B)_i \cos\phi + f \sin\phi}{((j_B)_i \sin\phi - \cos\phi)\cos\theta - (i_B)_i \sin\theta} \end{aligned} \tag{3.7}$$

where $(j_A)'_i, (j_B)'_i$ are the column coordinates of the line x_i after transformation and f is the focal length. The first order Taylor approximation can be solved analytically.
The solutions for θ and ϕ can then be used to compute the 3×3 tilt rectified homography matrix H_{tilt} to transform vertical lines to become vertical in the image. This transformation is applied where each pixel in the rectified image is given by eq. 3.3 where the colour is given by the colour of the pixel from the original image. This creates a rectified view where camera tilt effect is removed.
An example of the tilt rectification process can be seen in Fig 3.6. Here the images have been taken at two separate angles of tilt. The input image can be seen on the left. Once tilt rectified the vertical line segments become parallel in the rectified image (right). The tilt effects are clearly apparent in the original image. Rectangular structure will appear trapezoidal which is wider at the bottom in the case of camera pitching up. After rectification, the images of vertical world lines become parallel to the image columns.

Figure 3.6: *Multiple angle tilt rectification – The figure shows the same building tilt rectified from 2 different angles of tilt. The extracted horizontal lines (pink) and vertical lines (green) line segments are highlighted.*

3.4 Line grouping

The purpose of the line grouping step is to identify the dominant planar directions within the scene. The line grouping technique exploits the fact that man-made structures, particularly in urban environments, often contain a large number of horizontal parallel building lines. A group of lines which are parallel will converge to a point in image space, known as a vanishing point. This is illustrated in Fig 3.7. The line grouping step

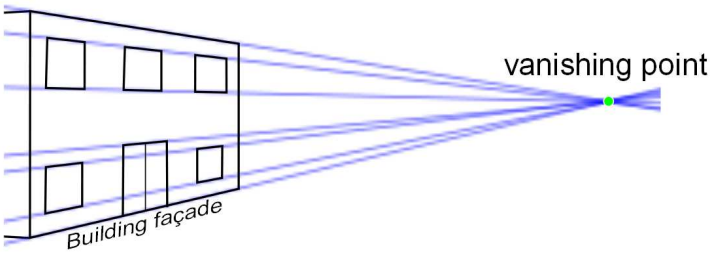

Figure 3.7: *Vanishing point example – The figure shows how lines on a building intersect at a vanishing point.*

groups the parallel lines within the original scene by identifying clusters of lines in the image that have the same or close vanishing point.
The input to the line grouping is the horizontal lines from section 3.2 which have been tilt rectified using the $\mathbf{H}_{tilt}$ homography from section 3.3. The steps involved in line grouping are displayed in Fig. 3.8.
The rectified image is first equally divided into a number of vertical strips. If the strips are too big we might have façades with different directions and if they are too small we might not find any segmenting strips to be able to identify direction. As a result there is a trade off between the number of strips and the precision of the correspondence to a single wall. Empirically 13 strips have been found to be the most effective. For each strip, the RANSAC [FB81] algorithm is used to find the coordinates of the dominant vanishing point for the lines contained within it. RANSAC stands for "RANdom SAmple Consensus" and is an iterative method for estimating the underlying dominant model in a set of data which can contain outliers. In order to estimate a vanishing point

, the technique first selects two lines at random. The intersection of the chosen lines creates a putative horizontal vanishing point. To evaluate the likelihood of this vanishing point, the other lines that agree with (i.e. are coincident with) this hypothesized vanishing point are counted. This set of lines is know as the consensus set. This process is repeated over multiple iterations. Once the largest consensus set is found the concurrent point is estimated using least squares fitting and chosen as the vanishing point for that strip.

Once a vanishing point for each strip had been determined a process of mean shift clustering is used on each of the vanishing directions computed. Expectation maximization iteratively estimates the coordinates of vanishing points as well as the probability of an individual line segment belonging to a particular vanishing direction. The application of the line grouping process being applied to a shop front is shown in Fig. 3.9. This is a single plane image where the lines can be seen converging to a vanishing point on the left(highlighted by the red dot). Fig. 3.10 shows the process being applied to an image where multiple planes are visible. As can be seen, the line grouping correctly identifies the two vanishing points for each of the visible façades.

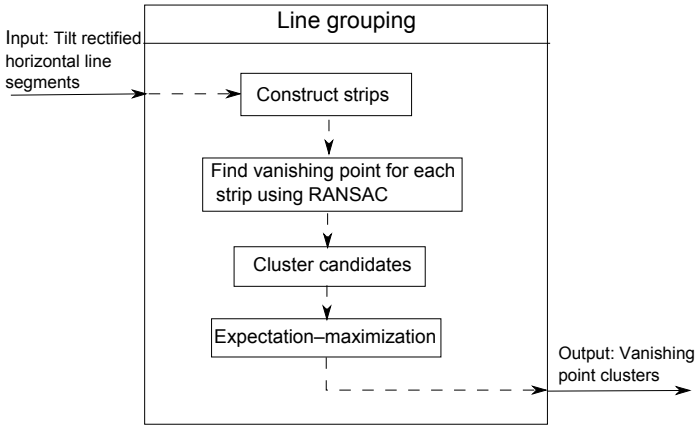

Figure 3.8: *Line grouping flow diagram – The figure shows the flow of each of the steps in the camera line grouping step.*

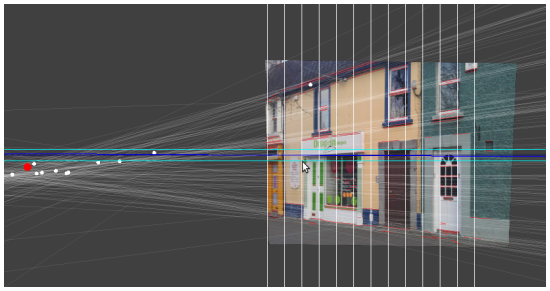

Figure 3.9: *Line Grouping – The figure shows the line grouping where parallel lines are converging at a vanishing point*

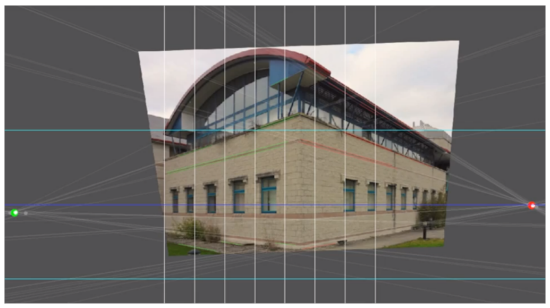

Figure 3.10: *Line Grouping Result 2 – The figure shows the line grouping where two planes are visible*

3.5 Layout Extraction

The main aim of the layout extraction step is to estimate the layout for the 3D structure of the building in the input image. This is achieved by generating a set of possible layouts. These layouts can have up to three façades. These possible layouts are then evaluated to find the most fitting for the scene. When this is computed we have extracted the planar structure from the image.
To do this it is assumed that the buildings have vivid enough vertical boundaries. Layout models that correspond to rectilinear structures in the scene are generated in a cascade manner by using the vertical lines extracted in Section 3.2. To generate these models the image is first partitioned into vertical strips. The leftmost and rightmost vertical lines are selected. This is to generate the model containing one single dominant

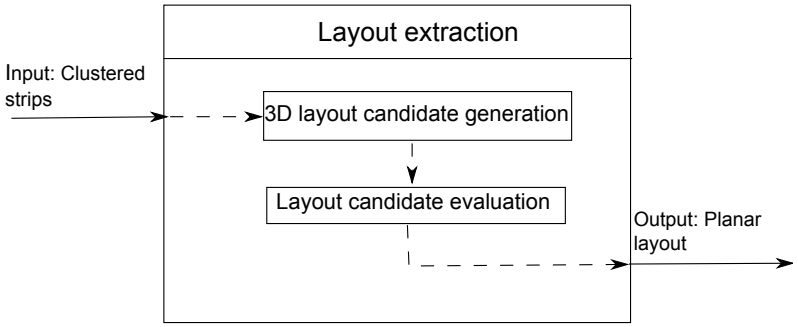

Figure 3.11: *Layout extraction - The figure shows the flow of each of the steps in the layout extraction*

3D plane. In turn this can be used to generate a piecewise planar model for a scene containing multiple 3D planes.
This is done by enumerating through the possible layouts to fit the scene for up to three possible planes. These layouts can then be evaluated in an attempt to a acquire the most accurate match for the scene. The line segment from a horizontal vanishing direction provides a strong indication of the existence of a 3D plane in its direction. After generating a number of layout models based on the extracted vertical lines, we evaluate how well each one fits the collection of horizontal line segments. The best fitting model is chosen to describe the underlying 3D geometry of the imaged scene.
Consider L_x is the candidate model which contains planes. Accordingly the image will be divided into x vertical image strips $S = \{s_1, s_2, ..s_x\}$. For a strip S_k, the supporting score for it belonging to a vanishing direction is computed as:

$$f\left(V_i, s_k\right) = \frac{\sum_{l_j \in C(V_i, s_k)} \mid l_j \mid}{\sum_{l_j \in C(s_k)} \mid l_j \mid} \tag{3.8}$$

where $C\left(S_k\right)$ is the set of line segments contained within the strip s_k, $C\left(V_i, s_k\right)$ is the set of lines belonging to the vanishing direction within the strip , and l_j denotes the length of the line. The direction V_k^* with the maximum supporting score will be assigned to this strip. Then the fitting score for this layout candidate is computed as:

$$F\left(L_x\right) = \sum_{s_k \in S} f\left(V_k^*, s_k\right) * AREA\left(s_k\right) \tag{3.9}$$

where, $AREA(s_k)$ is the area percentage of vertical strip s_k in the image. The model which produces the highest fitting score will be chosen to describe the layout of the scene. In practice, if the fitting score does not increase significantly (0.1 in our implementation) after adding more planes, we use the model of fewer planes for better efficiency. Fig. 3.12 shows the final result of image segmentation, in which the plane is correctly identified and the extracted planes can be seen on the right.

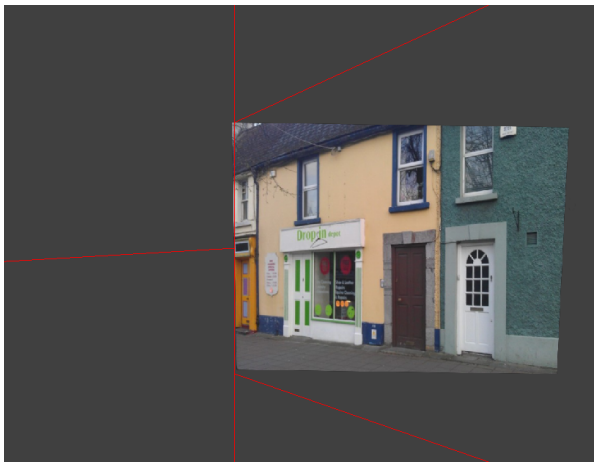

Figure 3.12: *The figure shows the planar layout of the being clearly identified*

In the Fig. 3.13 the both planes of the building are identified correctly.

Figure 3.13: *Layout extraction result – The figure shows the layouts for multiple planes being identified*

3.6 Homography estimation

Once the planes have been extracted they can be used to generate a 3×3 homography which will translate from the image to each of the planar façades of the buildings within the original image (see Fig. 3.14). This

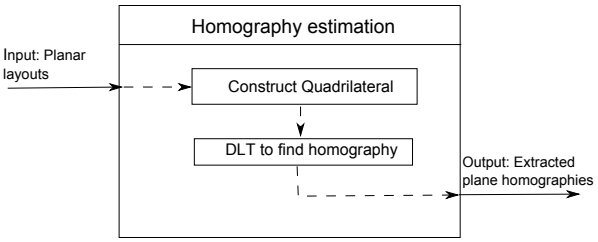

Figure 3.14: *Homography estimation overview – The figure shows an overview of the process of creating a homography from the extracted planes*

homography is utilised when creating AR content and is explained in greater detail in Chapter 4. This is also used for the creation of a front parallel view which is part of the AR authoring in Chapter 5.

3.7 Limitations of the technique

The system is effective when used on planer structures however an issue arises on buildings that have partially non planar façades or completely non planar façades. Any curved area of the building will be treated by the technique as if it had a planar profile. The system will not be able to augment correctly onto non planar areas of the building however it will work for any planar parts of the façade. An example of this can be seen in Fig. 3.15. In this the rounded centre of the building is treated as planar. This scene also shows that the planar extraction is still effective in the presence of occlusions. In the left figure it can be seen the effects of the trees and statues are removed by the line extraction and not used then in the subsequent processing on the right.

Figure 3.15: *Example of a scene with occlusions and a non planar façade.*

While the system can still work in the presence of occlusions the façade must be the most prominent aspect of the image. One of the cases where the system fails is when this is not the case. This is illustrated in Fig. 3.16. In this image the façade of the building is not the main focus of the image and the planar extraction algorithm completely fails to pick up the façade.
Another case where the planar extraction has been unable to pick up the façades can be seen in Fig. 3.17. This image is of two opposite corners

Figure 3.16: *Scene where planar extraction fails due to not enough focus being on the façade*

of two buildings. The algorithm unable to distinguish the middle area of the image because there no building façade. It is then left with two conflicting façades of each side of the image which there is no building layout. It may be possible to detect this building configuration and account for it in the future.

Figure 3.17: *Scene where planar extraction fails due to multiple conflicting façades. The image on the left show the tilt rectified image with the extracted lines. In the right image the red lines show the plane that has been extracted which consists of only the two camera facing facçades while the other façades are ignored.*

4 Façade based AR

4.1 Overview

The previous chapter provided details of the method used for the extraction of the planar structures from buildings within urban settings. This in turn gives a simplified 3D model of a building structure. One of the major contributions of this thesis, which is described in this chapter, is to build upon the the planar extraction step in order to create augmented content. The recovered planar structures provides a robust 3D frame of reference, which can be used to integrate 3D virtual content with the original scene. There are two steps to this process:

1. **Camera pose estimation** takes the homographies from the planar extraction and computes the camera pose which is needed to overlay augmented content on the building planes.

2. **Virtual camera rendering** uses the computed camera pose to augment virtual 3D content into the original image.

The camera pose estimation makes use of the extracted planes to estimate the extrinsic matrix of the camera. The estimated extrinsic matrix is used to set up a virtual camera to render the augmented content. 3D virtual content can then be combined with the original image to create a composite view. The overlaid content will give the effect of inserting

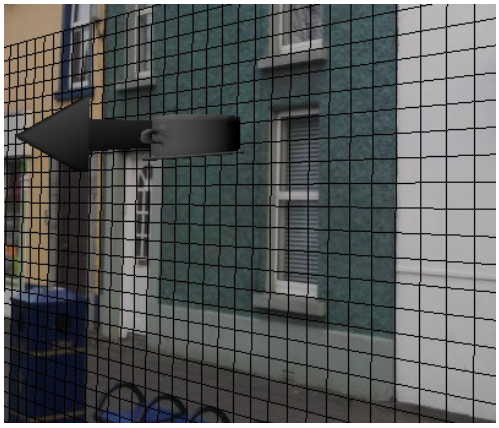

Figure 4.1: *Façade AR example – The figure shows the extracted plane (which is highlighted using a wire grid) being augmented with a virtual arrow and teacup.*

3D objects into the scene as if they were originally part of the real world scene, see Fig. 4.1.

4.2 Camera Pose Estimation

The planar extraction outputs directly feeds into the camera pose estimation. This provides a simplified model of the structure of the scene and can then be used to extract the camera orientation and translation. In particular, from the planar extraction there is a homography $\mathbf{H}_{fa}$ for each of the façades extracted from the scene. This homograhy $\mathbf{H}_{fa}$ maps the image of the façade to a front-parallel view which we call a *planelet*. Using the inverse of the homography associated with the façade, $\mathbf{H}_{fa}^{-1}$, it is possible to recover the exterior orientation of the camera relative to the plane.

Each of the recovered planes from the planar extraction step are set as the $z = 0$ plane. Four line segments are chosen within each of the extracted image strips that correspond to a plane. Two lines are chosen from the vertical direction $\overline{AB}$,$\overline{CD}$ and another two are chosen from the horizontal direction $\overline{AD}$, $\overline{BC}$. The intersection points of these lines are then computed. Since it is assumed they are coplanar A, B, C, D are four points belonging on a façade which form a rectangle with the following homogeneous world coordinates:

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & s.h & s.h \\ 0 & h & h & 0 \\ 0 & 0 & 0 & 0 \\ \underbrace{1}_{\mathbf{A}} & \underbrace{1}_{\mathbf{B}} & \underbrace{1}_{\mathbf{C}} & \underbrace{1}_{\mathbf{D}} \end{bmatrix} \tag{4.1}$$

where h is the height of the 3D rectangle, s is the ratio between the width and the height and $s.h$ is the (unknown) width of the rectangle. As can be seen, each point corresponds to an individual column of the matrix. This rectangular structure and the relationship with the homography H_{fa} given by the planar extraction can be seen in Fig. 4.2.

Let $\mathbf{x}$ be the matrix of the known 2D homogeneous coordinates of the projections of $\mathbf{X}$ into the image. Their projections into the image are then defined as follows:

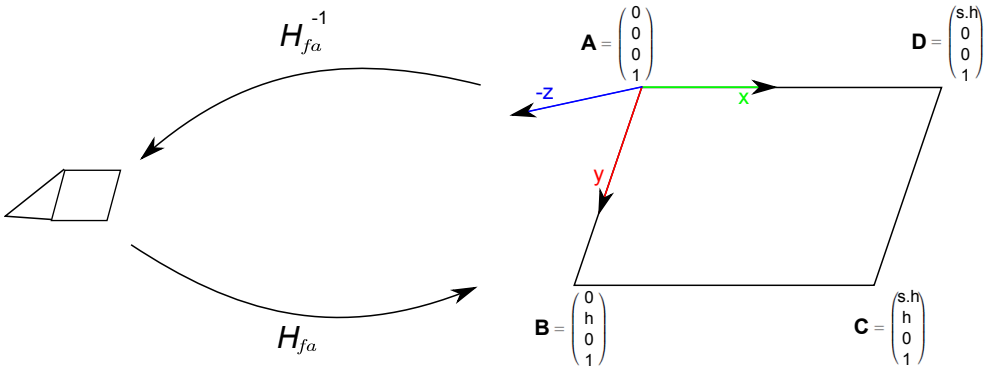

Figure 4.2: *Façade quadrilateral – The figure shows the relationship between the façade and the image.*

$$
\begin{aligned}
\mathbf{x} &= \mathbf{P}\mathbf{X} \\
&= \underbrace{\mathbf{K}[\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3\mathbf{t}]}_{\mathbf{H}_{fa}^{-1}}\mathbf{X} \\
&= \mathbf{K}[\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3\mathbf{t}]\begin{bmatrix} 0 & 0 & s.h & s.h \\ 0 & h & h & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \\
&= \mathbf{H}_s \begin{bmatrix} 0 & 0 & h & h \\ 0 & h & h & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}
\end{aligned}
\tag{4.2}
$$

where $\mathbf{P}$ is the 3×4 camera projection matrix and $\mathbf{H}_s$ is the 3×3 planar homography which transforms a square to a quadrilateral patch in the 2D image. The image coordinates of the four corners of the quadrilateral $\mathbf{H}_s$ can be solved in a closed form as follows. Since $\mathbf{R}_1$ and $\mathbf{R}_2$ are columns of a rotation matrix and should have unit normal, the aspect ratio s can be recovered as follows:

$$
s = \frac{\left\| \tilde{\mathbf{H}}_s^1 \right\|}{\left\| \tilde{\mathbf{H}}_s^2 \right\|} \tag{4.3}
$$

where $\tilde{\mathbf{H}}_s^1$ and $\tilde{\mathbf{H}}_s^2$ are the first and second columns of matrix $\mathbf{K}^{-1}\mathbf{H}_s$.

Using the value of s from eq. 4.3, the six-degree of freedom camera pose including three degrees of freedom for translation and three for orientation can be estimated as follows:

$$\mathbf{R}_1 = \frac{\tilde{\mathbf{H}}_s^{\ 1}}{s} \tag{4.4}$$

$$\mathbf{R}_2 = \tilde{\mathbf{H}}_s^{\ 2} \tag{4.5}$$

$$\mathbf{R}_3 = \mathbf{R}_1 \times \mathbf{R}_2 \tag{4.6}$$

$$\mathbf{t} = \tilde{\mathbf{H}}_s^{\ 3} \tag{4.7}$$

Note the camera pose with respect to a 3D scene plane is automatically computed given a single-view image. Furthermore we can compute the camera position for other image frames, using only the frame-to-frame homographies between multiple-views.

Figure 4.3: *Façade AR example – The original image is on the left and an teapot and arrow are augmentated into the scene on the right.*

4.3 OpenGL Rendering

In order to generate an augmented view we make use of the OpenGL framework to create a virtual camera with the same parameters as the real world camera (i.e. calculated as detailed in Section 4.2). In this way 3D objects that are placed into the virtual scene will be relative to the extracted planes in the original image. When this virtual scene is then overlaid on the original image, the virtual objects will have a consistent alignment with the extracted planes in the original image.
The difficulty here is that the mapping between the real camera parameters and the OpenGL virtual camera parameters is not a direct one. Both OpenGL cameras and real world cameras use a different pipeline in order to represent a projection to an image. In order to set up a virtual camera to use the parameters for rendering the virtual scene we must first do some conversions on both the intrinsic and extrinsic matrices of the real camera. To do this we must work through each section of the OpenGL pipeline (shown in Fig. 4.4) and compute its corresponding parameters in the real world camera model.

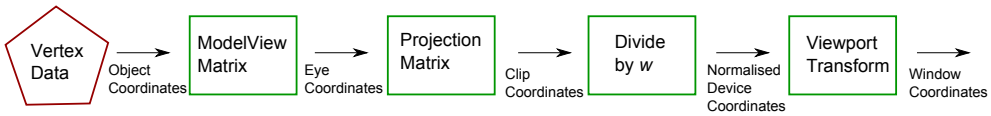

Figure 4.4: *OpenGL Pipeline – The figure shows the pipeline for OpenGL.*

The pipeline has three major elements which we must consider

1. The Model View Matrix.
2. The Projection Matrix.
3. The ViewPort Transform.

4.3.1 OpenGL Model View Matrix

In OpenGL the Model view matrix corresponds to the extrinsic parameters of a real camera. This transforms a 3D vertex within the eye coordinates to the cameras local coordinate system. The view matrix defines the position of the camera in the OpenGL 3D world and is built from the extrinsic matrix $[\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3\mathbf{t}]$. The model matrix is defined for each AR widget. It defines their position in the 3D world and it is given by the coordinates in planelet space. The model view is a 4×4 matrix where the 4th row is set to $(0, 0, 0, 1)$.

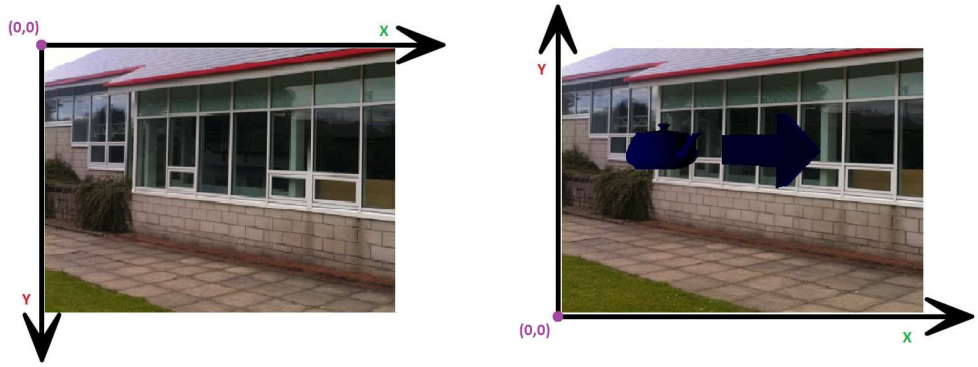

Figure 4.5: *Coordinate Comparison Example – The figure shows a side by side example of the coordinate system used by a real camera(Left) and the virtual 3*D *camera of OpenGL (Right)*

One point that is important to consider is that OpenGL has a different pixel coordinate system than a real image. This is due to the origin of an OpenGL scene being the bottom left as opposed to top left in real image. This difference of coordinate systems can be seen in Fig. 4.5. In order to correct for this the point p on a real image can be converted to OpenGL coordinates by

$$P = height - p \tag{4.8}$$

where P is the same point in an OpenGL scene. Within OpenGL the viewing direction is the opposite of the real camera model describes in

Chapter 1. This means the OpenGL camera model has the -Z direction as the viewing direction. OpenGL also has the positive Y direction as the up vector for the virtual camera. This can be seen in Fig. 4.6 In order to make these directions line up we first multiply by a scaling matrix $(1, -1, -1, 1)$

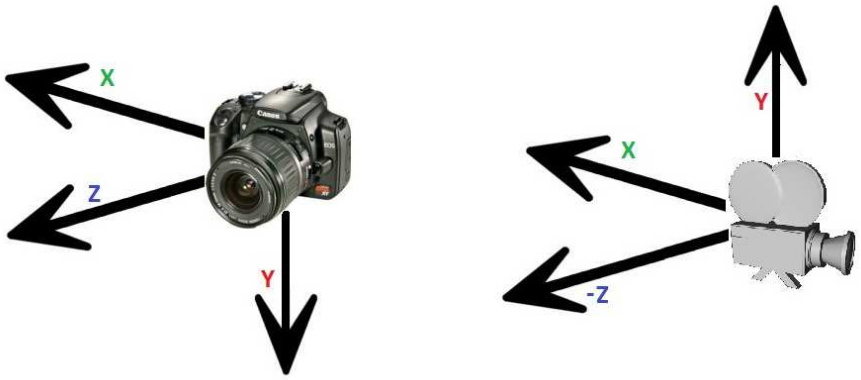

Figure 4.6: *Camera Comparison Example – The figure shows a side by side example of the real(left) and virtual camera(right) viewing directions*

4.3.2 OpenGL Projection Matrix

The OpenGL projection matrix will transform a vertex from the eye coordinate system to the clip coordinates. This matrix is shown in Eq.

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} \frac{cot\left(\frac{fov_y}{2}\right)}{aspect} & 0 & 0 & 0 \\ 0 & \frac{fov_y}{2} & 0 & 0 \\ 0 & \frac{-clip_{far}+clip_{near}}{clip_{near}-clip_{far}} & \frac{-2*clip_{far}*clip_{near}}{clip_{near}-clip_{far}} & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{4.9}$$

where fov_y is the field of view in the y direction and $aspect$ is the aspect ratio of the scene. Both $clip_{far}$ and $clip_{near}$ are values to determine what is inside or outside the clipping region. For our purpose we do not need to worry about the perspective division because OpenGL handles this internally in the graphics hardware. We need to provide the field of view given by

$$fov = 2\tan^{-1}(\frac{h_p}{2f}) \tag{4.10}$$

where h_p is the height of the planelet in pixels and where f is the focal length in pixels (known from the metadata and the specifications of the camera/device).

4.3.3 OpenGL Viewport Transformation

Finally, The Viewport Transform is used to define the rectangle of the rendering area where the final image is mapped. This transforms the vertex in clip coordinate system to window coordinates.

$$\begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} = \begin{pmatrix} \frac{width}{2} & 0 & \frac{width}{2} + x0 & \\ 0 & \frac{height}{2} & \frac{height}{2} + x0 & \\ 0 & 0 & & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{4.11}$$

4.4 Results

In order to demonstrate this method 3D graphical elements were augmented into various scenes of different complexity. Some examples of applying this method to the images from ZuBUD are shown in Fig 4.7. The top left image illustrates the quadrilateral being augmented onto both façades of the building. The image on the top left and bottom right show a directional arrow being placed within each scene. The image on the bottom right shows the algorithm working in spite of vertical tilt.
To show the robustness of this method we also applied the technique to images captured in the town of Maynooth, Ireland. Fig. 4.8 shows a set

Figure 4.7: *Example results of image augmentation.*

of images of a game shop captured by a hand-held camera from different viewpoints. The images on the right show the input image with an arrow added above the store to provide the user with directional information. As can be seen from the figure, the technique provides an accurate and stable rendering over considerable change in viewpoint.

In Fig. 4.9 we can see on the left is the original images and on the right directional information has been augmented into the scene. In these a wire grid is augmented into the scene to highlight the extracted plane.

4.5 Limitations of technique

The method of augmenting urban scenes works well on scenes that contain planar structures. Furthermore, although the technique is not designed for non-planar façades, as demonstrated in [CM12], the technique will approximate façades of low curvature as piecewise planar. That is, it will subdivide such façades in separate but contiguous planar components. In such instances the façade based AR approach described in this chapter is equally applicable.
With this said, we have found that in situations where the façade geometry is irregular or contains regions of high curvature this model does fail. Another failure mode of the system can also arise due to extraneous linear features in the scene e.g. lines on the grounds plane, or large numbers of parallel over head power lines. It is noted that the failure states for the façade based AR is highly dependant on the planar extraction discussed in Section 3.7.

4.6 Conclusions

This chapter detailed the steps involved in creating augmented content using the homographies from the planar façade extraction technique explained in Chapter 3. Once the camera pose has been estimated this is used to set up the virtual camera and augment 3D content into the original image. The simplified planar façade model is used as a 3D frame of reference for augmenting the scene. The planar method of creating AR content forms the basis for the full AR system described in Chapter 5

Figure 4.8: *Multi-angle planar AR Example – Example results of image augmentation from multiple angles*

Figure 4.9: *AR showing plane – Example of AR showing the wired grid for the extracted plane.*

5 AR system for urban navigation

5.1 Overview

This chapter presents a mobile augmented reality system for positioning and navigation within urban environments. With this system a user can take an arbitrary image of an urban scene with their mobile device. This image along with some meta-data is then sent off to a server which matches it against a geo-referenced image database.

This database stores linked images of planar façades along with their associated navigation and point of interest (POI) data. The system uses the façade based AR technique discussed in Chapter 4 in order to estimate the camera pose. Camera pose data and the relevant navigation data is then sent back to the client on the mobile phone. With this information the client can then augment the query image of the user with the stored 3D landmark information and navigation data.

The traditional computer vision approach for this type of system would be to create a full 3D representation for the area that is to be augmented. Our system proposed on the other hand in this thesis is based on the assumption that images are taken within urban environments and that many of the buildings façades within urban settings can be approximated using planes. As a result the system only needs to store a simplified piecewise planar representation for each façade instead of a full 3D representation. This allows for a much more scalable augmented

reality system.

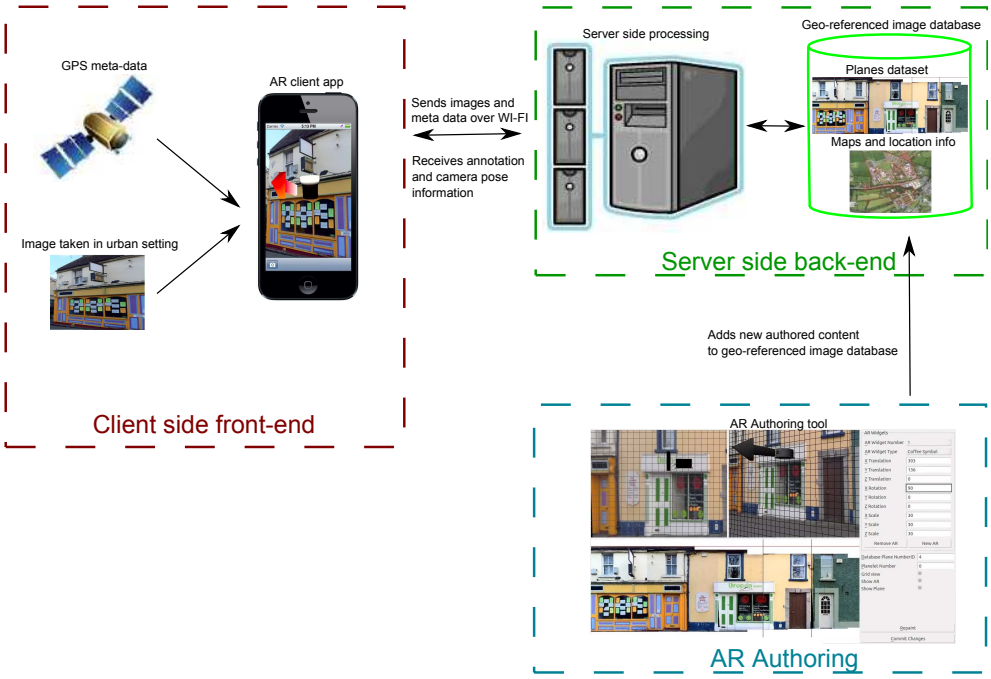

Figure 5.1: *AR System Overview - The figure shows the AR System Architecture*

The goal of this is to provide a planar façade based augmented reality system for images taken in urban environments. This system uses a client-server architecture which is based around the following three components:

- ***System front-end*** – The client side component can take a picture of a building in an urban setting. This is sent to the server back-end for processing and will return the estimate camera pose information along with the 3D AR widgets. The client can then augment the view using this information.
- ***System back-end*** – The server side component of the AR system takes an image sent by the client and performs the façade extraction step on it. The server then generates a homography mapping

the image to what we refer to as a *planelet* for each of the extracted façades. A planelet is defined as a viewpoint normalised image of the extracted façade where parallel lines of the façade appear parallel in the normalised image. The system will then match each of the extracted planelets against a database of planes as explained in Chapter 4. A plane represents the union of contiguously overlapping planelets. Once matched against a plane this can be used to localise the image and get the previously stored 3D data associated with that location. The parameters defining the planelets and navigation information associated with the scene are then sent back to the AR client on the mobile device for display.

- ***AR authoring*** – In order to add new content to the geo-referenced image database the system also requires an authoring interface. This tool uses the plane representation to act as a frame of reference for authoring: the (x, y) 2D coordinate system of the normalised image is extended to a 3D coordinate system by using the normal of the plane as the third dimension z. This 3D space is then used to localise objects within the scene. The authoring is performed by dragging and dropping 3D content relative to these set of planelets. These can be then be saved to a database with their associated planelets.

5.2 System front-end

The front-end to the AR system is the mobile AR application running on an iPhone. This application displays a see through camera view and uses a point and shoot method for acquiring images of urban scenes. Theses images can then be augmented with relevant 3D data.
Images taken are tagged by the client with meta-data which includes the focal length of the camera, timestamp and GPS coordinates. This image along with the associated meta-data is sent to the server back-end for processing. On the server the image is matched against a database to retrieve widgets which have been previously attached to a given façade. The server side matching process is described fully in section 5.3. Once

the processing is completed the camera pose and 3D widget information is sent back to the client. This visualisation application creates an augmented view using OpenGL to overlay the virtual AR widgets onto the urban environment scene with a consistent alignment. This is done using the planar AR technique detailed in Chapter 4. The front-end application augmenting a building can be seen in Fig. 5.2.

Figure 5.2: *Front End Application - The figure shows the AR mobile interface where the façade has been augmented with an arrow pointing at the direction of the closest pub.*

5.3 System back-end

The back-end infrastructure performs the image recognition and pose estimation for an input image. This is run on the server side so as to offset tasks that are too computationally expensive to run on the mobile client and to facilitate sharing of information between users. This back-end service has two main functions:

1. ***Populating the image database*** – This step takes a new image to be added and matches it against the current dataset. If this is successful the new image is used to extend the database plane to which it is matched to. If it is unable to find a match. This image will become the basis for a new plane in the database.

2. ***Image data lookup in the database*** – This service accepts incoming images from the front-end client, matches them against a database in order to localise and find the appropriate navigation data and calculates the camera pose.

Both populating the image database and the image data lookup require that the image be matched against the database first. A sequence diagram for the system back-end is shown in Fig. 5.3. There are three main stages in order to identify if the image has a match in the database:

1. ***Façade Extraction*** – This extracts the façades of the building within the image. The extracted façade is then used to create a front-parallel view we call a *planelet*.

2. ***Bag-of-words filtering*** – The bag-of-words [GLT11] filtering reduces the number of image candidates to be checked in the planelet matching.

3. ***Planelet matching*** – The planelet matching identifies if there is a plane within the database that the extracted façade matches to.

5.3.1 Façade extraction

The façade extraction is the first step performed when an image is to be processed. The server extracts each façade within the image. The façade

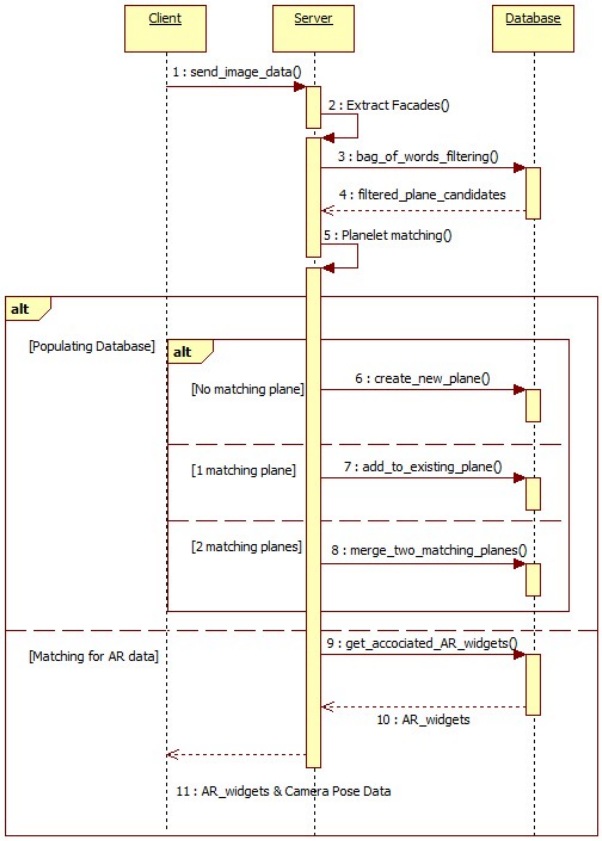

Figure 5.3: *Back-end sequence diagram – This shows the interactions between components for the back-end system.*

extraction algorithm detects vanishing points to find dominant planes. For each detected façade fa, a homograhy $\mathbf{H}_{fa}$ maps the image of the façade to a front-parallel view we call a *planelet*:

$$\tilde{\mathbf{x}} = \mathbf{H}_{fa}\mathbf{x} \tag{5.1}$$

where $\mathbf{x}$ is the vector of the homogeneous coordinates of a point in the original image within the façade fa and, where $\tilde{\mathbf{x}}$ is the vector of the homogeneous coordinates of its corresponding point in the planelet. Given the homography associated with the façade, $\mathbf{H}_{fa}^{-1}$, it is possible to recover the exterior orientation of the camera relative to the plane. This step is fully detailed in Chapter 3.

During the façade extraction step, a planelet model is used to represent the building façades within the query image. The planelet representation is important when it comes to matching for two main reasons:

- Firstly since the planelets are viewpoint normalized, there is an affine transformation between the two planelets that are to be matched. This is used to geometrically constrain the matching process and as a result leads to a more robust and accurate matching process [CM12].

- Secondly if we know this affine transformation between two planelets, then we know their relative positions and we can easily build a representation of an entire street. This is what allows the system to build a *plane* representation for a street. The origin of the first planelet added to a plane is used as the origin of the plane. For adding to the database it is these planes that the database is populated with. After the database has been populated this representation provides a coordinate system in which we can place data associated to the scene e.g. the sales prices of a shop at some coordinates within the plane. The calculations involved in this are further detailed in Section 5.3.2.

5.3.2 Bag-of-words filtering

Once the planelets have been extracted from the image the system begins matching by performing a prefiltering step using the GPS coordinates of the images to select only nearby planes. This provides a mechanism to avoid having to compute the matching against all planes stored in the database. In order to match the image the with the database we make use of the bag-of-words [GLT11] technique.
To compute the bag-of-words descriptor for a planelet the system first computes a set of feature points and descriptors using the FAST feature detector [RD06b] and ORB feature descriptor [RRKB11]. The ORB algorithm uses FAST in pyramids to detect stable keypoints. The strongest features are selected using FAST. Their orientation is then found using first-order moments and then the descriptors are computed using

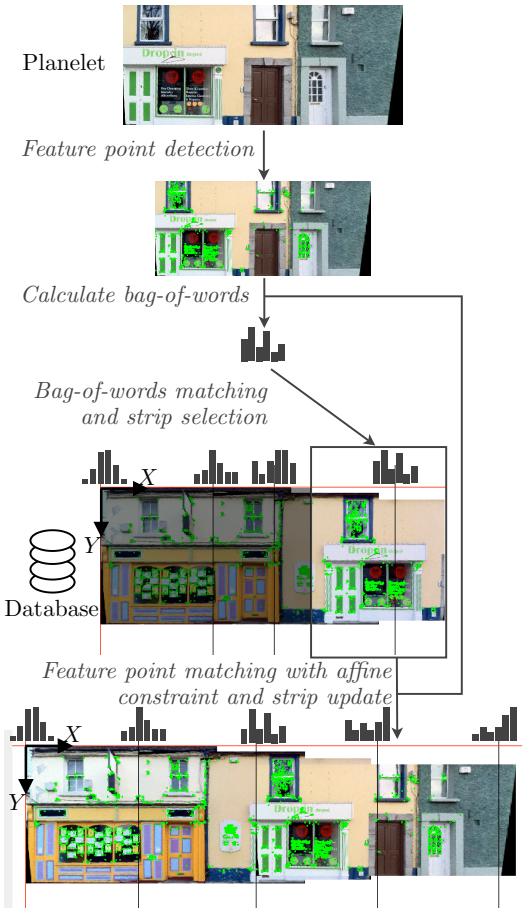

Figure 5.4: *System Matching – Matching process for computing planar façade mosaics.*

BRIEF [MCF10]. This ORB algorithm was chosen due to the low computational requirements and because it gives similar or better results than the more conventionally used SIFT [Low04] or SURF [BTG06] approaches. Since the system uses viewpoint rectified planelets it is also not necessary to compute for rotation normalization since there should be no rotational changes between two planelets from the same façade. To compute the bag-of-words from the feature point descriptor, we assume we have built a vocabulary of words from the feature point descriptors of a training set. This is done by using a large training set and computing

the features for each image in the training set. These features are then clustered into a codeword which is a representation of several similar feature patches throughout the training data. Now the bag-of-words is a histogram measuring the occurrence in the planelet of each word from the vocabulary. This occurrence value is weighted by the rarity of each word. The more infrequent a word is, the more suitable this word is for describing the planelet.
The bag-of-words technique is used to quickly select subsets of planes (we call *strips*) as candidates. The bag-of-words is a single numerical vector which provides a global description of that strip. The bag-of-words are computed and stored for each strip in the database. When a planelet is submitted for matching its bag-of-words descriptor is computed and compared against the one in the database. This can be seen in Fig. 5.4 where X and Y are the horizontal and vertical axis for the database plane respectively. The origin is set to the top left of the first planelet added to the database for that plane.

5.3.3 Planelet matching

The planelet matching step consists of matching the query planelet to one of the planes from the bag-of-words filtering. During the bag-of-words filtering the feature points' for the query planelet have already been computed. Let $\mathbf{X}_{planelet}$ be the matrix of these feature points homogeneous coordinates in the coordinate system of the planelet. For each strip from the bag-of-words filtering, the system finds the matrix of the homogeneous coordinates of the feature points in the coordinate system of the plane. Let this be $\mathbf{X}_{plane}$. The matching between $\mathbf{X}_{plane}$ and $\mathbf{X}_{planelet}$ is expressed purely by a translation and a scale change since these are both viewpoint normalized, shown by:

$$\mathbf{X}_{plane} = \underbrace{\begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{T}} \mathbf{X}_{planelet} \tag{5.2}$$

where we need to estimate the parameters of $\mathbf{T}$.
To estimate $\mathbf{T}$, for each feature point from the planelet the system measures the Hamming distance between the ORB descriptor and the feature

points from the strip. When this distance is below a certain threshold this match is added to the set of putative matches.

These matches are then used to initiate the RANSAC algorithm which is applied in order to get the largest consensus set of feature point matches that satisfies the geometric constraint 5.2. The planelet matches the current plane when the best consensus set is greater than a threshold that was obtained through empirical testing. This will have a transformation $\mathbf{T}$.

5.3.4 Populating the image database

By populating the database we can build up a plane based representation for an entire street. This plane is then used to augment that street with 3D AR widgets.

The matching process described above determines first if a new image has a matching plane already within the database. If there is no match found for the current image then the extracted planelet is used to create a new plane. If there is a match with the database then the extracted planelet is used to update the matched plane and is concatenated with it (see Fig. 5.5).

If there are multiple planes that the planelet matches then the resulting constraint merges these planes into one. The planelet with the first best result is merged with the whole second plane. The planelets associated with this plane are then updated to belong to the first. This relationship is calculated as follows:

$$\mathbf{T}_{plane_1} = \mathbf{T}_1 \mathbf{T}_2^{-1} \mathbf{T}_{plane_2} \tag{5.3}$$

where $\mathbf{T}_{plane_2}$ is the geometric relationship of a given planelet to the second best plane. This can be seen in Fig. 5.6.

It is by running this process over a dataset of input images from urban scenes that a plane database can be built. The process of building a full plane for one side of a street can be seen in Fig. 5.7. Furthermore, the GPS coordinates of each street can be stored to build a geo-referenced database.

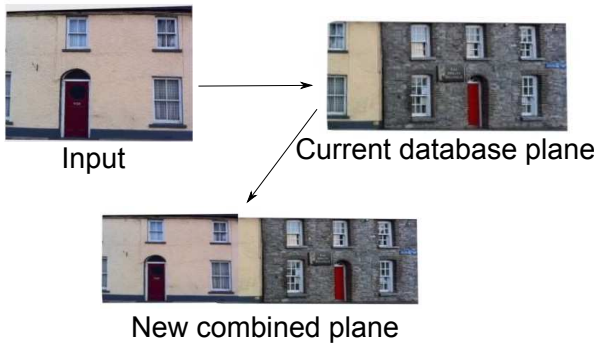

Figure 5.5: *Adding to Plane – The figure shows the combining of a new input planelet with an existing database plane. It is noted that the combined colors on the overlap zone do not match because this image is used only illustrate adding a plane and is not presented to a user.*

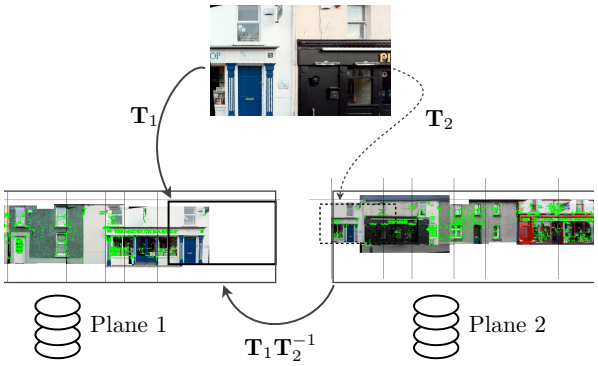

Figure 5.6: *Merging Planes – This figure shows the plane merging when a planelet matches two planes.*

5.3.5 Augmenting from the image database

Once the database has been populated with planes and authored with 3D widgets, a user can then augment a query image with those 3D widgets. When an input image has been matched against the database using the method described above the widgets associated with the matched plane are sent back to the visualisation client application. The full matching and AR widget retrieval process is illustrated in Fig. 5.8.

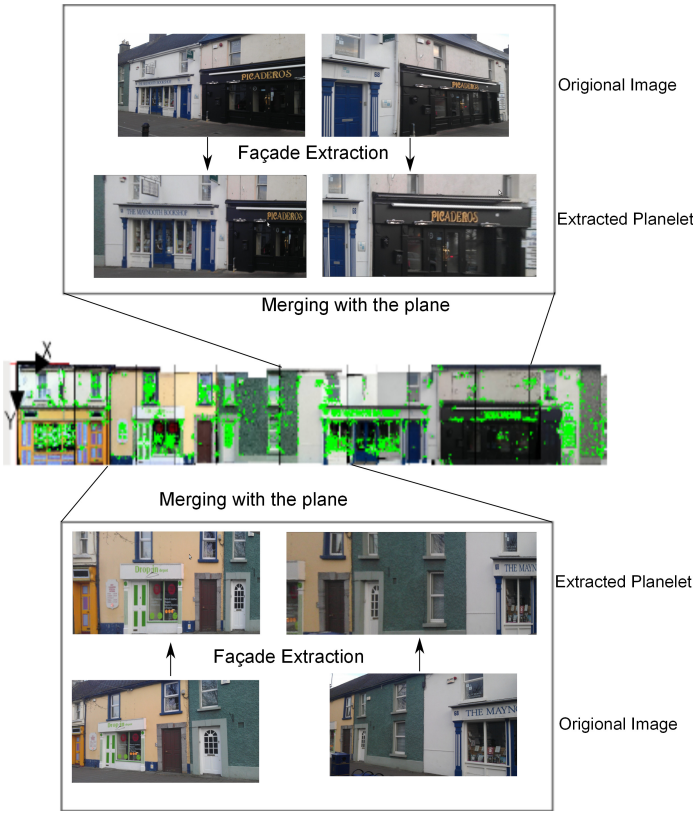

Figure 5.7: *Building a Plane – The figure shows multiple images being processed by the façade extraction. The extracted planelets are then merged to build a full plane for the street.*

5.4 Authoring System Content

The authoring tool provides a desktop application which allows a non-expert user to add new AR content to the system. Authoring consists of dragging and dropping 3D widgets into the front parallel view of the façade. The user can then set the position, orientation and scale of these widgets relative to the planelets. This permits a more simplified approach to authoring content without requiring a detailed global model of the scene. When the user is finished the system then uploads the 3D widgets with their associated position, scale and rotation stored relative to the matched plane in the database. These widgets can then be accessed later

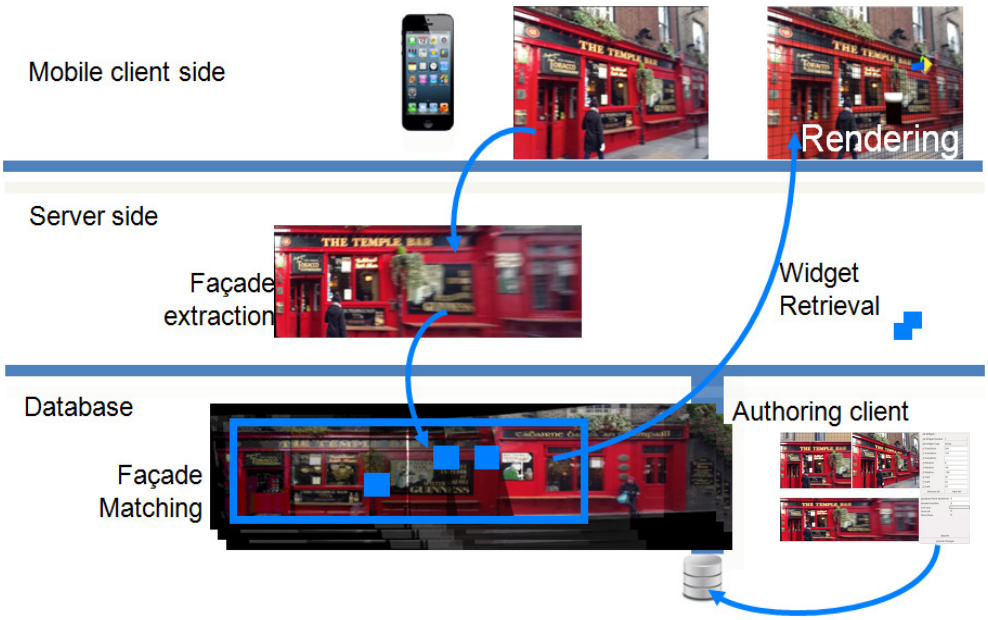

Figure 5.8: *Augmenting from Plane – The figure shows the process of retrieving 3*D *widgets from the database back-end and augmenting a query image.*

by the client when augmenting a query image.
The system uses predefined 3D models (3D widgets) for authoring. The set of models used currently in the system is set for navigation purposes but these can be changed or extended depending on the intended application for the system. For example:

- *Advertising* – If a shop owner wishes to advertise an in-store promotion. They can take an image of their shop front and load it into the authoring tool. The shop owner can then position an AR widget relative to their shop using the authoring tool. The promotion will now appear when a user of the client application is at the shop.

- *Social Media* – The authoring tool can be used socially by friends who wish to leave a comment about a particular location. Their comments can be added as widgets and their friends can then see it when they are at that location.

While these are just some examples of the uses of this type of system, it is also possible to allow a user to create and import their own set of widgets for their own application.

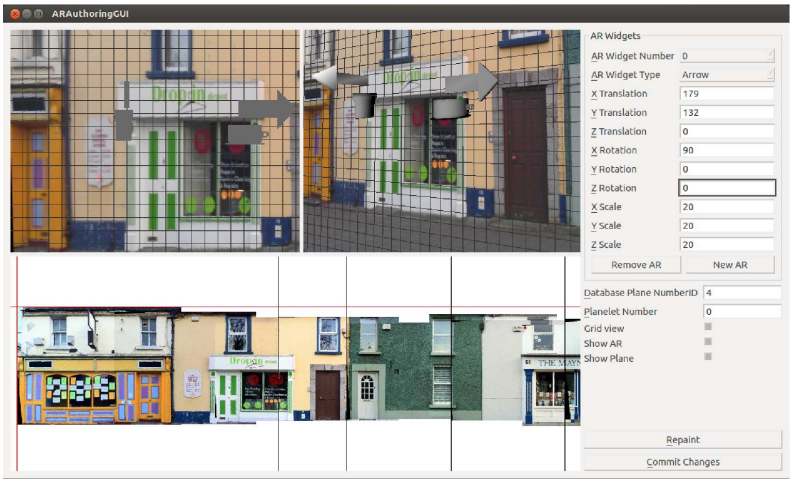

Figure 5.9: *The figure shows the AR desktop authoring interface*

The Authoring tool user interface can be seen in Fig. 5.9. The tool is divided into three screens:

- The top left is the authoring screen. This screen allows a user to add 3D content relative to the front parallel views of the façade extracted from the query image. First a user can select a widget from the panel on the right. They can then use their mouse to move and position the content in their desired location.

- The top right is the reviewing screen. This allows the user to view the 3D authored content re-projected back into their query image.

- The bottom screen gives the user a view of the overall plane that the extracted planelet from the query image has been matched to. When the 3D widgets are placed in a scene by a user any widgets can then be saved and stored relative to their plane. These can then be viewed using the AR client front-end.

There are 3 main functions of the authoring tool:

1. ***View AR widgets*** – Viewing previously added widgets is achieved by selecting an image of an area they wish to view. If there are multiple façades within the image, the user is asked to choose which façade they wish to view. The authoring tool then loads the AR widgets associated with that façade for viewing by the user.

2. ***Creating an AR widget*** – This is done by first selecting an image of an urban scene the user wishes to augment. The user can then add a new widget with their desired widget type which can be chosen from the selection tool on the right. Their new widget can then be translated, orientated and scaled using their mouse in the top left authoring screen. Once the user is happy with the result, their new widgets can be uploaded to the back-end database.

3. ***Add new planelets to the database*** – In order to add new planelets to the database the user again selects the image to be added. The tool will display the planelets extracted from the image which are available for addition to the database. The user can then pick which planelets they wish to add and upload their choice.

5.5 Conclusion

This chapter describes a mobile AR application where a user can take a picture and send it to the server where it is matched against a database to retrieve widgets previously attached to a given façade. The system has three main components:

1. The front-end application creates an augmented view using OpenGL to overlay the virtual AR widgets onto the urban environment scene with a consistent alignment.

2. The back-end system retrieves images from the front-end application, calculates the camera pose for the input image, matches against the database and finally retrieves the associated 3D widgets from the database.

3. The authoring tool provides a user friendly interface for the creation of new AR content by using the extracted façades to create a front parallel which in turn can be used as a frame of reference for authoring.

These three components provide a complete system for single image based urban AR. This system provides a platform for AR mobile application in urban environments. In this thesis, we demonstrated a navigational AR application. However, this platform is versatile and can be used for other applications such as advertising, tourism and social media.

6 Conclusion

In this thesis we have developed a state the art of augmented reality system focussed on mobile and markerless AR tracking applications in urban environments. Here we assume that the façades of a street can be approximated by planes which allows us to use a simplified map representation whereby the world is modelled as a set of local planar maps (i.e. for each contiguous façade) with only topological constraints between these local maps. The main contribution of this thesis is a method for augmenting images with 3D content using this planar building façade model of the environment.

A consequence of the façade based model is the fact that the 3D structure of the scene can be extracted from a single image. This technique exploits the often rectilinear structures found in urban settings permitting the recovery of the 3D geometry of the building façades. The advantages of this system include the fact that we do not at any stage create a complete 3D representation of the environment and therefore avoid the inherent complexities associated with this task.

The façade based representation also allows us to provide an authoring solution with an intuitive and easy to use interface for non-expert users. Here the local planar maps provide the XY-plane in a virtual 3D coordinate system, thereby allowing users to place 3D objects relative to a front parallel view of the extracted façades. To demonstrate this principal we have developed an authoring application which allows the user to load an arbitrary image of a street scene, and then apply the planar extraction algorithm to produce a linked visualisation of the input image, extracted façade, and the matched local planar map. Within this interface the user can select, position, and orientate AR graphical elements in full 3D relative to the scene. The resulting elements, including the positioning information, can then be stored centrally for later rendering

by other users.
The overall system detailed in the thesis employs a client-server based architecture. It allows a user to take an image using a mobile device which can then be sent to the back-end server for processing. This processing identifies and extracts the planar façades of buildings within the query image. These façades are then matched against a geo-referenced image database. Once matched the façades are then automatically related to associated 3D widgets of that plane. The matched façades are then used as a reference frame for augmenting the view using the 3D widgets associate with the query image's façade. Results of the application of this technique to real-world images of general urban environments have been shown.

6.1 Future Work

Currently the authoring solution is implemented as a desktop application with the AR viewer implemented on a mobile platform. We are currently moving our implementation of the authoring method to take advantage of HTML5. This will give the advantage of greater portability for the authoring system and most importantly allow the implementation to run on a wide variety of currently available mobile devices. This in turn will allow us to reach a wider group of non-expert users and facilitate open user testing.
We also wish to work on improving the system's image matching and localisation capability, and in particular incorporate more robust approaches to failure detection. While the focus has been on a system based on building façades, the approach will work for other planar surfaces e.g. indoor environments.
A separate issue addressed in [CM12] is the robustness of the planar extraction algorithm to non-planar façades. As shown in [CM12] in such situations the algorithm outputs a piecewise planar approximation of the façade's 3D geometry. Although this permits our system to augment AR content on such scene's, further work is required to evaluate the effect of this approximation on the user experience.
In the current system the paradigm employed is a point-and-shoot ap-

proach where the captured image is sent to the server for processing. The principal drawback of this approach is the latency due to the network transmission time. We are currently investigating a number of aspects of this issue including the possibility of moving the façade extraction algorithm to the mobile device through the use of the embedded GPU. We are also investigating the possibility of extending the planar extraction algorithm to allow tracking of the detected planes in an online fashion, thereby permitting live AR in the camera's video stream.
In the future we aim to fully deploy this approach as a vision based AR navigation system that will operate over a 1km^2 area of an urban environment. Here the user should be able to capture an arbitrary image within the environment and have the system graphically augment it with relevant navigation and point-of-interest (POI) information.

Bibliography

[ARN12] ARNav. Arnav. `http://arnav.eu/`, 2012.

[ART] Artoolkit overview image. `http://www.hitl.washington.edu/artoolkit/documentation/userarwork.htm`.

[Aut13] Autodesk. Ar-media plugin for autodesk 3ds max. `http://www.inglobetechnologies.com/en/new_products/arplugin_max/info.php`, 2013.

[Azu97] R.T. Azuma. A survey of augmented reality. *Presence-Teleoperators and Virtual Environments*, 6(4):355–385, 1997.

[BDR01] Y. Baillot, L. Davis, and J. Rolland. A survey of tracking technology for virtual environments. *Fundamentals of wearable computers and augumented reality*, page 67, 2001.

[Bed95] B. B. Bederson. Audio augmented reality: a prototype automated tour guide. *In Proc. of Conf. on Human Factors in Computing Systems (CHI)*, pages 210–211, 1995.

[BTG06] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. *In European Conference on Computer Vision*, pages 404–417, 2006.

[CAK93] M. Cohen, S. Aoki, and N. Koizumi. Augmented audio reality: Telepresence/vr hybrid acoustic environments. *Proc. of Workshop on Robot and Human Communication*, pages 361–4, 1993.

[CKM08] R. Castle, G. Klein, and D.W. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on*, pages 15–22, 2008.

[CM92] T.P. Caudell and D.W. Mizell. Augmented reality: An application of heads-up display technology to manual manufacturing processes. *Proc. IEEE Hawaii International Conference on Systems Sciences.*, pages 659–669, 1992.

[CM12] Y. Cao and J. McDonald. Improved feature extraction and matching in urban environments based on 3d viewpoint normalization. *Computer Vision and Image Understanding*, 116(1):86 – 101, 2012.

[DWB06] Hugh Durrant-Whyte and Tim Bailey. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *History*, pages 1–9, 2006.

[FB81] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *In Communications of the ACM*, 1981.

[FHP98] E. Foxlin, M. Harrington, and G. Pfeifer. Constellation: A wide-range wireless motion-tracking system for augmented reality and virtual set applications. pages 371–378, 1998.

[Fia04] M. Fiala. Artag revision 1, a fiducial marker system using digital techniques. *NRC Technical Report (NRC 47419), National Research Council of Canada*, 2004.

[Fia05a] M. Fiala. Artag, a fiducial marker system using digital techniques. *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.

[Fia05b] M. Fiala. Comparing artag and artoolkit plus fiducial marker systems. *IEEE International Workshop on Haptic Audio Visual Environments and their Applications, 2005.*, 2005.

[FNFB04] J. Fischer, M. Neff, D. Freudenstein, and D. Bartz. Medical augmented reality based on commercial image guided surgery. 2004.

[FSL05] F. Fritz, A. Susperregui, and M.T. Linaza. Enhancing cultural tourism experiences with augmented reality technologies. *The 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage VAST*, 2005.

[FSP+96] H. Fuchs, A. State, E. Pisano, W. Garrett, G. Hirota, M. Livingston, M. Whitton, and S. Pizer. Towards performing ultrasound-guided needle biopsies from within a head-mounted display. In *Visualization in Biomedical Computing*, pages 591–600. Springer, 1996.

[GAT10] GATech. UART - Unity AR Toolkit. `https://research.cc.gatech.edu/uart/`, 2010.

[GLT11] D. Galvez-Lopez and J. D. Tardos. Real-time loop detection with bags of binary words. In *InIEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 51—58, 2011.

[HFT+99] T. Höllerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway. Exploring mars: developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers & Graphics*, 23(6):779–785, 1999.

[HJT+04] Aki Härmä, Julia Jakka, Miikka Tikander, Matti Karjalainen, Tapio Lokki, Jarmo Hiipakka, and Gaëtan Lorho. Augmented reality audio for mobile and wearable appliances. *J. Audio Eng. Soc*, 52(6):618–639, 2004.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[KB99] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. *In Proc. IWAR '99, San Francisco,CA, October*, 1999.

[KCSC10] J. Kopf, B. Chen, R. Szeliski, and M. Cohen. Street slide: browsing street level imagery. *ACM Transactions on Graphics (TOG)*, 29(4):96, 2010.

[KD03] G. Klein and T. Drummond. Robust visual tracking for non-instrumented augmented reality. pages 113–122, October 2003.

[KHS89] K. Kelly, A. Heilbrun, and B. Stacks. Virtual reality; an interview with jaron lanier. *Whole Earth Review*, pages 108–120, 1989.

[KM07] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234, 2007.

[KO97] R. Kijima and T. Ojika. Transition between virtual environment and workstation environment with projective head mounted display. In *Virtual Reality Annual International Symposium, 1997., IEEE 1997*, pages 130–137. IEEE, 1997.

[KZ02] J. Kosecka and W. Zhang. Video compass. *European Conference on Computer Vision(ECCV)*, 2002.

[Lay11] Layar. Layar AR browser. `http://www.layar.com/browser/`, 2011.

[Led04] F. Ledermann. An authoring framework for augmented reality presentations. *TU Wien*, 2004.

[Leg11] Lego. Lego's augmented reality applicaiton at the intel developer forum. `http://www.engadget.com/2011/09/18/legos-augmented-reality-at-idf-eyes-on-video/`, 2011.

[LNBK04] G.A. Lee, C. Nelles, M. Billinghurst, and G.J. Kim. Immersive authoring of tangible augmented reality applications. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 172–181, 2004.

[Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *In International Journal of Computer Vision*, pages 91–110, 2004.

[LWMS12] T. Langlotz, D. Wagner, A. Mulloni, and D. Schmalstieg. Online creation of panoramic augmented reality annotations on mobile phones. *Pervasive Computing, IEEE*, 11(2):56–63, 2012.

[MBWF97] E. D. Mynatt, M. Back, R. Want, and R. Frederick. Audio aura: light-weight audio augmented reality. *Proc. of ACM UIST*, pages 211–12, 1997.

[MCF10] C. Strecha M. Calonder, V. Lepetit and P. Fua. Brief: Binary robust independent elementary features. In *In European Conference on Computer Vision*, 2010.

[MK94] P. Milgram and F. Kishino. A taxonomy of mixed reality visual displays. *IEICE Trans. Information Systems*, pages 1321–1329, 1994.

[MTS+05] K. Mikolajczyk, T. Tuytelaars, C. Schmid, a. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1-2):43–72, October 2005.

[Ols11] E. Olsen. Apriltag: A robust and flexible visual fiducial system. *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[Pap01] D. Pape. Virtual Reality Example Image. `http://www.flickr.com/photos/dpape/2720629446/in/photostream/`, 2001.

[Pha12] D. Pham. New research shows smartphone growth is global. `http://googleblog.blogspot.ie/2012/05/new-research-shows-smartphone-growth-is.html`, May 2012. Google Mobile Ads, Google.

[PTB+01] I. Poupyrev, D. Tan, M. Billinghurst, H. Kato, H. Regenbrecht, and N. Tetsutani. Tiles: A mixed reality authoring interface. In *INTERACT 2001 Conference on Human Computer Interaction*, pages 334–341, 2001.

[PYN98] J. Park, S. You, and U. Neumann. Natural feature tracking for extendible robust augmented realities. In *Proc. Int. Workshop on Augmented Reality*, 1998.

[RD06a] G. Reitmayr and T.W. Drummond. Going out: robust model-based tracking for outdoor augmented reality. In *Mixed and Augmented Reality, 2006. ISMAR 2006. IEEE/ACM International Symposium on*, pages 109–118. IEEE, 2006.

[RD06b] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *In European Conference on Computer Vision*, pages 430—443, 2006.

[Rek98] J. Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. *Proceedings of Asia Pacific Computer-Human Interaction (APCHI)*, 1998.

[RRKB11] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski. Orb: An efficient alternative to sift or surf. *In the International Conference on Computer Vision*, pages 2564–2571, 2011.

[SB97] J. Seagull and M. Beauer. A field usability evaluation of a wearable system. 1997.

[SL04] I. Skrypnyk and D.G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*, pages 110–119. IEEE, 2004.

[Son12] Sony. AR play augmented reality gaming on ps vita system. `http://us.playstation.com/psvita/apps/psvita-app-ar.html`, 2012. Sony PSP Vita.

[Str96] G.M. Stratton. Some preliminary experiments on vision without inversion of the retinal image. *Psychological review*, 1896.

[Str97] G.M. Stratton. Vision without inversion of the retinal image. *Psychological review*, 1897.

[Sut68] I.E. Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 757–764, 1968.

[Uni09] Unity3D. UnityAR - ARToolkit Interface. `http://forum.unity3d.com/threads/30817-UnityAR-ARToolkit-Interface`, 2009.

[Wik09] Wikitude. Wikitude. `http://www.wikitude.org/`, 2009.

[WLS08] D. Wagner, T. Langlotz, and D. Schmalstieg. Robust and unobtrusive marker tracking on mobile phones. *In IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR'08)*, 2008.

[WRM+08] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality.*, 2008.

[WS09a] D. Wagner and D. Schmalstieg. Making augmented reality practical on mobile phones, part 1. *IEEE Computer Graphics and Applications*, 2009.

[WS09b] D. Wagner and D. Schmalstieg. Making augmented reality practical on mobile phones, part 2. *IEEE Computer Graphics and Applications*, 2009.

[WW11] M. Wozniewski and P. Warne. Towards in situ authoring of augmented reality content. *IEEE ISMAR Workshop on Authoring Solutions for Augmented Reality*, 2011.

[ZFN02] X. Zhang, S. Fronz, and N. Navab. Visual marker detection and decoding in ar systems: A comparative study. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 97. IEEE Computer Society, 2002.

[Zha00] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000.