Augmented Reality System with Planar Homographies using Building Façade

Nan-Hung Cho (Jack)

Master of Engineering (Research)

University of Technology Sydney
Faculty of Engineering and Information Technology

School of Electrical and Data Engineering

August 2017

Principle Supervisor: A/Prof. Qiang Wu

Co-Supervisor: A/Prof. Jian Zhang

Acknowledgement

My two years in University of Technology Sydney have been an apprenticeship under my principle supervisor, Qiang Wu. I would like to express my sincere gratitude for his continual guidance, technical consultation and discussion during my research period. His guidance and insight have shaped this thesis, and I remain deeply grateful for everything that he had taught me.

I would like to thank my co-supervisor, Jian Zhang, for his help and support while Qiang was absent, and my other colleagues for their help on answering my questions regard to technical issues and related mathematical theory.

Finally, I dedicate this work to my beloved family, who offered unconditional support and their mentor have always been there for me. Thank you so much.

Table of Content

List of Figures and Tables

List of Acronyms

Acronym	Definition
AKAZE	Accelerated KAZE
AR	Augmented Reality
CMR	Computer Mediated Reality
FED	Fast Explicit Diffusion
FPS	Frame per second
HMD	Head-mounted display
kNN	K-nearest Neighbour
LDB	Local Difference Binary
M-LDB	Modified-Local Difference Binary
NNDR	Nearest Neighbour Distance Ratio
POI	Point-of-interest
RGB	Red, Green, Blue
SIFT	Scale-invariant Feature Transform
VR	Virtual Reality
KAZE	KAZE feature as described in [1]

Abstract

In recent years, there has been widespread adoption of AR technology with various applications in the urban environment. The essence of an AR application includes the estimation of camera and object placement, which is still a challenging research problem. The first issue is that the structure of the real-world environment is usually very cluttered and involves many unforeseen uncertainties which cannot be generalised as a model. Secondly, the reprojection of the real-world scene to the camera's viewpoint flattens 3D information down to a pixel level, which causes what is known as perspective projection.

Fortunately, it is possible to recover the pixel level of the camera pose via planar homography. Methods commonly used in typical AR applications to recover the homography are based on feature-point matching, but many of them still suffer from the problem of perspective and camera movement.

By taking characteristics of the urban environment, such as buildings which are widely seen on the street view, we propose a simple, yet efficient, approach to recover the camera pose from a single image. Our approach exploits the building façade of rectilinear structures portrayed by man-made structures (e.g. windows and brick structure). Using lines from these structures, our proposed algorithm can produce a rectified homography for each detected building façade, so we can then recover the fronto-parallel view of each building façade. Since these planar homographies are invertible, we can use them to recover the camera pose of the building façade for the object emplacement task. Finally, we implemented a tracker-based AR application that uses the idea of a fronto-parallel view that results in better matching effectiveness.

Chapter 1

Introduction

This chapter covers the background information on AR and the scope of this thesis. Section 1.1 introduces the general concept of what AR is and how the notion of camera calibration is related to achieving a realistic AR effect.

Section 1.2 summarised a brief history of AR's development and a formal definition of AR. This is followed by an overview of several approaches to solving the problem of camera pose estimation required by AR. The remaining sections is about the problem statement and structure of this thesis

1.1.Overview

AR is a technology that brings computer graphics in line with a user's viewpoint via an image or a video stream of the real world. A similar and 'trendy' technology, VR, presents the user with a completely computer-generated virtual world. Both technologies focus on creating a computer system in which users cannot tell the difference between the real world and virtually created elements, to enhance users' experiences and human interactions. More specifically, AR is aiming to provide a tool for computers to "see" the world. When users point their device towards a scene, the computer devices should computationally obtain spatial information from the scene, then overlay graphics on top of the scene that is contextually relevant to the user (i.e. as if it were part of the scene). Fig. 1 is an example of a typical vision-based AR application where the 3D model of a hot desk was rendered with respect to the camera orientation of the tablet device. These computer-generated graphics will behave like a real object so that, when users start to move around, they should be able to see different sides of the virtual objects, even at a different scale moving away from the scene.

Fig. 1 A man is fitting hot-desks using a tablet device

The capability to obtain the camera pose of the augmentation environment is a crucial aspect of AR application. One obvious reason for this is that the visual effect of how well virtual content fits on the scene throughout the footage or video stream determines the overall rating of the AR effect. To achieve the visual effect, the virtual contents need to be "transformed" or "warped" according to the constantly-changing augmentation area in the scene.

Camera pose is a mathematical model that presents the information of camera orientation, which includes, firstly, an intrinsic camera parameter representing the internal configuration (focal length, image scaling factor, etc.) of a pinhole camera and, secondly, an extrinsic camera parameter representing how the camera is moving in the space of the augmentation environment. This problem is usually referred to as *camera pose estimation*.

1.2. Background Information

Computer Mediated Reality (CMR) refers to the topic in computer technology which centres around changing one's perception of reality by inserting, removing or manipulating a real-world object in some way using computer devices [2], as shown in Fig. 2.

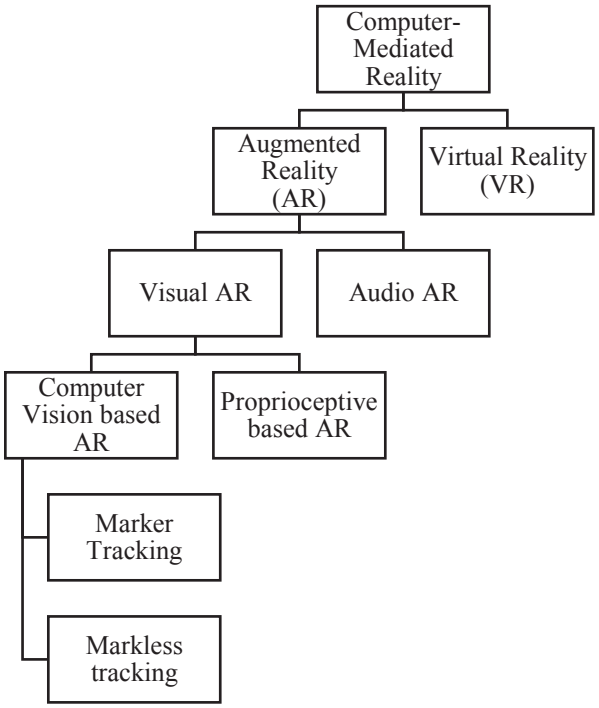


Fig. 2 Taxonomy of CMR application

Both AR and VR have many possible applications in a wide variety of fields, such as military, industrial, and medical applications. Historically, Jaron Lanier is known as the pioneering computer engineer who coined the concept of the modern VR system [3] in 1989. He created a computer simulated environment and a bodysuit to transport the user into an alternative, computer-generated world. An example of a VR system is illustrated in Fig. 3. In this system, the user was wearing a HMD with two interactive remote controllers; from the user's perspective, it brings a completely immersive experience of the virtual world, as shown from the LCD monitor.

Fig. 3 A typical environment setup for a VR game

By contrast, AR is aiming to change only part of the real-world environment, instead of manipulating the entire environment. It allows the integration of one or more sources of computer-generated content to be superimposed on the real-world content. The biggest virtue of AR is to provide synthetic computer-generated information or content in composite with, and usually relative to, the real-world part of the scene, in order to enhance users' experience with regard to their physical world.

The first vision-based AR invention can be traced back to Ivan Sutherland's head-mount display in 1968 [4], although the term AR is not believed to have been coined until Thomas Caudell in 1992 [5]. A more commonly accepted definition by Azuma [6] states that an AR system will contain the following properties:

- It combines real and virtual objects with each other
- It registers (aligns) the virtual objects with the real world in three dimensions
- It runs interactively

AR can also refer to the audio domain, as suggested from its name: the focus is on audio and aural augmentation. In 2016, a kickstarter project namely Doppler Lab [7] introduced an augmented audio solution that provided a software package allowing users to adjust and customise the audio spectrum

so they can attenuate the sound they do not want to hear and amplify the sound they do want to hear from their surrounding environment. For the current research, the focus will be solely on visual-based AR methods.

Vision-based AR is usually achieved by estimating and tracking real-time parameters of the six degrees of freedom, or 6-DoF (yaw, pitch and roll, each with two degree of freedom), of the camera pose or the user's viewpoint of the scene to seamlessly overlay the computer-generated content as though it consistently aligns with the real-world camera feed. In computer research, this problem is referred to as 'camera pose estimation'. As mentioned earlier, the accurate estimation of camera pose is usually the most crucial and one of the most challenging problems to which many researchers were striving to come up with their own solution. Image registration between two environments is difficult in practice due to the complexity of object structures presented in the scene, which usually require either many computer vision algorithms with some presumptions about the environment, or the use of multiple input data from different sources. Fundamentally, the state-of-the-art approaches for solving the camera pose estimation problem can be divided into three main categories:

1. **Proprioceptive sensors**

   Use of a sensory device such as accelerometers, GPS or other digital compass to obtain the raw data as an input for further filtering and estimation.

2. **Computer vision techniques**

   Exploiting the advantages of "feature point" or other geometric properties, with some presumptions about the environment, to estimate the camera pose from the image taken from the camera feed.

3. **Hybrid approaches**

   Combining both proprioceptive sensors and computer vision techniques for better estimation of a more complicated environment.

Full details for each of these approaches are provided in a literature review in Chapter 3, where the focus for this thesis will be on the computer vision technique; however, a brief outline of the advantages and disadvantages is now given for each approach.

Proprioceptive sensors

One of the obvious advantages to using proprioceptive sensors is that they provide the most direct way to measure the relative camera pose compared to the other approaches, thereby avoiding high computational complexity. In practice, these raw signals usually contain inherent inaccuracies caused by internal hardware or external noise; therefore, many modern AR applications do not solely rely on the use of these data unless pixel-level positioning accuracy is not possible.

Computer vision technique

The use of computer vision-based techniques is perhaps the most popular and common approach due to its high accuracy on camera registration and robust alignment of the content with the camera feed, as well as the fact that the camera has become the most ubiquitous sensory device which people can easily access. Computer vision approaches for performing camera registration can be divided into two categories: marker-based tracking and markless tracking.

For marker-based tracking, the calculation of the camera pose is simplified by putting artificial markers (or fiducial markers, as they are also called in the literature) into the scanning environment. These markers are purposely designed to be distinctive and easily detectable, so they will then be segmented and tracked by a properly configured camera. Fig. 4 shows a typical marker-based AR system, where a computer-generated house object been superimposed onto wooden furniture. While this method works stably on a small to medium scale in a controlled environment, such as an office desk or a small office, it is not suitable to set up in a large and uncontrolled environment, such as a building site, because it will be impractical to place

the markers and simply does not look aesthetically pleasing. Due to this impracticality, there is a need to achieve AR in a better way that makes use of the natural feature tracking method, namely markless tracking.

Markless tracking focuses on extracting and calculating the pose information of the camera from natural features such as planes, edges and corner points. One of the advantages of markless tracking is that it does not need fiducial or any kind of artificial markers to be set up prior to the estimation of camera pose; however, this also implies that the markless tracking approach requires a greater computational cost compared to the markless as well as the proprioceptive approach., since in many cases the estimation of some 3D structure information for the scene must be computed. This built-up model can be used as a reference to estimate the camera pose, artificially adding elements onto the real scene. An example of markless tracking is shown in Fig. 5.

Fig. 4 – A house is rendered onto a marker

Hybrid approaches

Hybrid approaches combine both proprioceptive sensors and the computer vision-based technique [8]. One of the advantages of the hybrid approach is that it can offset a certain amount of computational cost usually required by the pure computer vision technique by using the hardware sensors. For instance, the measurements from the proprioceptive sensors can be used as

reference seed values for the computer vision technique, allowing the reduction of searching space or only applying searching techniques to a sub-region rather than the entire image space, which thereby reduces the overall computational costs. The main difficulties of this approach are to find the balance between the computer vision and hardware-based method during the design phase and to later devise a method that maps these multiple data from observation into a consistent, accurate and useful representation. This method is dependent on the required level of alignment accuracy, as well as on the hardware capabilities available to the applications.

Fig. 5 A virtual creature "Yoshi" is rendered on the office desk [9]

1.3. Problem Statement

The aim of this thesis is to develop an AR system for the urban environment. More specifically, the main question is, "how to maintain a reasonable and believable perspective of the virtual content in a short video clip with respect to the changing ROI in the target building façade?" To address this problem, existing computer vision-based solutions that use natural features must be investigated to identify the problem of using this approach for AR. This examination is followed by an investigation into how to exploit some geometric properties of the man-made environment to achieve a better result. The contributions of this thesis are

1. An efficient method for recovering the camera pose relative to the automatically-selected dominant plane of a building façade model in the urban environment, including recovering the fronto-parallel view of the dominant plane.
2. An intuitive way to overlay AR content at pixel-level accuracy, targeted at non-expert users.
3. Development of a simplified, yet efficient, method to track camera pose under canonical view (fronto-parallel) throughout footage that is mainly in an urban environment.

The solution must be able to author and augment the POI planar image using the urban scenes of buildings. The principle requirements for the resulting system are:

1. *The system must first be able to recover the relative camera pose for each building façade from a single image of an urban scene, and then augment with another image at pixel-level accuracy*. The computer vision-based AR is in general more robust for obtaining the alignment of augmented content at pixel-level accuracy. Moreover, this approach must be markless since in most cases it is impossible and impractical to manually put the markers on building surfaces; however, there are several challenges that need to be solved. Firstly, as mentioned in the previous section, markless AR tends to be computational expensive. Secondly, the effect of perspective projection of the image plane distorts the usual (Euclidean) geometry of a building's structure. Additionally, although the feature point matching approaches are commonly used in many applications by detecting and matching the correspondence point of consecutive frames, false-matching can sometimes be a problem. In this thesis, we aim to find a method that can exploit the geometric properties of man-made buildings to recover the camera pose without using the feature point correspondence matching approach.

2. *The system should support non-expert authoring of AR content.* The second objective is to provide an easy and intuitive mechanism for users to insert AR content relative to the building façade.

1.4. Publication List

The following publications have been resulted from the work reported in this thesis:

- Nan-Hung Cho, Qinag Wu, Jingsong Xu, Jian Zhang, Content Authoring Using Single Image in Urban Environments for Augmented Reality. In Proc. of *Digital Image Computing: Techniques and Applications*. pages 1-7, 2016

1.5. Structure of the Thesis

The overall layout of the subsequent chapters is as follows. Chapter 2 briefly reviews the important mathematical concepts prerequisite to understanding the underlying theory behind camera calibration processes. A detailed review of existing works is summarised in Chapter 3.

Chapter 4 presents a walkthrough of our research solving the problem of pose estimation using a single image, as well as the sample augmentation results. This is followed by the development of tracking camera pose for consecutive frames under the canonical viewpoint, as presented in Chapter 5. Lastly, conclusions and future works are summarised in Chapter 6.

Chapter 2

Mathematics prerequisite

Given that the focus of this thesis is on camera pose estimation for AR, this chapter provides a short introduction to the essential mathematical notations used throughout the thesis.

2.1.Geometric Transformation and Homography

Creating a consistent geometrical appearance with respect to the planar surface in the image plane is the most crucial requirement of AR applications. Some common geometric manipulations include rotation, scaling, shearing, reflection, and so forth. These fundamental operations are usually referred to as *transformations*, which can be described with a *transformation matrix*. A transformation matrix takes an input vector existing in $\mathbb{R}^m$ that maps the vector of points in the m-dimension to a new, corresponding vector of points in the n-dimension (i.e., $\mathbb{R}^n$) according to the parameters inside a transformation matrix. In the case of a 3D world where any location is described as a vector of 3 components (x, y and z), the transformation will have a size of 4×4.

The transformation (or warping) between a pair of images can also be described by a transformation matrix, with a size of 3×3. The difference is that the camera model, in this case, treats everything observed in the scene as if they exist in the same plane, so this transformation matrix is also known as *homography*. In other words, a homography is a special case of 2D image transformation between the different viewpoints of the same plane object.

2.2. Projective Space and Homogenous Coordinates

Euclidean geometry is the most commonly used to describe the geometry that we are familiar with. It has some important properties, such as that the sides of objects have lengths, intersecting lines determine angles between them, etc., which make it intuitive. Importantly for this study, these properties do not change under Euclidean transformations (translation and rotation). One way to "locate" or identify objects in Euclidean space is via the Cartesian coordinate system, where any point defined in 2D space can be described by a vector of two elements (x- and y-coordinate). For a digital image, a Cartesian vector is what defines a pixel location. Similarly, a point in 3D space is described by a vector of three elements (x-, y- and z-coordinate), and so on for the higher dimensions (even though we cannot visualise them).

The assumption here is that each point has a "uniquely" defined vector with a numerical coordinate and that the geometrical properties are fixed; for example, a rectangle in Cartesian space is always 90 degrees at its four corners (which never move) but, in projective space, any geometry is alive – a rectangle does not have to be 90 degrees and can move around or transform itself in any way, on condition that it is projected from the two fixed points at infinity, as shown Fig. 6.

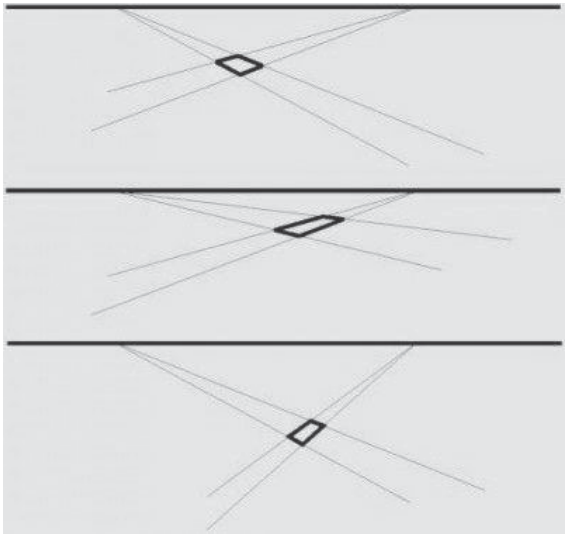


Fig. 6 – A rectangle in projective space

The image projection process of a camera follows the same principle, as although it is possible to apply Euclidean geometry to describe the transformation process up to a certain degree, the calculation is subtle and, in many cases, the mapping relation is not *linear*; e.g., lengths and angles are no longer preserved, and parallel lines may intersect, as illustrated in Fig. 7.

Fig. 7 – A sequence of images shooting at the same building under different camera poses.

To convert Cartesian coordinates (more specifically, inhomogeneous coordinates) to a homogenous coordinate system from a 2D image plane, append an extra '1' to the 2D vector; i.e. $(x, y)^T$ to $(x, y, 1)^T$. Here, the extra 1 is just a placeholder that represents the plane $z = 1$, so that eventually this is the resultant projection in the image plane. When undergoing a transformation, the (x, y, z) might ended up having the same x- and y-coordinates, except that they exist in a different projective plane where $z \neq 1$. To project them back to the image plane, simply divide the x- and y-coordinate by z, since plane $z = 1$, then the resultant x- and y-coordinate value is what will appear in the image plane. The conversion from the homogenous coordinate image plane (Cartesian) can therefore be written as $(\frac{x}{z}, \frac{y}{z}, 1)$.

2.3.Camera Projection Model

It is commonly assumed that image formation comes with the assumption that it follows the rules of *perspective projection*, resulting in the so-called 'pinhole model'. This model's concept has been well-developed in the past decades, [10] giving a full and detailed explanation on the topic. Concisely speaking, the projection describes the mathematical relationship mapping coordinate positions in the 3D world which are transferred to the view plane (of the image) along lines that converge on a point called the *centre of projection*. A pinhole model can also be used to describe the camera motion of a smartphone or a digital camera.

In this thesis, we assume there is no lens distortion. An example is shown in the right-hand image in Fig. 8, where the fence and the other elements in the scene appear to be the same as a human eye's perception, whereas in the left-hand image, part of the elements were distorted meaning it is not a rectilinear lens.

Fig. 8 – An example of lens effects. In this thesis, rectilinear lens is assumed.

Following the overview of a pinhole camera model described above, we now define a brief mathematical expression of the pinhole camera projection. This projection model denotes a projective mapping of a three-dimensional point in the real-world to the corresponding two-dimensional point in the image plane. For more information and a derivation of the formula, refer to [10].

The 3D homogenous coordinate point $X \in \mathbb{R}^4$, a column vector of the 3D point with an extra 1 pending at the forth element, is mapped to the image coordinate $x \in \mathbb{R}^3$ by the projection according to the following formula:

$$x = K[R|t]X \tag{2.1}$$

Eq.2.1 factorises the projection process into two parts: K for the *intrinsic camera matrix*; and [R|t] for the rotation matrix and translation, or *extrinsic camera matrix*.

Intrinsic Matrix

The elements of K are defined as follow:

$$K = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Where the parameters f_x and f_y are the focal lengths of the x- and y-coordinate in millimetres, respectively; s is the sheering parameter; and u_0 and v_0 are the x-coordinate and y-coordinate of the centre of projection (or the principle point).

There are many approaches to estimating these parameters. Throughout this thesis, unless otherwise stated, the intrinsic camera matrix is assumed to be known. In addition, we also assume that the camera model used in this thesis follows the standard perspective projection and the properties of the square array sensor model; therefore, the intrinsic camera matrix can be simplified as:

$$K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Where f is assumed to be the same in both the x- and y-direction; and the centre of projection is half of the length, width and height of an image.

Extrinsic Matrix

The camera's extrinsic matrix describes the coordinate location of a camera in the world. It has two components: a 3×3 rotation matrix and a 3×1 translation column-vector.

$$[R|\boldsymbol{t}] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right] \tag{2.4}$$

In fact, it is common to use this matrix with an extra row vector of $[0 \quad 0 \quad 0 \quad 1]$ appended to the bottom, which makes it a square matrix.

$$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

The matrix describes the mapping relation of the coordinates in the real-world to the camera coordinates. Here, $\boldsymbol{t}$ can be interpreted as the position of the real-world origin in the camera coordinates, with each column of $\boldsymbol{R}$ representing the world-axis of the camera coordinates.

Complete camera model

The combination of the intrinsic and extrinsic matrix multiplied together forms a transformation matrix for representing a complete camera projection model, denoted by C, with the intrinsic matrix denoted by K. The mathematical representation is as follows:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \underbrace{\underbrace{\begin{pmatrix} \frac{1}{\rho_u} & 0 & u_0 \\ 0 & \frac{1}{\rho_v} & v_0 \\ 0 & 0 & 1 \end{pmatrix}}_{K} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0_{1\times3} & 1 \end{pmatrix}}_{C} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.6}$$

The first component of K consist of the scaling factors, ρ_u and ρ_v which scale from meters to pixels. Secondly u_0 and v_0 shift the origin to the top-left corner in the pixel coordinate. Then, this matrix is multiplied with focal length

then the extrinsic matrix. This process maps the points from world coordinates to an image, denoted by $\tilde{u}$ and $\tilde{v}$. Together, they form what is known as transformation matrix, with the size of 3×4. Lastly dividing $\tilde{u}$ and $\tilde{v}$ by $\tilde{w}$ to convert back from homogenous coordinate to pixel coordinate, denoted as u and v.

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.7}$$

$$u = \frac{\tilde{u}}{\tilde{w}}, v = \frac{\tilde{v}}{\tilde{w}}$$

Homography is a special case of transformation that maps a point in one image plane to another image plane, as shown in Eq.2.7. The assumption is that every point in the image is from the same projection plane. In much of the literature, it is usually referred to as the $z = 0$ plane, since the Z-component of the real-world coordinates was set to 0. Therefore, the equation can be simplified as:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} \tag{2.8}$$

As the Z coordinates were set to 0, the third column of the transformation matrix will also be zero during the multiplication, leaving the matrix with a size of 3×3, as follows:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{pmatrix} = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.9}$$

The letter $C_{i,j}$ was changed to $H_{i,j}$ to distinguish that a homography is a special case of the camera model. Essentially, both models encapsulate the camera orientation (extrinsic matrix) as well as the camera-dependent parameter (intrinsic matrix).

In this thesis, the homography will be the model of the transformation matrix to be estimated, which is then used to warp the virtual content. We assumed that the camera internal parameters, f, equals the pixel width of the input image, and that the centre projection is equal to half the size of the image width and height, respectively.

Chapter 3

Related Works

3.1. Proprioceptive sensor-based AR

The proprioceptive sensor-based method uses sensors such as a GPS or digital compass to directly (or partially) measure the orientation of the camera pose (quaternion, axis rotation etc.). The first AR system was developed in 1968 by [4], who tracked the head position of users and projected the aligned graphics using a see-through head-mounted display (HMD). In the late 1990s there were many follow-ups works proposed [5, 6] to measure the orientation and position of parts of human body to achieve pose estimation. By the end of the 2000s, the concept of AR became more well-defined and some promising applications were proposed to assist doctors carrying out patient biopsies [11].

One of the early prototypes for the AR system was developed by Hollerer in 1999 [12]. This system measures the 6DoF camera orientation of the user's viewpoint using GPS, gyroscopes and accelerometers. GPS, magnetic, ultrasonic, and Time-of-Flight (ToF) sensors then became, and are still, popular sensory devices for many research applications in AR. In addition, some commercialised systems like Layer [13], Wikitude [14] and ARNav [15] also use the proprioceptive sensors in-built in the smartphone to create a realistic AR effect. Of course, as mentioned earlier, modern AR applications do not usually rely solely on the raw data because they suffer from limited accuracy. A more sophisticated approach is to combine the image data and exploit feature points of some sort, applying sensor fusion techniques and performing mathematical optimisation processes to achieve a decent augmentation result.

3.2. Computer Vision-based AR

The fundamental theory of the computer vision-based technique is to make use of the geometric properties or photometric information from image frames to estimate posture information about the camera. This approach, when compared to the proprioceptive sensor and its much greater computational costs, permits a significant boost in the accuracy of registration. Currently there are two main streams of computer vision based approaches: marker-based and markless-based tracking.

Marker-based AR

For marker-based AR, one or more fiducial markers or other artificial markers [16, 17] are placed into the field of view. These markers are usually designed to be distinctive and easily detectable, so they can be used as a reference point in the image and be used for later recovery of the 3D orientation and position of the camera. The early implementation of markers in AR can be traced back to the publication by Rekimoto et al. [18] who first created a system that mounted a camera on a mobile phone screen to display some text or graphical information according to the input received from the camera, using colour-coded markers. Later, they further adopted this idea with a 2D Matrix code which allowed the system to track the 6DoF posture information of the camera [19]. Since then, marker-based registration has gained more attention in the research area and many improvements have been proposed. Zhang et al. [20] compiled a review of several of the common approaches.

The square marker is perhaps the most common approach and is still a popular method after the first release of ARToolkit – an open-source library for marker AR in 1999 [17]. The library provides the functionality for tracking the 6DoF posture information of a camera using square markers. The system performs the following process to detect the square marker: image thresholding; contour extraction and line fitting; and using the extracted corner points to estimate an initial camera pose. This method is relatively

simplistic, yet robust enough to allow image-based AR using commercial off-the-shelf (COTS) cameras.

The succeeding work, namely ARTags, was released in 2004 and became another well-known marker-based tracking AR open-source library [21]. Similar to ARToolkit, it uses digitally-encoded fiducial markers. The pattern of markers is predefined and has been shown to outperform the ARToolkit, which uses binary images, by allowing for a greater robustness towards changes in lighting conditions and partial occlusion, resulting in lower detection rate of false positives. In another study, Garrido-Jarado et al. proposed an improved pipeline of marker-based tracking, namely ArUco [22]. Their proposed method tries to solve the issues of many other existing methods; for example, previous methods used a predefined pattern (dictionary) of markers which meant that applications requiring more than the size of the dictionary had trouble setting up the environment. Similarly, for many scenarios where the number of markers required is smaller, it is preferable to use a smaller dictionary size so their inter-marker distance is as high as possible, which will reduce the inter-marker confusion rate. Another issue they handled was of partial occlusion, as marker-based AR has a major drawback due to the limitation to augmented experience when the marker is partially covered by our hand during the process, meaning that users cannot interact with the virtual object directly. The overall robustness and visual effect has been shown to outperform many previously-developed methods and, therefore, it has become a popular method. It has been adopted in OpenCV [23] – a well-known open-source library for computer vision – as part of the component to realise marker AR implementation.

To sum up, marker-based AR has become a popular technique to create a high degree of accuracy for camera pose estimation, as compared to utilising the sensory device alone; however, this approach only works under the confining condition where it is feasible to manually set up the artificial markers within the scanned environment. For an uncontrolled environment, such as a

building site or larger region, setting up the markers simply becomes impractical and would not look aesthetically pleasing even if it were feasible. This drawback has driven the motivation to develop another approach to allow scalability: markless-based tracking.

Markless-based AR

Given the limitations and impractical aspects of marker-based AR approaches, there has been considerable study devoted to using only the information from naturally-occurring features in the scene, such as edges, corner points or blobs that can be extracted without artificial markers; this is known as markless-based, as opposed to marker-based, AR [24-30], as shown in Fig. 9. An earlier work by Park et al. presented a method for using natural feature points to update the camera pose calculation after the initial fiducial marker is taken away from the view [31]. In descriptors such as SIFT and SURF, the encoded feature points are robust against changes in image scale, or even partially occlusion. A review of recent methods can be found in [32]. The feature point extraction relies on searching for potential local extrema and the generation of SIFT descriptors for subsequent feature matching. There are other works [33, 34] intending to further improve speed and overall robustness, to better handle the affine transformation for feature encoding and matching.

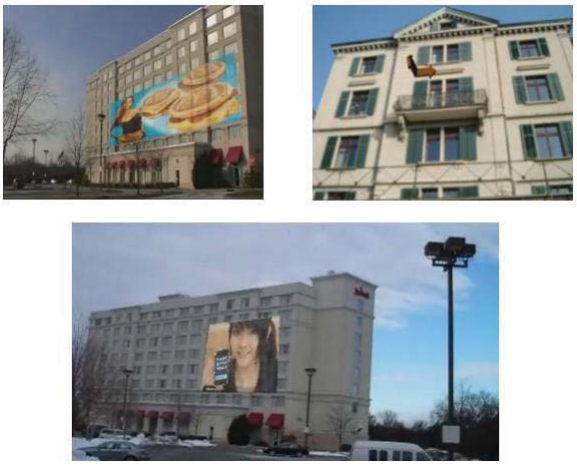


Fig. 9 – Some examples of AR in an urban environment

There are other markless-based approaches which rely on an *a priori* model or map of the scene to track against. One of the advantages offered is that the estimation of the camera is more stable and robust, although it also comes with more cost in computation time during the training stage compared with marker-based approaches. More recent and popular methods for real-time AR include Structure-from-Motion (SfM) [35] or Visual Simultaneous Localisation and Mapping (SLAM) [36-38]. These methods primarily deal with uncalibrated camera image sequences, such as from a hand-held camera. Fundamentally, the core techniques for feature-based markless AR usually comprises the following components:

- Feature detection and descriptor generation – SIFT is the most popular and generally-used; there are many variations proposed for different purposes and needs.
- Feature matching – also referred to as "correspondence matching" in many research papers.
- Structure / motion estimation

In 2007, a novel AR implementation called Parallel Tracking and Mapping (PTAM) was proposed by Klein et al. [39], who demonstrated the capability of achieving real-time performance to augment a scene while concurrently generating a map for the target scene to track against. PTAM has tracking and mapping components running in separate threads. Initially, the map is created from a stereo pair of images via a 5-point algorithm [40]. The mapping component then plays an important role in matching the key frames using the Bundle Adjustment algorithm. In addition, an epipolar search method [41] is used to initialise and track new points for new keyframe insertion, allowing the map to grow during the runtime. Overall, PTAM has not only demonstrated the ability to perform in real-time, but also the ability to operate without having any prior knowledge of the target AR environment. As the original paper on PTAM was targeted at a small AR workstation, an extension work has been presented called Parallel Tracking and Multiple Mapping

(PTAMM) [42] which works over a larger area and can swap between multiple generated maps of the target AR environment.

One of the shortcomings of the markless AR techniques mentioned above is that they require the image sequence be taken at similar viewpoints to a certain degree in order to establish inter-frame correspondence for matching. When applying these methods in a perspective projection of the scene structure, there is a higher chance that the matching will fail, thus leading to erroneous camera pose estimation.

3.3. Planar Rectification

Images taken from a camera often change the geometric structure of the building due to the nature of the camera projection model. For the AR solution that uses the markless approach, this could be problematic as plane geometry undergoing perspective transformation distorts Euclidean properties such as length, angle and parallelism. As a result, feature tracking and correspondence matching will fail and thus lead to erroneous estimation of the camera pose.

There are many research groups aiming to overcome this issue, which is often referred as perspective rectification [43, 44]. Fig. 10 shows the corrected perspective view of a building façade where the plane geometry observed in the original picture was brought back to the camera fronto-parallel. In fact, perspective rectification has numerous applications, including image mosaicking [45], planar object recognition [46] and single-view 3D building reconstruction [47].

Fig. 10 – Perspective rectification of a building façade

The mathematical model for the process is well-understood – the mapping relation between the world plane and perspective is a planar projective transformation, which is formulated by 2D homography [48-50] in which it has eight degrees of freedom (see Eq. 2.9). The goal of the planar rectification process is to employ a mathematical optimisation approach to compute the optimal solution so that the Euclidean properties of the building structure are recovered (Fig. 10). Often, however, only little or none of this information is available. One earlier work by Collins and Beveridge [51] demonstrated the process on registering satellite images. Their work has shown that the vanishing point plays an important role in reducing the projectivity to affine transformation. In addition, they also showed that the orientation of the plane with respect to the camera view can be computed if the intrinsic camera matrix is known.

There are other methods that exploit geometric properties such as epipolar lines rectification [52], or the rectangle structure in a man-made environment. Such solutions are commonly acknowledged as the Manhattan model [53], which assumed that parallelism of scene structure and angle regularity appeared in man-made environments; this thesis belongs to that category. The proposed solution for this thesis focuses on exploring these geometric properties for a man-made environment in order to develop an accurate yet efficient image registration pipeline for our AR application.

Previous works have observed that these man-made environments usually contain a great number of parallel lines aligned along with some fixed orthogonal lines; however, due to perspective projection, these parallel lines in the 2D image plane will eventually converge at the vanishing point. The parallel lines and their corresponding orthogonal directions can be used to produce a rectified view of the image scene; i.e. metric rectification. The purpose of generating this rectified view is to transform a set of consecutive image planes with different and unknown pose to a common ground in order to estimate the homography relation between each inter-frame.

The presence of such common ground is a viewpoint invariant and can be helpful for simplifying the camera calibration application. [47] utilised this approach in their pipeline to recover rectangle structures from the rectified view in order to realise a single view-based 3D reconstruction. [54] also proposed a method for finding the vanishing point and camera parameters using mutual orthogonal line segment patterns found in man-made structures. [55] took advantage of the vanishing point in man-made structures to develop a method for building façade extraction and recovering the camera pose [16] for a typical AR application; however, [55] took all line segments into account throughout the entire process. This result can be easily distorted if the building structure does not fill up the entire image plane, because they rely heavily on the result of vanishing point detection. In addition, it requires manual selection of line segment structure to recover the rectified view.

3.4. Literature Review Summary

This chapter conducted a comprehensive literature review on the current technology for visual AR systems. The current research will focus on those that use markless-based computer vision approach. The goal is to investigate a solution to exploit the rectangle structure and geometric properties appearing in the Manhattan model, or other properties of such models, to improve the limitations of markless-based AR. As discussed in the chapter, the current technology for markless methods is not considered to be the best solution in many street view AR systems due to perspective distortion of the scene structure that causes the feature point to fail to detect or match.

Chapter 4

Planar Façade Rectification

This chapter contains a detailed explanation of our proposed solution for camera pose estimation in AR systems, as illustrated in Fig. 11. In addition, our approach allows a single image from an urban environment to be augmented without the need for expensive computational resources to obtain the 3D structure of the scene. The key assumption in our proposed solution is that, given a captured urban scene, there are plentiful line segments that meet perpendicularly to each other. More specifically, the goal of our method is to exploit the geometric constraints of the 3D scene to bring these perpendicular line segments in a so-called fronto-parallel view which maximises the number and orthogonality of projected line segments.

The overall process is described as follow: an image is first passed to the line segment detector to obtain the line structure of the scene. This is followed by the line gap filling and line extension as part of pre-processing in order to combine the clumsy and noisy line segments into longer line segments. The next step is to construct a so-called orthogonal pair matrix based on a set of geometrical constraints to build up the candidate orthogonal line-pairs, which will be used in the dominant rectification process (Section 4.4).

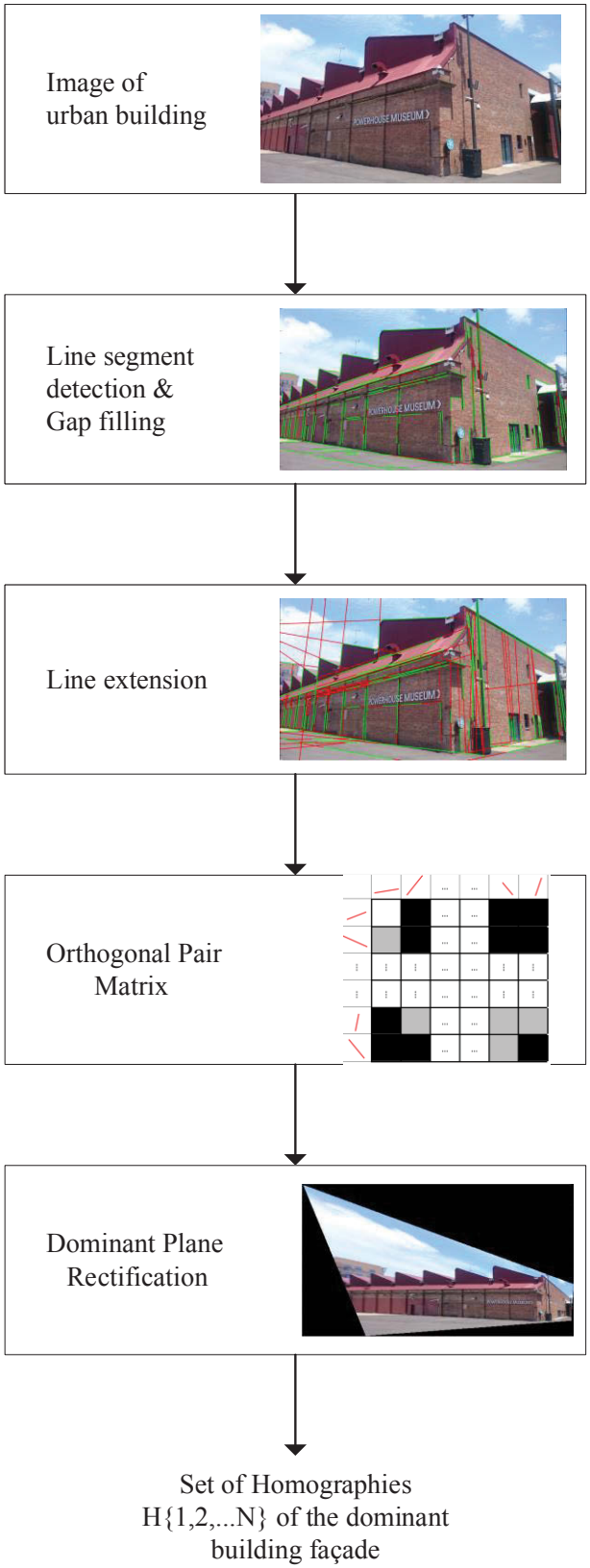


Fig. 11 – An overview of the planar rectification algorithm.

4.1. Line Segment Detection

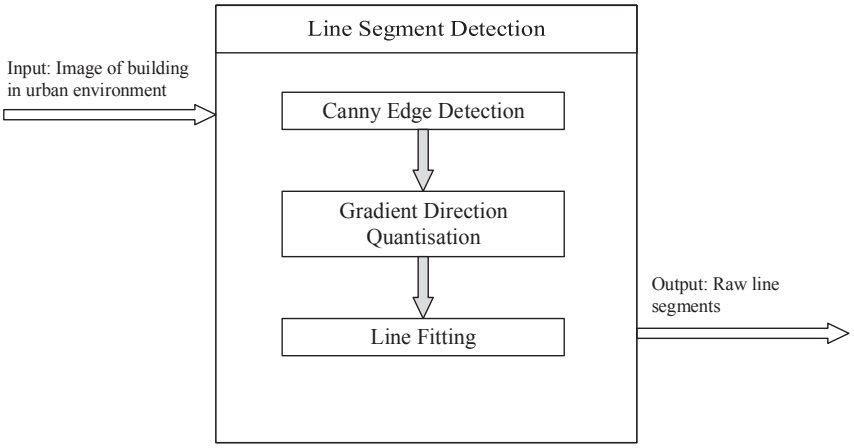


Fig. 12 – Illustration of the line segment detection algorithm.

A line segment is the section of the line defined by two definite end-points outlining the building structure. The process of this stage is illustrated in Fig. 12.

In the first part of our pipeline development, this is adopted from the existing algorithm by [56]. An input image is firstly processed by the Canny edge detector and then the gradient is defined as follows:

$$\theta = \tan^{-1}\left(\frac{\partial I/\partial x}{\partial I/\partial y}\right) \tag{4.1}$$

The gradient direction is then quantised into a predefined number of bins ($k = 8$ in our implementation). Edge pixel orientation will fall within a range of intervals according to the following definition:

$$bin_i = \frac{\theta}{\pi k} mod(k) \tag{4.2}$$

The edge pixels that belong to the same bucket are then grouped together using connected-components labelling. A line fitting stage will be performed to generate the straight line. During the line fitting, only those larger segment components are considered. Although the threshold length depends on the actual environment, in our case, lines with 40 pixels' length or above are selected.

4.2. Gap Filling and Line Segment Extension

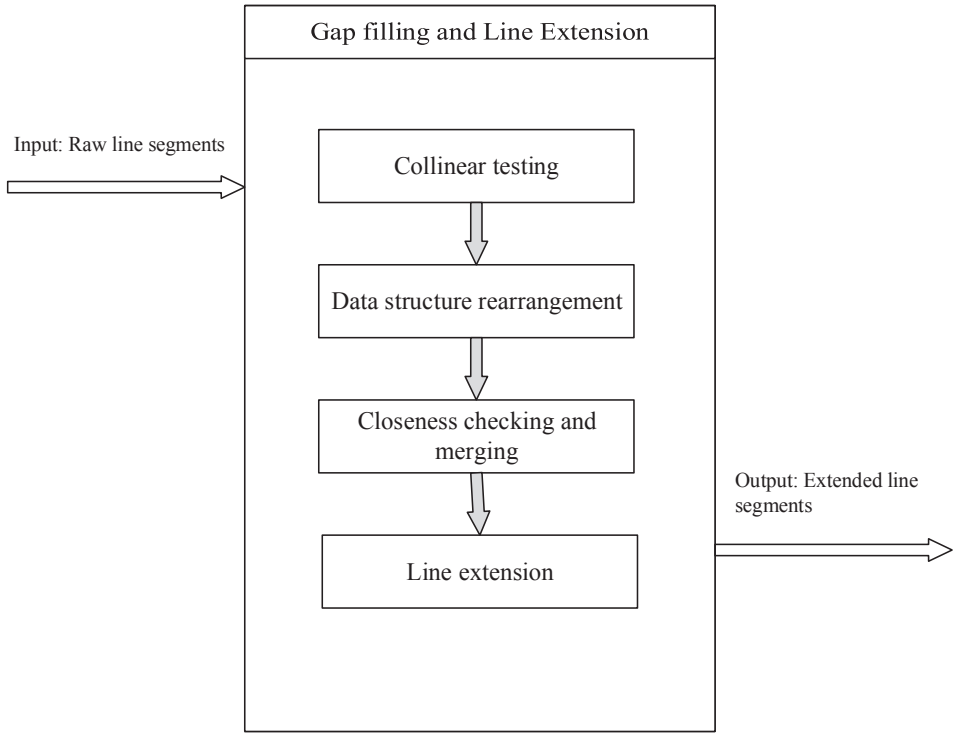


Fig. 13 – Illustration of gap filling and the line segment extension algorithm.

Gap-filling algorithm

The line segment detection provides a strong clue on how a building looks, as shown in Fig. 14, but these line segments are unevenly distributed and cluttered, so the overall pattern of a building structure may not be salient at this stage. To overcome this issue, we first perform line gap-filling, followed by line extension which will be discussed in the next part. The process of gap-filling and extension is illustrated in Fig. 13.

The gap-filling condition is examined based on two criteria: collinearity and distance closeness. The term collinear and collinearity in the content of this chapter refers to a set of line segments that almost lie on a single straight line. Distance closeness is defined by a threshold that measures between the gap of a line segment to another collinear one. If these two conditions are met, then we can join the cluttered line segments together to form a longer line segment.

The first condition of being collinear between any given pair of line segments is to check a set of angles between them, as shown in Fig. 15. Line segment are defined as $l_a = [v_{a1}, v_{a2}]$ and $l_b = [v_{b1}, v_{b2}]$ containing two endpoints v_i. There are nine sets of angle measurement needing to be checked, and only if these measurements are smaller than a certain threshold, θ_a, are they considered to be almost collinear with each other.

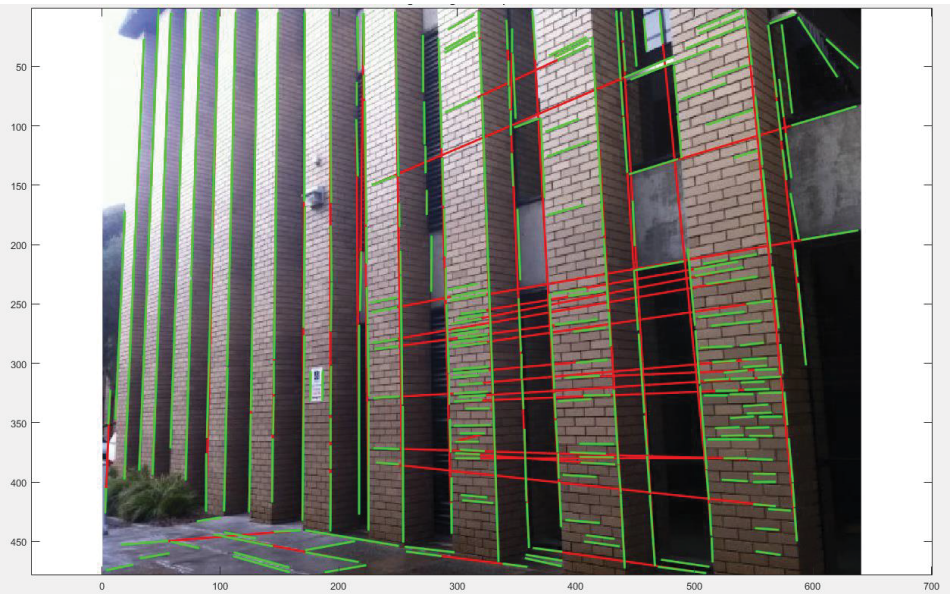


Fig. 14 – Result of line segment and gap-filling algorithm. The red lines are the gap-filling result from the original line segment result, indicated by the green colour.

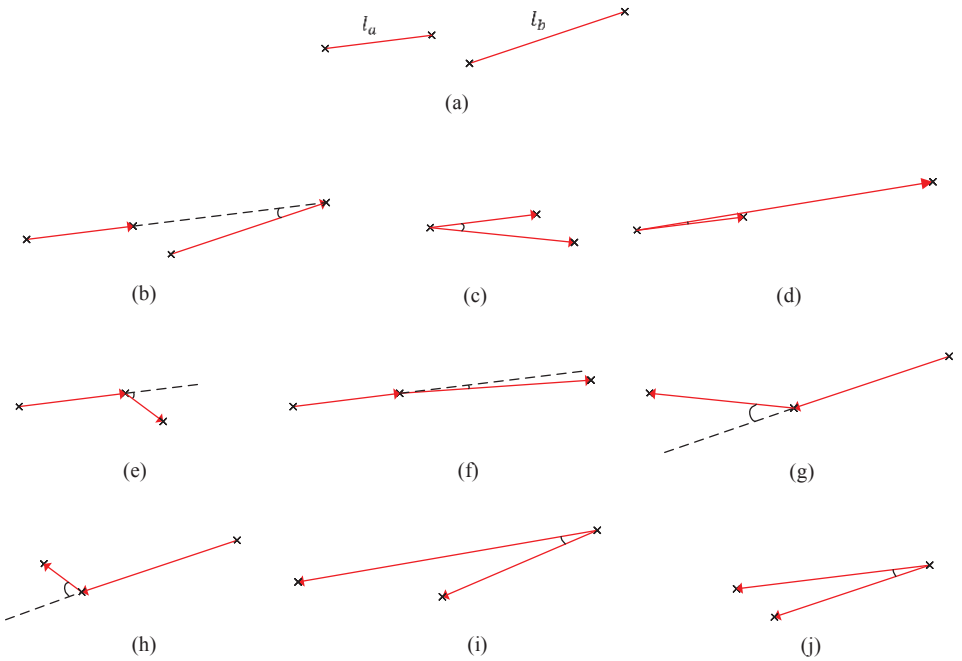


Fig. 15 – An illustration of collinearity testing: (a) is the selected line segment pair to be tested, a line segment is defined by its two endpoints; (b) ~ (j) represent different cases formed by the endpoints of the line segment pair.

The second condition is to check the gap distance. Given a pair of line segments, we want to merge them together if the gap is within a threshold distance, as shown in Fig. 16. Essentially, this can be calculated by picking up the right endpoint of the first line segment, v_2, and the left endpoint of the second line segment, v_3.

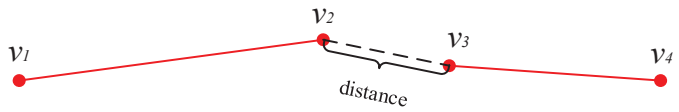


Fig. 16 – Gap distance testing between line segments.

Here, we assume when measuring the gap distance between any pair of line segments that the convention of head-to-tail formation of a vector representation is consistent throughout the entire data structure.

When checking the gap distance between a pair of line segments, we also take into consideration two cases. Assuming both cases have passed the gap distance threshold, the first case is shown in Fig. 17 (a) where the length of both line segments are noticeable, then they should be merged. In the second case shown in (b), however, the line segments are significant shorter even though the gap distance passed the threshold. In this case these line segments will not be merged.

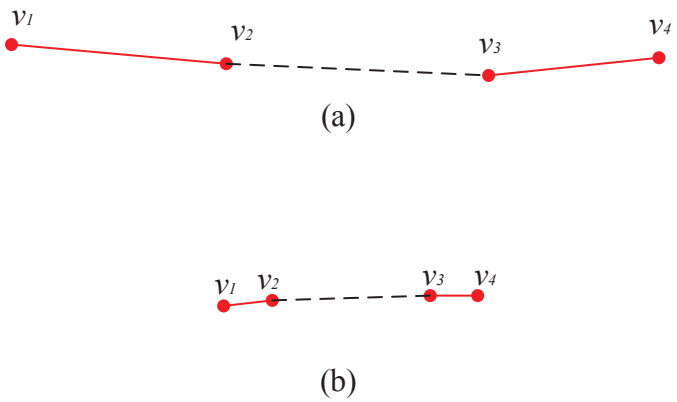


Fig. 17 – An example of line segments which satisfy the gap distance threshold: (a) will be merged together; while (b) will not be merged because the total length of line segments is smaller than its gap distance.

Line segment extension algorithm

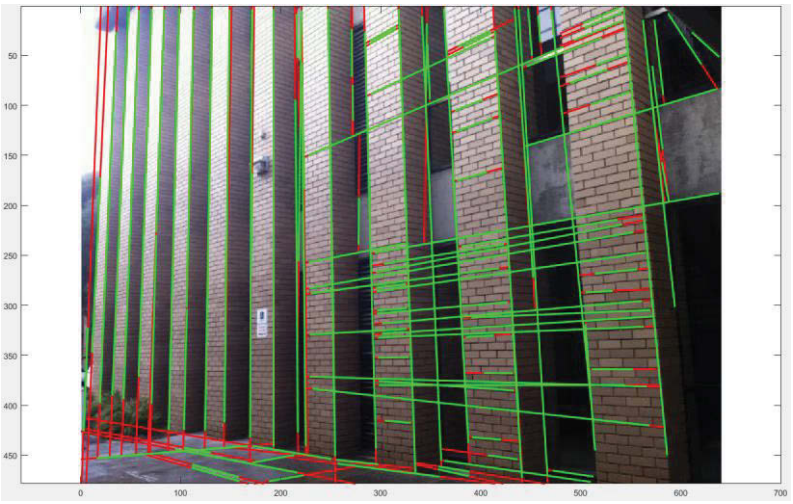


Fig. 18 – Sample result of line segment extension. Red lines are the result of line extension of the original ones shown in red colour.

The next step is to extend all the line segments so they at least intersect with another line segment; an example is shown in Fig. 18 where the red line segment is the extended line segment from the original line segments in green. Notice each resulting line segment is extended as far as it intersects with the most-nearby line segments. By doing so, the angle regularity becomes more noticeable for checking in the subsequent processes.

Our implementation was built for this special requirement of the line extension process, based on previous work by Antonio [57]. In the following, we briefly describe the mathematical formulation before presenting our implementation for the sake of completeness.

Consider the vector representation of a point $P_i = (x_i, y_i)^T$, a line segment $L_{1,2}$ can be defined by two endpoints P_1 and P_2. Assume a point P_x lying on $L_{1,2}$, then it can be represented parametrically using a linear combination of P_1 and P_2 as follows:

$$P_x = \alpha P_1 + (1 - \alpha) P_2 \tag{4.3}$$

Or alternatively,

$$P_x = P_1 + \alpha(P_2 - P_1) \tag{4.4}$$

It can be observed that when $\alpha = 0.5$, P_x will be the midpoint of the line segment, while increasing α towards 1 will move P_x close to P_1; or otherwise P_2 when decreasing α towards 0. Moreover, any value outside the $[0, 1]$ range will move P_x away from P_2 when $\alpha < 0$; or otherwise P_1 when $\alpha > 1$. This idea can be illustrated in Fig. 19.

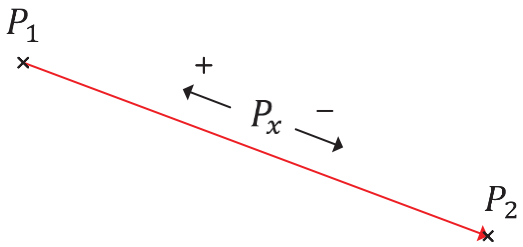


Fig. 19 – Point P_x lies on line segment $P_{1,2}$, can be represented using parametric equations.

Now consider a pair of line segments defined by two endpoints, $L_{1,2}$ and $L_{3,4}$, and a point P^* that lies on both line segments somewhere in-between, as illustrated in Fig. 20.

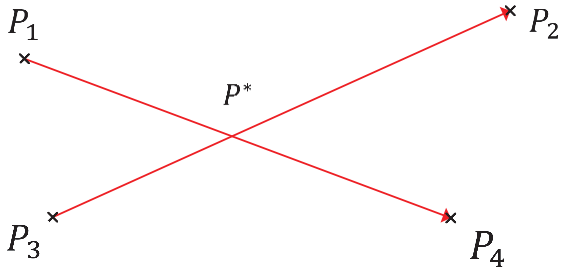


Fig. 20 – Intersection of line segments by solving its linear system of parametric equations.

The solution can be constructed by setting up a linear system of equations by applying Eq.4.4 with two parameters, α and β, whose ranges are within the interval $[0, 1]$, as follows:

$$\begin{cases} P^* = P_1 + \alpha(P_2 - P_1) \\ P^* = P_3 + \beta(P_4 - P_3) \end{cases} \tag{4.5}$$

Then, subtracting the top equation from the bottom, we get:

$$0 = (P_1 - P_3) + \alpha(P_2 - P_1) + \beta(P_3 - P_4) \tag{4.6}$$

The parameter α and β are the parametric solutions of the intersection for $L_{1,2}$ and $L_{3,4}$ respectively. Our goal is to extend the endpoints of $L_{1,2}$ to the most-nearby line segments, then we are seeking a solution that can indicate how much of each endpoint needs to be extended to lie on these nearby line segments.

We first calculate all possible solutions of α and β with respect to the selected line segment $L_{1,2}$, at each iteration, then remove those with β outside the range of $[1, 0]$, since the they do not lie on $L_{3,4}$. For α, we set up the following criteria to select the best two candidates, denoted as ΔL_1 and ΔL_2, from the intersection point to be jointed with $L_{1,2}$:

- ΔL_1 represents the new endpoint of the vector head of line $L_{1,2}$, which is located at the largest α value for $\alpha < 1$.
- ΔL_2 represents the new endpoint of the vector tail of line $L_{1,2}$, which is located at the smallest α value for $\alpha > 1$.

After this step, each line segment is guaranteed to be intersected with its nearest line segments by extending the original endpoints of $L_{1,2}$, which can be derived from Eq.4.3 as follow:

$$\begin{cases} P_{1'} = \Delta L_1 P_1 + (1 - \Delta L_1) P_2 \\ P_{2'} = \Delta L_2 P_1 + (1 - \Delta L_2) P_2 \end{cases} \tag{4.7}$$

Which can also be illustrated as in Fig. 21.

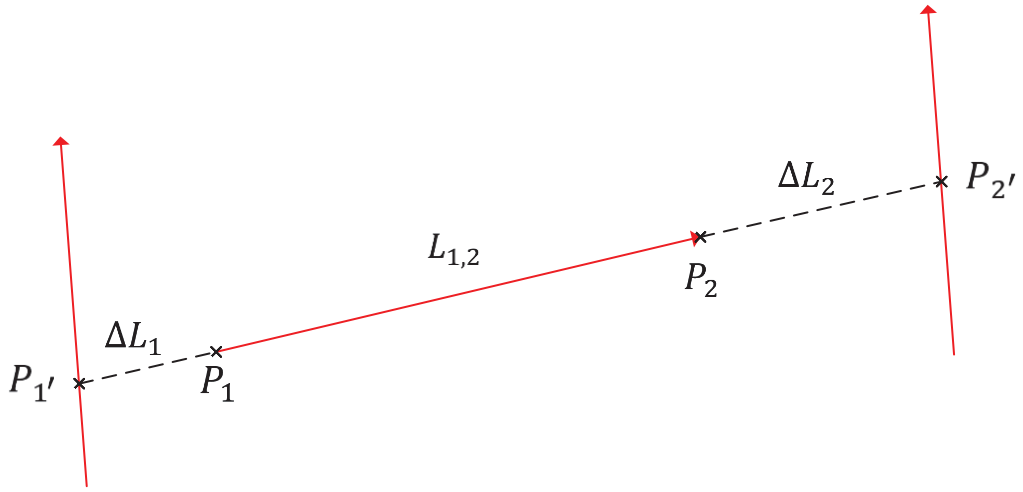


Fig. 21 – An illustration of a line segment, $L_{1,2}$ extended to its nearest-neighbour line segments.

4.3. Orthogonal Pair Matrix Generation

Our aim is to generate a 2D array giving an index of all possible orthogonal line-pair candidates found in the line segments. We express this matrix as a matrix of $C \in R^{n \times n}$, where n is the total number of line segments. For each entry in the matrix, the index column and row represent the corresponding line segment index, and a Boolean indicator is assigned, as shown in Fig. 22. If line segments, l_i and l_j, intersect with each other and are orthogonal, a '1' will be assigned, otherwise '0' is assigned; an illustration is shown in Fig. 22.

The candidate selection is based on the following criteria:

$$C_{i,j} = \begin{cases} 1, & \theta_{adj} \geq \lambda \\ 0, & otherwise \end{cases} \tag{4.8}$$

Where $\theta_{adj} = \left(l_i^T \cdot l_j\right)$ is the cross-product of l_i and l_j. Their corresponding $C_{i,j}$ will be set to 1 if θ_{adj} is greater than a pre-defined threshold λ, otherwise 0 will be assigned. The resulting matrix is called the orthogonal pair matrix, which provides all possible orthogonal line-pairs to be used in the subsequent rectification process.

Line Segment Index	[1]	[2]	[3]	…	[n-2]	[n-1]	[n]
[1]	0	1	0	…	1	0	1
[2]	1	0	0	...	0	1	1
[3]	0	0	0	…	1	1	0
...	...	...	...	…	...	...	...
[n-2]	1	0	1	...	0	1	0
[n-1]	0	1	1	…	1	0	1
[n]	1	1	0	...	0	1	0

Fig. 22 – An illustration of a sample orthogonal pair matrix. Each element in the matrix indicates if a pair of line segments by its index is orthogonal. Notice that the diagonal elements are always 0 since comparing to the line segment is invalid.

4.4. Dominant Plane Rectification

So far, all the pre-process components discussed above represent preparation for the step now described. The main idea of our proposed solution is to exploit the angle regularity of rectangle structures found in many building scenes, and thus, in the previous step, we generated so-called orthogonal line segments pairs for this metric rectification process. Throughout the entire optimisation we are seeking a set of best homographies that recover the dominant planes a building façade to the frontal-parallel view.

Recovery of a façade view via RANSAC

As discussed in Chapter 3.3, an image plane under perspective projection can be rectified to the frontal-parallel view. One important point is that the aspect ratio of a 2D quadrilateral needs to remain constant. According to [10], a transformation homography for viewing the same 3D plane due to pure rotation can be defined as follow

$$H = KR_Z^{\gamma}R_Y^{\beta}R_X^{\alpha}K^{-1} \tag{4.9}$$

K denotes intrinsic camera matrix. Firstly, it consists of three rotation matrix for each axis, our goal is to obtain these rotational parameters to recover the rectifying homography. For now, we ignore the effect of R_Z^{γ} for now since it is only a similarity transformation (which we will consider it in the next section). Therefore, the metric rectification function for x and y axis can be reduced to:

$$H_{x,y} = KR_Y^{\beta}R_X^{\alpha}K^{-1} \tag{4.10}$$

From Eq.4.10, our aim is to search for a rectifying homography. Given a pair of line segments, l_i and l_j, the arbitrary homography will bring these lines, i.e. $H^{-1}l_i$ and $H^{-1}l_j$, to the orthogonal basis. We denote the product of $H^{-1}l_i$ and $H^{-1}l_j$ as $\tilde{v}_i$ and $\tilde{v}_j$, which is normalised to the unit norm, respectively. Since they must be perpendicular to each other, the resultant dot product $\|\tilde{v}_i \cdot \tilde{v}_j\|$ should be as minimal as possible (i.e. $\|\tilde{v}_i \cdot \tilde{v}_j\| \cong 0$). Thus, the optimisation of finding this arbitrary homography is to minimise the sum of the dot product result over the line segment pairs, giving the following equation:

$$\min_{\alpha,\beta} \sum_{i,j} \|\tilde{v}_i \cdot \tilde{v}_j\|^2 \tag{4.11}$$

The orthogonal line segment matrix play an important role in this optimisation process, but they are rather a rough extraction based on a simple assumption of being perpendicular. Here, we apply RANSAC [58] to iteratively estimate an underlying model to remove outliers' data, as well as labelling each line-pair to the corresponding surface plane. The process starts with an initial homography of $diag(1,1,1)$, and picks up any two sets of line segment pairs from the adjacency matrix, computing a rectifying homography through several iterations according to Eq.4.11 until the total cost function reaches below a predefine threshold value, t.

The resulting homography, D, will be used to extract the inliers by the number of line-pairs that have become orthogonal in the rectified image plane. Here,

we added a requirement to determine whether this homography can be considered a frontal-parallel view of the dominant plane by examining if there are enough inliers, denoted as $||D||$, to be considered a dominant planar surface. Depending on the number of orthogonal line segment pairs, the threshold for $||D||$ will be proportional to it. In our implementation, this threshold is set to greater than 10% of the number of orthogonal line segment pairs, otherwise it does not count as a plane. Furthermore, at each iteration, the corresponding line segment pairs in the inliers are removed from the orthogonal line segment pairs matrix. Next, the remaining elements in the matrix are processed again until either the sizable inlier line-pairs $||D||$ cannot be found or the remaining matrix of orthogonal line segment pairs is less than 10% of its total amount. After the RANSAC process, the original line segment pairs are separated into a set of containers that belong to the same dominant planar surface as well as a set of rectified homographies that recover up the similarity transformation. Fig. 23 illustrates the pseudo-code for this part of algorithm [59].

Algorithm 1: Metric rectification using orthogonal line-pairs

Input:
I: An input image with one or more dominant planes;
C: The orthogonal line-pair matrix;
d: RANSAC threshold value for cost function evaluation
t: Minimal number of line-pairs;

Initialization:
$C_{remain} = C$;
$S = \{\}$: selected inliers;
$H_{xy} = \{\}$: rectified homographies;
$m = 0$: number of plane detected
$\| \cdot \|$ represents the number of '1' s in the matrix
while $\|C_{remain}\| > t$ **do**
    $D = \{\}$: current inliers
    Randomly pick up two orthogonal line-pairs from C and perform iteratively cost function optimisation of finding the best transformation parameter W using
    for ***all*** C_{remain} **do**
        Apply homography transformation W to all line segment pairs in C_{remian}
        if $(v_i^T \cdot v_j)^2 < d$ **then**
            $D_{i,j} = 1$;
        end
    end
    if $\|D\| > t$ **then**
        $H_{xy} \leftarrow W$;
        $S \leftarrow S \cup D$;
        $m \leftarrow m + 1$
    end
    Subtract the C_{remain} from C
end

Output:
S: selected inliers in m planes;
$H_{xy} \in R^{3 \times m}$ rectified homography in m planes.

Fig. 23 –Dominant planar façade classification via RANSAC

Pure rotation recovery

In the second stage of optimisation, we aimed to recover the similarity homography H_z that we did not considered from the previous section. From our observation of running many building images, we concluded that rotation effects on detected building façades roughly range from 20 to 40 degrees. Thus, our approach is to manually construct a normalised vertical vector [1,0] and horizontal vector [0,1]. These two vectors are the ideal orientation we

want to recover from the similarity transformation, H_z. The estimation of this transformation is described in next paragraph.

The formulation of the cost function consists of two parts, the vertical and horizontal. For each component, we are aiming to find a homography defined by a rotation parameter about the Z-axis, denoted as $\gamma \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, that will minimise the result of dot product for these two components (i.e. be as close to zero as possible). This cost function can be written as follows:

$$\min_{\gamma}(\sum_{i}\left|l_i^T \cdot [0,1]\right| + \sum_{j}\left|l_j^T \cdot [1,0]\right|) \qquad (4.12)$$

Where l_i and l_j are the horizontal and vertical line segments, respectively. After each iteration, the cost function will be re-calculated by transforming l_i and l_j via $H_z l$ where $H_z = KR_z^{\gamma}K^{-1}$, until this cost function reaches the local minimum.

The resultant transformation is then combined with two homographies computed from the previous two parts; i.e., $H = H_{xy}H_z$. This homography is finally adopted to recover the camera pose with respect to each planar surface detected.

4.5. Experiment

The configuration in our experiment is now described. When an input image is provided [1], the system first attempts to obtain the intrinsic matrix. The system is then proceeded to the main algorithm of planar homography recovery, as discussed in Chapter 4.4, which will generate a set of fronto-parallel views when a hypothesis plane is detected.

---

[1] Our sample images are usually 640x360. In case of using an image bigger than 1000 pixel in either dimension, we resized the image according to a scale factor so that either dimension is up to maximum 1000-pixel unit length.

Experimental setup

The experiment is run on a general-purpose PC with Intel® i5-6300U 2.4 GHz, with no GPU or accelerated component used. There are two sets of data used for the experiment, the first set being the randomly generated graphics from a perfect square grid, which mimic some of the common viewpoints of a building. The second set are photos taken from a smartphone camera, in which there is an analytical solution to calculate the intrinsic matrix (focal length and projection centre), and also a data set from a publicly available databased namely ZuBUD [60], as well as random images from the Internet. In this case, the default intrinsic matrix is used (see the last paragraph in Chapter 2.3).

Experimental data

The first experimental data are denoted as "noiseless" graphics so that we have control of the environment. Here, we use the grid shown in Fig. 24 as a base image. The purpose of using these experimental data is to examine the overall performance of each stage of the proposed algorithm in several scenarios. For the core part of the rectification algorithm, we further formulated some mathematical criteria to measure the correctness of the rectification results.

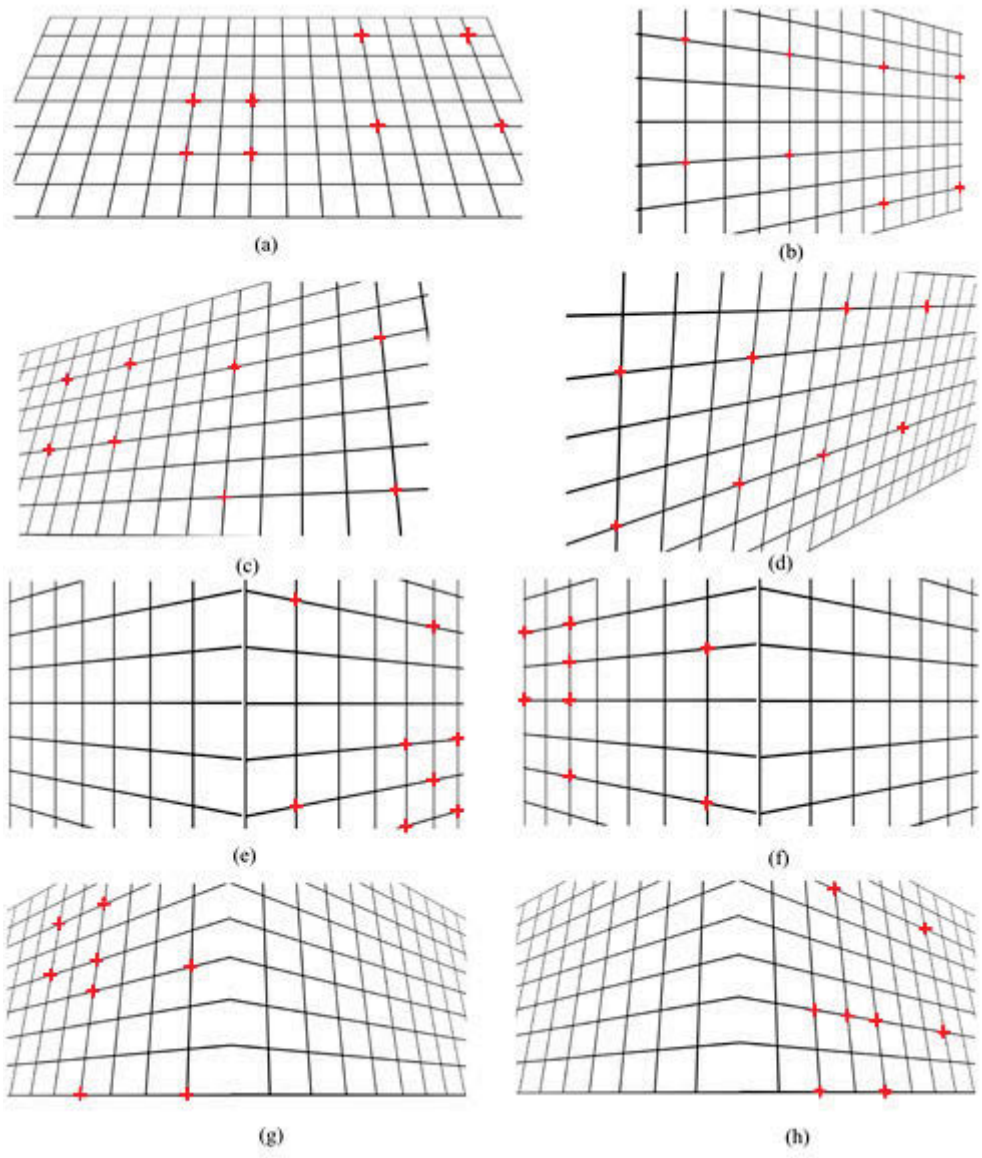


Fig. 24 – Rectification sample images (a) ~ (d) consist of only one face, and (e) ~ (f) are pairs of images with two faces

Performance measurement

Few measurements were considered to quantify the amount of correctness from a given rectified view. For the geometric distortions, it is measured in three criteria: the diagonal ratio (r_{diag}), top-bottom length ratio (r_{tb}) and orthogonality ($\theta_{\perp}$).

Let us denote a rectification homography, H^{-1}, and four manually annotated corner points of a rectangle region in the original image $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ and $\boldsymbol{l}$. By transforming these corner points to the rectified view, the diagonal ratio can then be measured from the length ratio between two diagonals as:

$$r_{diag} = \max\left(\frac{d(H^{-1}\boldsymbol{i}, H^{-1}\boldsymbol{l})}{d(H^{-1}\boldsymbol{j}, H^{-1}\boldsymbol{k})} - 1, \frac{d(H^{-1}\boldsymbol{j}, H^{-1}\boldsymbol{k})}{d(H^{-1}\boldsymbol{i}, H^{-1}\boldsymbol{l})} - 1\right) \tag{4.13}$$

The same principle for measuring the top-bottom length ratio can be formulated as follow:

$$r_{tb} = \max\left(\frac{d(H^{-1}\boldsymbol{i}, H^{-1}\boldsymbol{j})}{d(H^{-1}\boldsymbol{k}, H^{-1}\boldsymbol{l})} - 1, \frac{d(H^{-1}\boldsymbol{k}, H^{-1}\boldsymbol{l})}{d(H^{-1}\boldsymbol{i}, H^{-1}\boldsymbol{j})} - 1\right) \tag{4.14}$$

In both equations, the operator $d(\cdot,\cdot)$ is the Euclidean (L^2) distance between two homogenous coordinate points. The subtraction of -1 is to offset the resulting value so that in the most ideal case, i.e. in a perfect rectangle, both ratios of r_{diag} and r_{tb} are 0. Finally, the orthogonality of the four corners is measured as follow:

$$\theta_{\perp} = \cos^{-1}\left(\frac{l_i^T C_{\infty}^* l_j}{\sqrt{(l_i^T C_{\infty}^* l_i^T)(l_j^T C_{\infty}^* l_j^T)}}\right) \tag{4.15}$$

l_i and l_j are the vector representations of lines formed by joining any two of the clockwise or counter-clockwise corner points. C_{∞}^* is the matrix representation of a conic dual. Since the fronto-parallel view in the current context is the standard 2D Euclidean space, $C_{\infty}^* = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$. After four corner angles are measured, take the largest angle difference as the final resulting error.

Experimental Results and Discussion

Here, we present the experimental result at each stage of the algorithm and a discussion of the results and limitations.

Line gap-filling and extension

The gap-filling works effectively on the perfect square grid, as shown in Fig. 25(a) where the red parts indicate the new (extended) line segments. One can see that most of the missing lines are filled except a pair in the vertical direction; in this case they are treated as different line segments even though they are ideally the same. As discussed in Chapter 4.2, the line gap-filling only concerned line segments in its vicinity if the threshold condition was met (see Fig. 17). After the line gap-filling was performed and before it proceeded to line segment extension, we removed 30% of line segments based on their shortest length ranking.

When applying the same setting on multiple planar surfaces the resulting line segments sometimes do not appear to be perfectly recovered. An example is shown in Fig. 26, where the some of the horizontal line segments are falsely merged with other line segments.

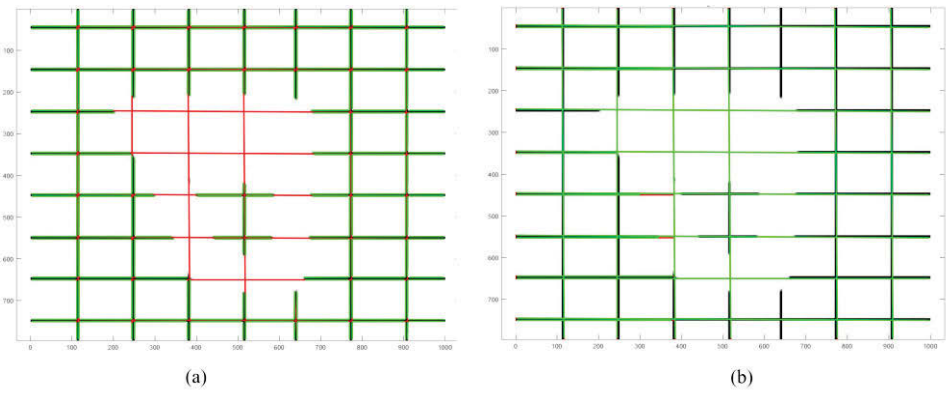


Fig. 25 – Sample experimental result of (a) line gap-filling and (b) line segment extension.

We will see later that this can be compensated to a certain degree during the rectification stage if the majority of the patterns are preserved.

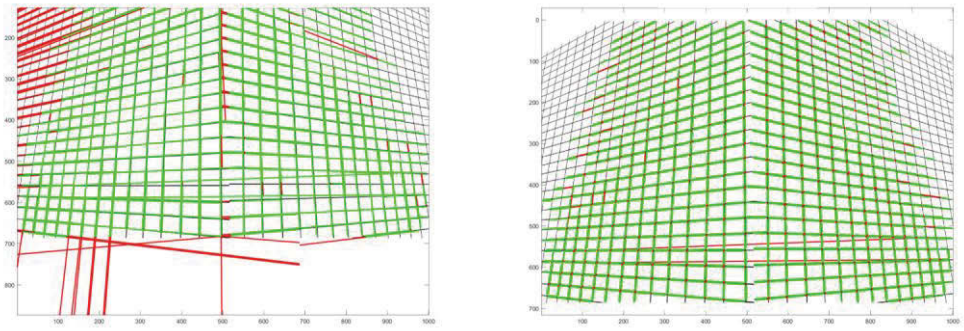


Fig. 26 – Line segment and gap-filling on a grid image.

Rectification results

We manually selected some of the quadrilateral regions, as shown in Fig. 24. The red markers are the four corner points of the rectangle regions, which should ideally be a rectangle in fronto-parallel view. The goal was to perform mathematical measurements on these regions to evaluate the correctness of the rectification results.

Throughout the experiment we measured against several square regions and compared with another similar method in [55] and [61]. [55] mainly focuses on finding the boundary line on different planar façades and requires users to provide two horizontal and vertical line segments on each façade to obtain the rectification homography. For [61], we provide an initial region upon initialisation of the method.

Table 1 is the average error of measurements. Our result show a significant improvement compared to [55], while [61] generally outperform our proposed solution (excepted for the width-height ratio). Despite this, the advantage of our solution is that it does not require manual operation from the users during the optimisation, as opposed to the other two methods which can both result in a different rectified view from the user's input.

Average	r_{tb}	r_{daig}	$\theta_{\perp}$	width-height ratio
[55]	0.0118	0.0649	6.2810	0.2778
[61]	0.0048	0.0048	0.5222	0.3180
PROPOSED	0.0151	0.0139	1.5166	0.1575

Table 1 – Rectification performance comparison on square grid images.

Real-world image rectification results

The source of building images was from [60] and random images from the internet. Since these images do not contain EXIF data, we will use the image width as the initial focal length for the intrinsic parameters. Some of the example results are shown in Fig. 27. Our proposed method works robustly on many of the building images, even with objects such as power cables and trees acting as outlier elements. Of course, there were still cases of failure and limitations, which will be discussed in the next section.

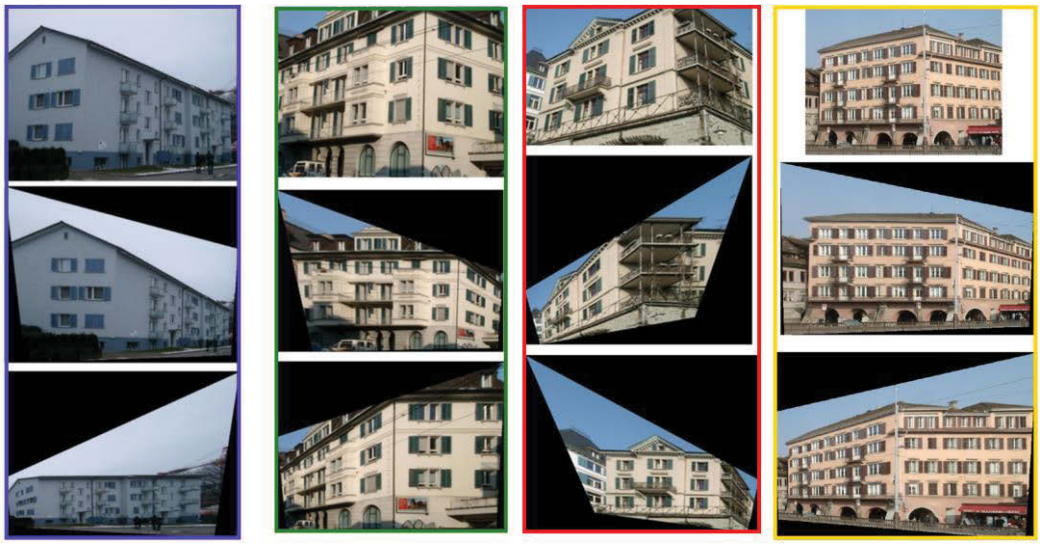


Fig. 27 – Sample rectified view result on various buildings' façades.

2D AR content results

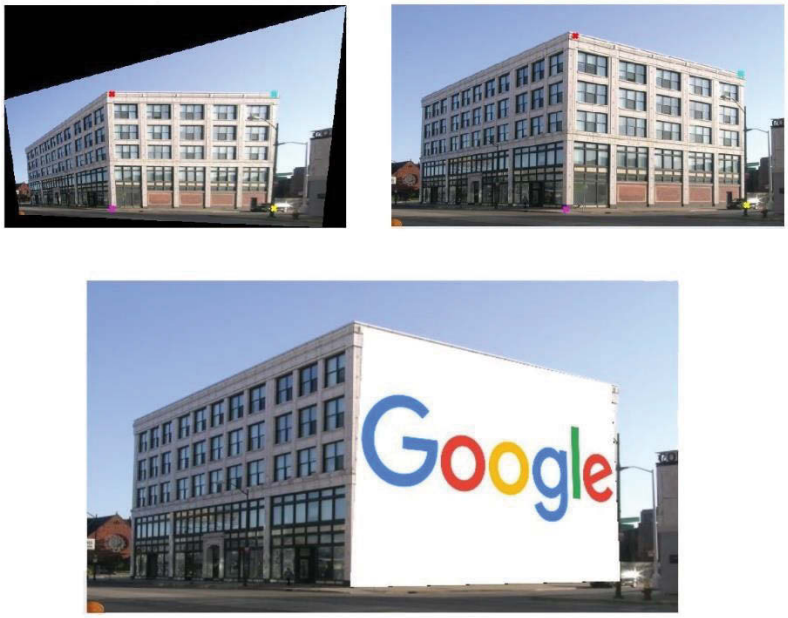


Fig. 28 – Augmentation results:.user manually choose the diagonal points, indicated as red and yellow markers in the original image, so then we can construct a rectangle region by back-projection from the fronto-parallel view. Once the corners are found, a 2D image can be replaced on the region of augmetation.

The sample augmentation results are shown in Fig. 28. Our proposed technique is effective for buildings where the facade structure is dominantly a planar surface. In the presence of outlier objects, our proposed technique will still work when the overall planar surface is still predominant in the image; for example, in Fig. 29(c), the tree and plant generate the outlier line structures which do not affect our method.

There are some limitations to our method; for instance, a building containing a curved shape region will be treated as a flat planar surface with respect to the dominant planar orientation. This is shown in Fig.30(c), where the window is a slightly curved shape. Throughout many trials we found this failure can arise when the X and Y rotation parameter $[\alpha, \beta]$ is greater or less than 1.1 radian.

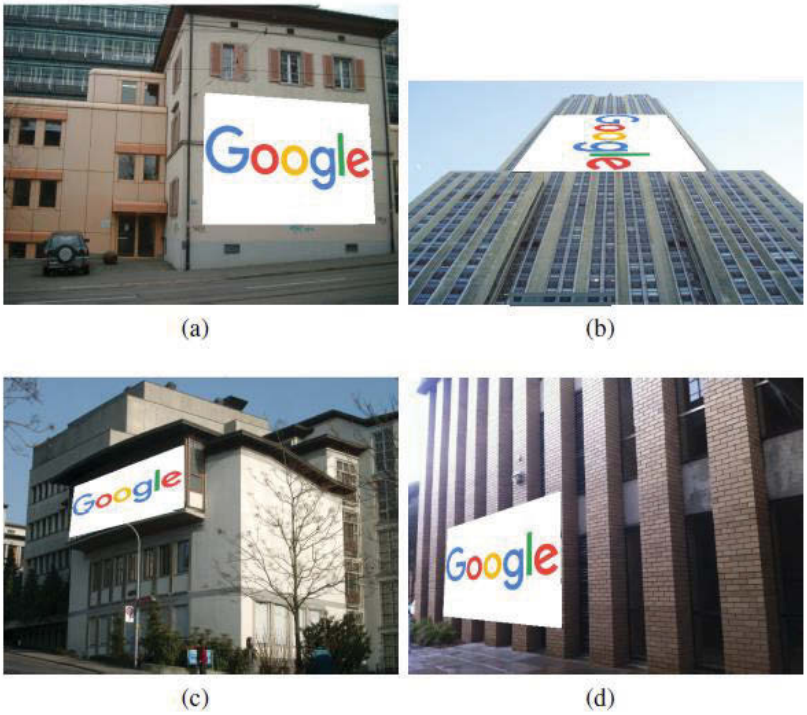


Fig. 29 – Augmentation result

Our proposed solution works effectively when the captured scene contains planar structures. While it will still work on a partially non-planar building façade to a certain degree, the chance of failure increases and it will not work on completely non-planar façades. Since our rectification solution is based on the dominant planar region, any curved or other non-planar region will not be recovered properly. An example is shown in Fig. 30, where the building façade at the front were of the non-rectangle shape while the other buildings are more rectangle-like. As a result, one of the fronto-parallel views corresponding to the non-rectangle building façade were severely distorted.

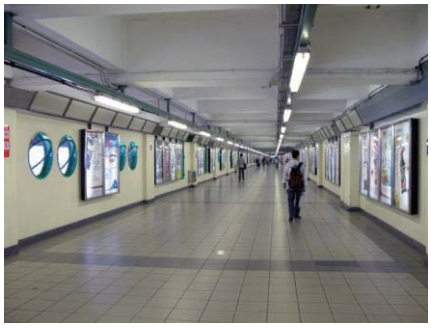
The solution will work on partial occlusions as long as the angle regularity can still be captured by the line segment detection algorithm, but will tend to fail once the occlusion becomes severe.

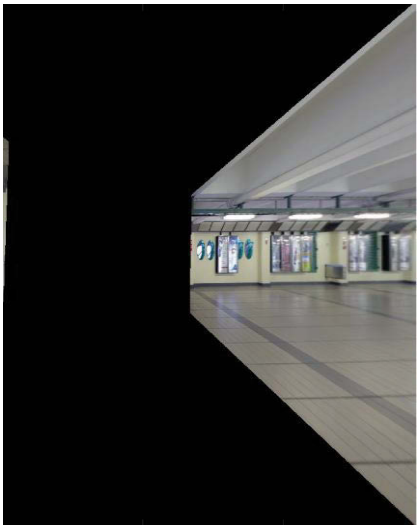
As for other extreme cases, such as in Fig. 31 where the two opposites of the wall face each other and merge into a point in middle of the scene. Our proposed algorithm will also fail to process such environment settings.

(a) (b)

(c)

Fig. 30 - (a) an example of non-planar facade; (b) and (c) are the result of the fronto-parallel view. While it successfully recovers the buildings at the back, it fails to recover the non-planar building facade at the very front.





(a) (b)

Fig. 31 – (a) indoor tunnel example as an extreme case for planar rectification. While the proposed algorithm managed to recover one side of the wall shown in (b), it does not recover the other side of the wall.

Chapter 5

Façade-based AR

5.1.Overview

We performed camera pose recovery of multiple-façades in an urban environment from a single image. In this chapter, we develop a tracking component based on the same idea of a fronto-parallel view. This chapter explains the implementation of the façade-based AR system, which has three main components as shown in Fig. 32. The term 'tracking' in the context of this thesis refers to the estimation of the camera pose for a specific ROI in the image scene by tracking correspondences between the image pair.

Our framework works in the following way. We first used a simple tracker to estimate the inter-frame homography. This homography can then be used as part of the cascading parameter with the previously rectified homography to compute an estimate of fronto-parallel homography. This is followed by the implementation of the AKAZE tracker to perform robust tracking and matching. Lastly, we refined the result via a linear Kalman filter and finally oriented the AR image onto the scene.

The advantage of using the fronto-parallel view in the tracking task is that it does not suffer from perspective distortion, and thus the subsequent matching process can be vastly improved.

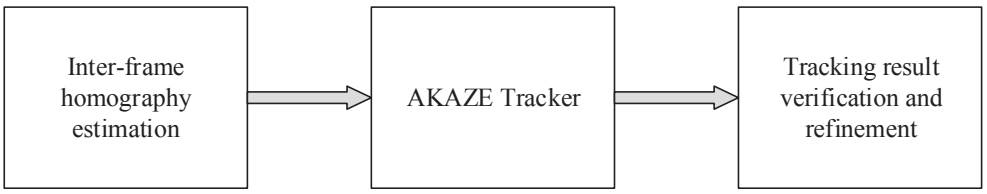


Fig. 32 – Framework of our façade-based AR system

5.2. AKAZE

As discussed in Chapter 3.2, computer vision markless AR is conventionally achieved by tracking feature points and correspondence matching. In this thesis, we chosen AKAZE [62] as the core technology for the pose tracking component. AKAZE is based on the previous KAZE [1] algorithm with two speeded-up modifications: the Fast Explicit Diffusion (FED) for the feature detection part and the Modified-Local Difference Binary (M-LDB) descriptor. Here, we briefly summarised the framework of AKAZE for the sake of completeness – for the detail of each step one should refer to the original paper.

The first step of AKAZE is to build an image pyramid of scale space for the feature point selection process. Like KAZE, it uses the nonlinear scale space which is experimentally demonstrated to be a better way to build the image pyramid. Conventionally, the formulation of the image pyramid utilises a fixed set of scales determined by the width (variance) of Gaussian kernels, categorised as an isotropic diffusion[2] operation. KAZE uses Perona and Malik diffusion [63]. The formulation difference is that the diffusion coefficient varies with the position within the image and depends on edges. Furthermore, its succeeder framework AKAZE speeds up the process of scale space building via FED. Once the image pyramid is built, a selection of keypoint locations that exhibit a maximum of the scale-normalised determinant through the Hessian response is then obtained in nonlinear scale space (of the image pyramid) as follows:

$$L^i_{Hessian} = \sigma^2_{i,norm}(L^i_{xx}L^i_{yy} - L^i_{xy}L^i_{xy}) \qquad (5.1)$$

[2] Diffusion in the context of image smoothing refers to the width of the Gaussian kernel which operates on the image. In other words, at each octave of the image pyramid, different scales correspond to different diffusion durations: as ink (of a pixel) slowly spread apart over the picture with time, the image's scale gets coarser, dependent only on the level in the scale pyramid.

The parameter σ is the same set of scale factors used in the creation of nonlinear scale space. The technical detail is omitted here but can be found in the original paper.

The feature description stage is based on the Local Difference Binary originally developed by [64], which is similar to a well-known binary descriptor BRIEF [65]. For each detected feature point, a square region of its neighbourhood pixel is partitioned into 2×2, 3×3 and 4×4 grids. Each of these grids will need to capture three parameters: the average intensity level and average image first-derivatives I_{dx} and I_{dy}. The next step is to perform a binary test on a pair of randomly select grids and compare their captured parameters. '1' will be assigned if a parameter in the first grid is bigger than the second one, otherwise '0' will be assigned, as seen in Fig. 33.

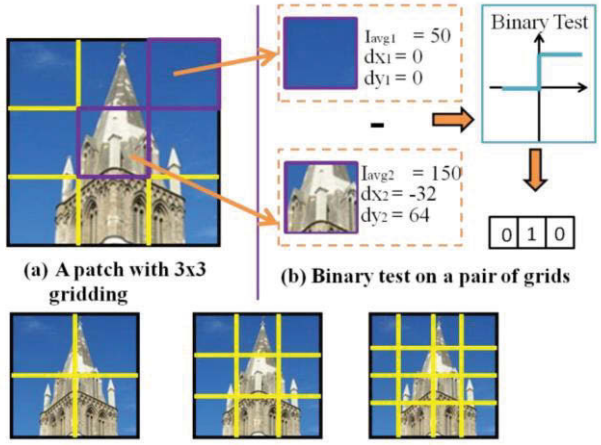


Fig. 33 – Local Binary Difference test illustration.

As a result, a single binary test generates three bits. According to [64], the result is concatenated from 2×2, 3×3 and 4×4 grids, which form 486 string bits from the total combination of $\binom{4}{2} + \binom{9}{2} + \binom{16}{2}$ gird pairs. The Modified LDB (M-LDB) follows a similar framework as the original LDB with less computational cost, while also being robust against scale and rotation change, which is done by taking the rotated grid of LDB into account and scale-dependency of the grid accordingly.

5.3. Tracker implementation

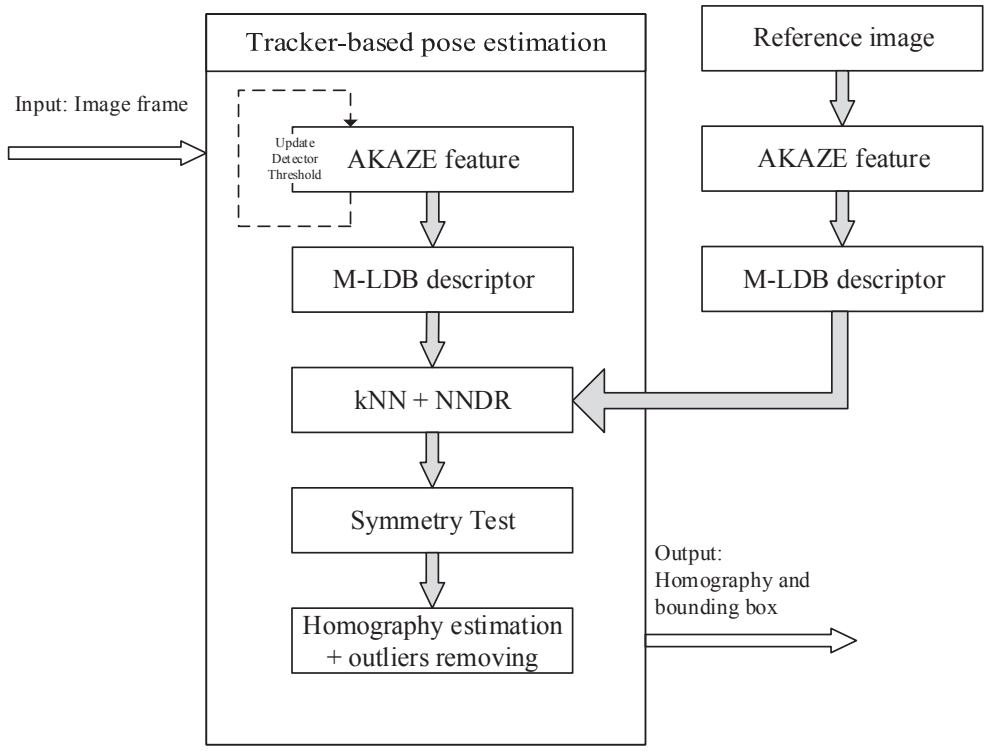


Fig. 34 – Tracker framework

Feature detector threshold adjustment

Fig. 34 shows the framework of our AKAZE tracker implementation. We borrowed the implementation from OpenCV3 for the AKAZE feature and M-LDB descriptor generation. To control the feature selection in AKAZE feature detector, we implemented an extra component that changed the threshold value of the detector accordingly so that the number of feature points were maintained within a range.

We measured the number of detected feature points at each iteration and dynamically adjusted the threshold value for the next loop via linear line fitting. The steps are described as follows:

(1) Define a cut-off value in log scale, i.e. $\log(y_{target})$. In our implementation, this value is the average of the desired number of feature points[3].

$$y_{target} = \log_{10}(\frac{MAX_{n\_feature} - MIN_{n\_feature}}{2}) \quad (5.2)$$

(2) Map the previous threshold value, d_{t-1} and the number of keypoints from last frame, $lastK_{np}$ to a log scale:

$$\begin{aligned} x &= \log_{10}(d_{t-1}) \\ y &= \log_{10}(1 + K_{np}) \end{aligned} \quad (5.3)$$

(3) Perform linear line fitting:

$$x_{new} = x + m(y_{target} - y) \quad (5.4)$$

Where m is a negative number such that $-1 \leq m < 0$; closer to 0 means a slower approach to the target (y_{target}). In our implementation, the value of m is -1.

(4) Map the x_{new} back from the log scale as follows:

$$d_t = e^{\log(10) * x_{new}} \quad (5.5)$$

kNN + NNDR

In the matching process, the k-nearest neighbours algorithm (kNN) and the nearest neighbour distance ratio (NNDR) test are used to find the best matching correspondence. We used the OpenCV kNN search algorithm to return the two closest descriptors from the query set (template image) for every descriptor from the train set (current image); in our implementation, $k = 2$. This is followed by a NNDR check to obtain good quality matches.

[3] The conversion to log scale is applied so we can follow the simple linear equation of $y = bx + c$ to estimate the detector threshold value.

The value of the ratio test is 0.8, which was originally suggested by Lowe [66]; if the ratio is below this threshold, the match is discarded as being low-quality.

Symmetry Test

This component servers as an extra refinement in the framework to remove any non-symmetrical matches; for example, if the matching index between the reference image and image frame is inconsistent. At the end of this process, we can use the matching pairs to obtain the correspondence relation via computing homography.

Homography estimation + outlier removal

We performed a homography estimation from the matching result with the RANSAC algorithm to label the outlier pairs by calculating reprojection errors. In our implementation, the maximum reprojection error is 2.5 to treat a matching point as inliers. Finally, we stored the resulting homography and the bounding box.

5.4.Fronto-parallel view via inter-frame homography

Given the correspondence between consecutive frames, a homography is found. Our goal is to synthesise a frontal view of the current image frame with respect to the frontal view of the first frame, denoted by s_0. In other words, we must find a rectifying homography, denoted by C_i where i indicates the frame index, which transforms one view, say frame 2, to the frontal, metrically rectified s_0.

Given that the frontal-parallel view of the first frame is recovered using our proposed solution, we denoted this homography as for fronto-parallel rectification C_1. The homography that synthesised the fronto-parallel view of the next frame can be deduced via a cascade relation of inter-frame homography as follows:

$$\begin{aligned} \mathrm{C}_2 &= C_1 H_{1,2} \\ \mathrm{C}_3 &= \mathrm{C}_2 \mathrm{H}_{2,3} \\ &\vdots \\ \mathrm{C}_\mathrm{N} &= \mathrm{C}_{\mathrm{N}-1} H_{N-1,N} \end{aligned} \tag{5.6}$$

Where $H_{1,2}$ is the homography that describe the correspondence relationship from frame 2 to frame 1 on the original view compared to the fronto-parallel view, and so on. In our implementation, the performance speed for this part is around 20 FPS.

5.5. Tracker result verification and refinement

The result of the tracker contains a homography with the bounding box of the matched region in fronto-parallel view. To examine whether the resulting transformation is "valid", we need some rules to determine whether a given homography from the tracking result violates the rigid body transformation. We also implemented a linear Kalman filter to stabilise the noisy trajectory motion of the bounding box due to errors in the tracking result, as well as to predict the trajectory motion in the case of no homography if found to be due to a lack of inliers or if the homography were rejected due to violation of rigid body transformations.

Problems of RANSAC in homography estimation

To clearly address the problem of homography estimation, we first need to understand the requirement for solving its underlying linear system and how RANSAC is approached to solve this system. According to [10], estimation of homography is the problem of solving eight degrees of freedom parameters. In other words, at least four pairs of correspondence points are necessary to estimate all the homography parameters.

RANSAC is a renowned algorithm for solving linear systems by an iterative estimate of the parameters of a mathematical model from sample data that may contains both inliers and outliers. This turned out to be extremely useful for homography estimation in the image registration tasks. It starts with a putative set of point correspondences from two different images. Essentially,

four pairs of sample correspondences are randomly selected and then iteratively evaluated by first finding an optimal solution of homography using these sample correspondences followed by checking the consistency of all the putative correspondences with respect to this estimated homography.

Even with kNN + NNDR, symmetry tests, and consistency checks in the RANSAC process, an invalid homography can still occur. Such absurd homography due to selection of degenerated samples, as the result of eight bad point correspondences, were unfortunately consistent, which somehow led to violation of rigid body transformation.

Analysis of rigid body transformation via a homography

We are seeking a set of rules which can be implemented to examine whether a given homography obeys rigid body transformation. Here, we take two artefacts into account: the parameters of a homography itself and the change in a quadrilateral after applying this homography. The arrangement of a quadrilateral is clockwise and their initial corner points are consistent with the dimension of the image, as $(0,0)$, $(width, 0)$, $(width, height)$, $(0, height)$ before transformation. We included the following test cases. One should be aware that some of these solutions are analytical while others are more empirical.

- **Homography should preserve the direction of quadrilateral points**: Given the four corners of a quadrilateral with a clockwise arrangement, the resultant points, after applying homography transformation, should always remain in the same order, otherwise it is a "bad homography". In addition, we also check the cross-product of the rotational part of a homography such that any homography with a negative value will be rejected since they flip the basis vector.
- **The homography will not enlarge the scale of the original space by too much**: Following the same arrangement of corner points of a quadrilateral, the area for resultant points after applying a homography

should be maintained within a certain scale, otherwise it is bad. In our implementation, this cut-off value is 4.

- **Low values of perspectivity**: This is an empirical result, that the perspectivity value should always remain extremely low. These values are defined at the third row of a homography (see Eq. 2.9) and they should ideally stay close to $[0, 0, 1]$. More specifically, since we always work with image size within a 1000-unit pixel width, the value of the first and second entry should be as low as $0.0007 \sim 0.0001$ or possibly even smaller. For the third entry, the value should also empirically stay within the range of $[0.6, 1]$, otherwise it will cause enormous distortion.

Kalman Filter

Kalman filter is applied to the bounding box from the tracking result. The given data consists of four sets of pixel locations represented by the x- and y-coordinates, given as (X_1, X_2, X_3, X_4). Our goal is to perform data fusion on these data so that the trajectory motion of the bounding box is more stabilised during system run-time. Since our data structure contains four individual measurements (i.e., $X_i = (x_i, y_i), i = [1, 2, 3, 4]$) each of them is assumed to have a different model of noise affecting the raw measurements. Here, we utilised OpenCV implementation as the base code to build up our linear Kalman filter. The construction of a simple linear Kalman filter requires knowledge of these major components, namely the state vector, process model and measurement model.

We began by defining the state vector, denoted as X_k, where k represents the state at time k. It contains the positional data with its instantaneous velocity (i.e., the first-derivative) as follows:

$$X_k = (x, y, \dot{x}, \dot{y})^T \tag{5.7}$$

In our application, we have 4 sets of X_k that represent ROI of the tracking region.

The process model describes how a current measurement is obtained from its previous state, $k-1$. Matrix A is called the transition matrix; Bu_k, is the control input and is discarded in our application since there is no known control input, and the column vector, w, is the normal distribution process of noise with some covariance Q:

$$X_k = AX_{k-1} + Bu_k + w_{(k-1)} \tag{5.8}$$

Setting Bu_k to 0 and applying this equation to our problem we get the following:

$$\begin{pmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{pmatrix} + w_{k-1} \tag{5.9}$$

Where Δt is the time between measurements, which in our application it is set to 1.

Lastly, the measurement model relates the measurement of the current state with a matrix, H. Column vector v is the normal distribution of noise with covariance R:

$$z_k = Hx_k + v_k \tag{5.10}$$

Again, we can map this equation to our problem as follows:

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{pmatrix} \tag{5.11}$$

The overall framework of the Kalman filter consists of two stages: the prediction stage and the update stage. The prediction stage uses the previous measurement results from the last timestamp and the covariance obtained an a priori estimate P_k^-. This is then followed by an update stage in which P_k^- is

combined with a new measurement to refine the improved posteriori estimate of P_k.

Prediction stage	Correction stage
(1) Project the state ahead $\hat{x}_k^- = A\hat{x}_{k-1} + w_{(k-1)}$ (2) Project the covariance ahead $P_k^- = AP_{k-1}A^T + Q$	(1) Compute the Kalman gain $K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$ (2) Update state estimate $\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$ (3) Update the covariance $P_k = (I - K_k H)P_k^-$

The dynamic system of the Kalman filter varies by the diagonal covariance matrices Q and R, accordingly. In OpenCV these variables need to be setup upon the initialisation step. Even though there is no analytical approach for choosing the best combination of these variables, the contribution of effect can be summarised as follows:

- The process noise Q controls the overall uncertainty; when Q is large, the system will track large changes in data more closely than a smaller Q.
- The measurement noise R controls how much information from the measurement is used; when R is large, the measurements are treated as having a low accuracy and will not follow the measurement, while a smaller R means the estimate will follow the measurements more closely.

The choice between these matrices is more of empirical approach rather than analytical. In our implementation, $Q = diag(10^{-5})$ and $R = diag(10^{-4})$ where $diag(x)$ means set the diagonal elements to value x and 0 for other off-diagonal elements, and the initial value of $P_k^- = I$, where I is the identity matrix.

5.6. Experimental Results

In this section, we test the performance of the proposed AR pose tracker framework. The experimental setup is now described. We used the testing images taken by a mobile phone, where the images are of the same scene captured by up to 20 frames with similar weather conditions. We picked the first frame and the other frames at significantly different viewpoints. Our experimental goal is to demonstrate that our approach of fronto-parallel images helps to improve the matching effectiveness. We compared it with the most widely-used feature extraction technique SIFT for descriptiveness evaluation, followed by evaluation of the proposed AR framework.

Descriptiveness evaluation

The corresponding scene elements in fronto-parallel view have a similar appearance, so the resulting features should then suffer less from perspective distortions and show better descriptiveness. In this section, we would like to test how well two corresponding features can match with each other in terms of the Euclidean distance between their corresponding descriptors. The measurement takes each pair of matched correspondences, is described by a 128-dimensions SIFT descriptor. The L2 distance measurement was chosen as used in much of the literatures.

We choose SIFT as our candidate of comparison for better visualisation of descriptor space. A pair of features are considered matched if the overlap error of their corresponding regions is minimal and less than a threshold value. We adjusted this value to control the number of generated feature correspondences and calculated the average squared Euclidean distance[4] of their 128-dimension descriptors vector. The result is shown in Fig. 36. The

[4] AKAZE uses Hamming distance, which is in a way less-intuitive for representation and another reason we choose SIFT to compare with.

distance fluctuated between 27838 to 34770-unit length in the normal view while fronto-parallel was between 4422 to 6997-unit length, which indicates the feature correspondences in fronto-parallel view had much higher similarity matches (with a lower Euclidean distance between descriptors).

Fig. 35 – Descriptiveness evaluation: 1st row is the image pair in normal view while 2nd row is in fronto-parallel view;

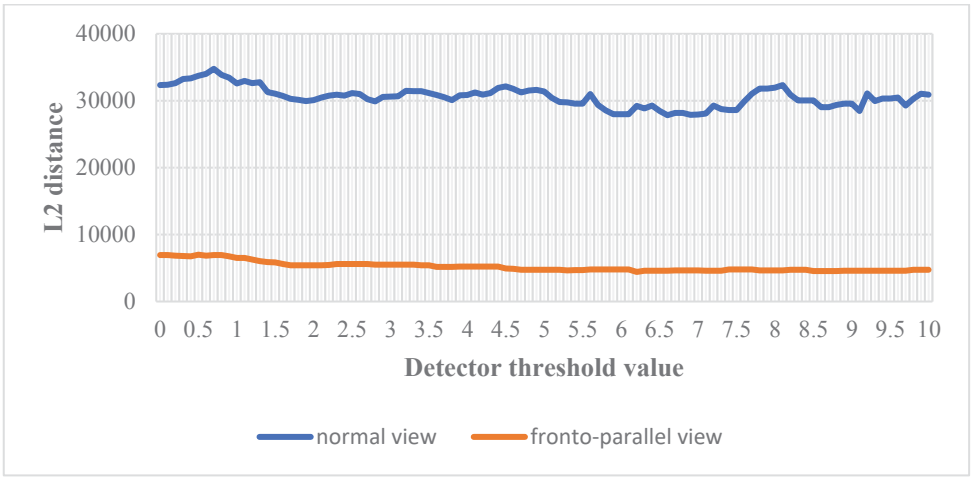


Fig. 36 – Descriptiveness evaluation: The vertical axis represents the squared Euclidean distance while the horizontal axis is the threshold value that controls the number of features. The blue line represents the normal view while the orange line represents the fronto-parallel view.

Tracker effectiveness

In the next part, we evaluate the correspondence of our tracker under the same set of scenarios: fronto-parallel view versus normal view. An instance of the screenshot is shown in Fig. 35. It is noted that features in the fronto-parallel view can handle large viewpoint changes by AKAZE more easily compared to the normal view as shown in Fig. 36.

The number of corresponding elements is essential for the estimation of homography and, hence, fronto-parallel generally produces better estimation since there are more matches found; for example, two graffiti images were detected and matched in the fronto-parallel view, which is in contrast to the original where only one was matched. In addition to the second result, the number of corresponding elements matched in the original view barely satisfied the minimum of eight points for homography estimation, and our result still pick up more matchings.

Fig. 37 – Experimental results on correspondences matching

5.7. Discussion and Conclusion

The experiment has shown the advantages of using a fronto-parallel view for obtaining feature points in homography pose estimation applications. These features are very robust against perspective distortion and viewpoint changes and are suitable for AR application. In terms of performance speed, the

baseline of OpenCV implementation of AKAZE is fast (about 20 FPS), while the matching process sometimes slowdown the overall speed. The initial prototype runs about 5 FPS, and in the later on optimised version we managed to increase the speed to 10-13 FPS.

The verdict of testing this approach is that the quality of estimation relies heavily on the initial rectification homography of the first frame, as well as the accuracy of inter-frame homography, and only works on the image taken in an urban environment (since this is our initial assumption in this thesis); however, we also found that, when matching against severely rectified images, they tend to fail drastically due to either no matches being found or too many bad matchings. Some examples include skyscrapers and high buildings structures. In these cases, due to severe distortion of the building façade from the camera viewpoint, the rectified views were severely shrunken which caused the tracker to fail to extract any feature points.

Chapter 6

Conclusion

In this thesis, we have developed an augmented reality system focused on markerless AR application in urban environments. Our primary assumption is that the urban environment mainly consists of building with a certain pattern of angle regularity revealed by windows and other man-made structures. The main contribution of this thesis is an efficient method to recover the planar homography pose automatically with respect to each detected building façade for augmented reality applications from a single image. Our approach exploits the observation that many buildings in an urban environment are portrayed by rectilinear structures. Our approach does not require a complete 3D representation of the environment, nor other types of data externally, and thus avoids inherent complexities associated with pose estimation.

The rectified fronto-parallel view also allows us to provide an authoring solution with an intuitive and easy-to-use interface for non-expert users. To demonstrate this principal, we have developed an authoring application which allows the user to load an arbitrary image and then apply the planar façade rectification to produce a rectified view of the input image for each detected façade. These rectified views allow users to easily select, position, and orientate AR graphical elements relative to the scene.

6.1. Future Works

Currently, the solution is implemented as a desktop application with MATLAB and C++. We are looking forward to moving our implementation of the rectification method into mobile phone platforms of either Android or iOS. This will give the advantage of greater portability for the authoring system as well as providing a better measurement of how well it can process on a mobile platform and, most importantly, allow the implementation to run on a wide variety of currently available mobile phone 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 rectification process to incorporate more robust approaches to handle failure cases, as discussed in Chapter 4.5. In addition, since our focus is on building façades, we also wish to improve our work for other types of planar surface, e.g. indoor environments.

As for the tracker component, the current implementation relies heavily on feature point detection to perform the fronto-parallel view generation. Apart from the severe perspective distortion which leads to failure, this approach will also likely fail on building façades with little or no texture, and we wish to revamp this deficiency in the future.

Reference

[1] P. F. Alcantarilla, ndez, A. Bartoli, and A. J. Davison, "KAZE Features," in *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, Berlin, Heidelberg, 2012, pp. 214-227.

[2] S. Mann, "`WearCam' (The Wearable Camera): Personal Imaging Systems for Long--term Use in Wearable Tetherless Computer--mediated Reality and Personal Photo/Videographic Memory Prosthesis," in *Proceedings of the 2Nd IEEE International Symposium on Wearable Computers*, Washington, DC, USA, 1998, pp. 124-.

[3] A. H. K. Kelly and B. Stacks, "Virtual reality; an interview with jaron lanier," pp. 108-120, 1989.

[4] I. E. Sutherland, "A Head-mounted Three Dimensional Display," in *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, New York, NY, USA, 1968, pp. 757-764.

[5] T. P. Caudell and D. W. Mizell, "Augmented reality: an application of heads-up display technology to manual manufacturing processes," in *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, 1992, pp. 659-669 vol.2.

[6] R. T. Azuma, "A Survey of Augmented Reality," *Presence: Teleoper. Virtual Environ.,* vol. 6, pp. 355-385, August 1997.

[7] (2015). *Doppler Lab*. Available: https://hereplus.me/ ,last visited on 09/2016

[8] J. Jung, S. Lee, H. Lee, H. S. Yang, L. Weruaga, and J. Zemerly, "Real-time sensor-fusion based indoor localization for mobile Augmented Reality," in *International Conference on Virtual Systems Multimedia (VSMM)*, 2014, pp. 184-191.

[9] A. Petit, E. Marchand, and K. Kanani, "Combining complementary edge, keypoint and color features in model-based tracking for highly

dynamic scenes," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4115-4120.

[10] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*: Cambridge University Press, 2003.

[11] H. Fuchs, A. State, E. D. Pisano, W. F. Garrett, G. Hirota, M. Livingston*, et al.*, "Towards performing ultrasound-guided needle biopsies from within a head-mounted display," in *Visualization in Biomedical Computing: 4th International Conference, VBC'96 Hamburg, Germamy, September 22--25, 1996 Proceedings*, h. K. H. H\"o and R. Kikinis, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 591-600.

[12] 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 and Graphics,* vol. 23, pp. 779-785, 1999.

[13] (2011). *Layar AR browser*. Available: http://www.layar.com/browser/ , last visited on 09/2016

[14] (2009). *Wikitude*. Available: http://www.wikitude.org/, last visited on 09/2016

[15] (2012). *ARNav*. Available: http://arnav.eu/, last visited on 09/2016

[16] Z. Wei and J. Kosecka, "Extraction, matching and pose recovery based on dominant rectangular structures," in *IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, 2003, pp. 83-91.

[17] H. Kato and M. Billinghurst, "Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System," in *Proceedings of the 2Nd IEEE and ACM International Workshop on Augmented Reality*, Washington, DC, USA, 1999, pp. 85-.

[18] J. Rekimoto, "Navicam: A Magnifying Glass Approach to Augmented Reality," *Presence: Teleoper. Virtual Environ.,* vol. 6, pp. 399-412, August 1997.

[19] J. Rekimoto, "Matrix: A Realtime Object Identification and Registration Method for Augmented Reality," in *Proceedings of the Third Asian Pacific Computer and Human Interaction*, Washington, DC, USA, 1998, pp. 63-.

[20] 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*, Washington, DC, USA, 2002, pp. 97-.

[21] M. Fiala, "ARTag, a fiducial marker system using digital techniques," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 590-596 vol. 2.

[22] (2011). *ArUco: a minimal library for Augmented Reality applications based on OpenCV*. Available: https://www.uco.es/investiga/grupos/ava/node/26, last visited on 09/2016

[23] (2007, 23 May). *OpenCV*. Available: http://opencv.org/, last visited on 03/2017

[24] G. Simon, A. W. Fitzgibbon, and A. Zisserman, "Markerless tracking using planar structures in the scene," in *IEEE and ACM International Symposium on Augmented Reality (ISAR)*, 2000, pp. 120-128.

[25] C. Zhuo and L. Xinyu, "Markless tracking based on natural feature for Augmented Reality," in *International Conference on Educational and Information Technology (ICEIT)*, 2010, pp. 126-129.

[26] Y. Du, Z. Miao, and Y. Cen, "Markless augmented reality registration algorithm based on ORB," in *International Conference on Signal Processing (ICSP)*, 2014, pp. 1236-1240.

[27] L. Ahyun, L. Jae-Young, S. H. Lee, and J. S. Choi, "Markerless augmented reality system based on planar object tracking," in *Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, 2011, pp. 1-4.

[28] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose Tracking from Natural Features on Mobile

Phones," in *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, Washington, DC, USA, 2008, pp. 125-134.

[29] Y. Xia and W. Zhou, "A tracking and registration method based on ORB and KLT for augmented reality system," in *Wireless and Optical Communication Conference*, 2013, pp. 344-348.

[30] S. Yonemoto, "A Video Annotation Tool Using Vision-based AR Technology," in *International Conference on Cyberworlds (CW)*, 2012, pp. 226-230.

[31] U. Neumann and S. You, "Natural Feature Tracking for Augmented Reality," *Trans. Multi.,* vol. 1, pp. 53-64, March 1999.

[32] I. Rabbi and S. Ullah, "A Survey on Augmented Reality Challenges and Tracking," *Acta Graphica,* vol. 24, pp. 29-46, 2016.

[33] J.-M. Morel and G. Yu, "ASIFT: A New Framework for Fully Affine Invariant Image Comparison," *SIAM J. Img. Sci.,* vol. 2, pp. 438-469, April 2009.

[34] M. Hamidia, N. Zenati-Henda, H. Belghit, and M. Belhocine, "Markerless tracking using interest window for augmented reality applications," in *2014 International Conference on Multimedia Computing and Systems (ICMCS)*, 2014, pp. 20-25.

[35] F. Dellaert, S. M. Seitz, C. E. Thorpe, and S. Thrun, "Structure from motion without correspondence," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, 2000, pp. 557-564 vol.2.

[36] B. Li, K. Peng, X. Ying, and H. Zha, "Simultaneous Vanishing Point Detection and Camera Calibration from Single Images," in *International Symposium on Advances in Visual Computing*, 2010, pp. 151-160.

[37] J. Fuentes-Pacheco, J. e. Ruiz-Ascencio, and n.-M. Rend\'o, Juan Manuel, "Visual Simultaneous Localization and Mapping: A Survey," *Artif. Intell. Rev.,* vol. 43, pp. 55-81, January 2015.

[38] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics \& Automation Magazine,* vol. 13, pp. 99-110, 2006.

[39] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Washington, DC, USA, 2007, pp. 1-10.

[40] r. Nist\'e, David, "An Efficient Solution to the Five-Point Relative Pose Problem," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 26, pp. 756-777, June 2004.

[41] R. Hartley and R. Gupta, "Computing matched-epipolar projections," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1993, pp. 549-555.

[42] G. Klein and D. Murray, "Parallel Tracking and Mapping on a Camera Phone," in *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, Washington, DC, USA, 2009, pp. 83-86.

[43] C. Yanpeng and J. McDonald, "Viewpoint invariant features from single images using 3D geometry," in *IEEE Workshop on Applications of Computer Vision (WACV)*, 2009, pp. 1-6.

[44] D. Liebowitz and A. Zisserman, "Metric Rectification for Perspective Images of Planes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998, p. 482.

[45] R. Szeliski, "Image mosaicing for tele-reality applications," in *WACV*, 1994.

[46] C. A. Rothwell, A. Zisserman, J. L. Mundy, and D. A. Forsyth, "Efficient Model Library Access by Projectively Invariant Indexing Functions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1992, pp. 109-114.

[47] A. Zaheer, M. Rashid, and S. Khan, "Shape from Angle Regularity," in *European Conference on Computer Vision*, 2012, pp. 1-14.

[48] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice (2Nd Ed.)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990.

[49] J. G. Semple and G. T. Kneebone, *Algebraic projective geometry*: Clarendon Press, 1979.

[50] R. Szeliski and H.-Y. Shum, "Creating Full View Panoramic Image Mosaics and Environment Maps," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1997, pp. 251-258.

[51] R. T. Collins and J. R. Beveridge, "Matching perspective views of coplanar structures using projective unwarping and similarity matching," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1993, pp. 240-245.

[52] C. Pirchheim and G. Reitmayr, "Homography-based planar mapping and tracking for mobile phones," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2010, pp. 27-36.

[53] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Manhattan-world stereo," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1422-1429.

[54] C. Rother, "A new approach to vanishing point detection in architectural environments," *Image and Vision Computing,* vol. 20, pp. 647-655, 2002.

[55] E. McClean, Y. Cao, and J. McDonald, "Single Image Augmented Reality Using Planar Structures in Urban Environments," in *Irish Machine Vision and Image Processing Conference (IMVIP)*, 2011, pp. 1-6.

[56] W. Zhang, "Video Compass," in *European Conference on Computer Vision*, 2002, pp. 657-673.

[57] F. Antonio, "Faster line segment intersection," in *Graphics Gems III*, K. David, Ed., ed: Academic Press Professional, Inc., 1992, pp. 199-202.

[58] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM,* vol. 24, pp. 381-395, 1981.

[59] N. H. Cho, Q. Wu, J. Xu, and J. Zhang, "Content Authoring Using Single Image in Urban Environments for Augmented Reality," in *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2016, pp. 1-7.

[60] T. S. H. Shao and L. V. Gool, "Zubud-zurich buildings database for image based recognition," ed, 2004.

[61] Z. Zhang, A. Ganesh, X. Liang, and Y. Ma, "TILT: Transform Invariant Low-rank Textures," *CoRR,* vol. abs/1012.3216, 2010.

[62] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces," in *British Machine Vision Conf. (BMVC)*, Bristol, UK, 2013.

[63] P. Perona and J. Malik, "Scale-Space and Edge Detection Using Anisotropic Diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 12, pp. 629-639, July 1990.

[64] X. Yang and K.-T. Cheng, "LDB: An Ultra-fast Feature for Scalable Augmented Reality on Mobile Devices," in *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Washington, DC, USA, 2012, pp. 49-57.

[65] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a Local Binary Descriptor Very Fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI),* vol. 34, pp. 1281-1298, 2012.

[66] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vision,* vol. 60, pp. 91-110, November 2004.