Federal University of Bahia
State University of Feira de Santana

MASTER THESIS

A multi-view environment for markerless augmented reality

Caio Sacramento de Britto Almeida

Master in Computer Science – MMCC

Salvador
December 15th, 2014

MMCC-Msc-0007

CAIO SACRAMENTO DE BRITTO ALMEIDA

A MULTI-VIEW ENVIRONMENT FOR MARKERLESS AUGMENTED REALITY

Thesis submitted to the Master Program in Computer Science from Federal University of Bahia and State University of Feira de Santana, in partial fullfillment of the requirements for the degree of Master in Computer Science.

Advisor: Antônio Lopes Apolinário Júnior

Salvador
December 15th, 2014

Index card.

Sacramento de Britto Almeida, Caio

A multi-view environment for markerless augmented reality/ Caio Sacramento de Britto Almeida– Salvador, December 15th, 2014.

47p.: il.

Advisor: Antônio Lopes Apolinário Júnior.
Thesis (master)– Federal University of Bahia, Institute of Mathematics, December 15th, 2014.

1. Multi-view environment. 2. Augmented reality. 3. Computer graphics.
I. Apolinário Jr., Antônio Lopes. II. Federal University of Bahia. Institute of Mathematics. III A multi-view environment for markerless augmented reality.

CDD 20.ed. 123.45

APPROVAL SHEET

CAIO SACRAMENTO DE BRITTO ALMEIDA

A MULTI-VIEW ENVIRONMENT FOR MARKERLESS AUGMENTED REALITY

This thesis was considered worthy of acceptance of the master's degree in Computer Science and approved on its final form by the UFBA-UEFS Master Program in Computer Science.

Salvador, December 15th, 2014

Prof. Dr. Antônio Lopes Apolinário Júnior 1
Federal University of Bahia - Brazil

Prof. Dr. Michelle Ângelo 2
State University of Feira de Santana - Brazil

Prof. Dr. Rodrigo Silva 3
Federal University of Juiz de Fora - Brazil

ACKNOWLEDGEMENTS

I would like to thank my family for all the support during all those years, not only the time spent on the master program, but also during all my graduation. Namely, first I would like to thank my brother, Rodrigo, for being my inspiration on following this career; my twin sister, Thalita, for being my support of all times; my parents, for the great education that was given to me; my grandmother Lourdes for everything I learned from her all these years; my girlfriend Jéssica, for giving a new meaning to my life; and specially my advisor Antônio Apolinário, for trusting on me to make this work happen.

ABSTRACT

Augmented reality is a technology which allows 2D and 3D computer graphics to be aligned or registered with scenes of the real-world in real-time. This projection of virtual images requires a reference in the captured real image, which is often achieved by using one or more markers. But, there are situations where using markers can be unsuitable, like medical applications, for example. In this work, we present a multi-view environment, composed by augmented reality glasses and two Kinect devices, which doesn't use fiducial markers in order to run augmented reality applications. All devices are calibrated according to a common reference system, and then the virtual models are transformed accordingly too. In order to achieve that, two approaches were specified and implemented: one based on one Kinect plus optical flow and accelerometer data from augmented reality glasses, and another one based purely on two Kinect devices. The results regarding quality and performance achieved by these two approaches are presented and discussed, as well as a comparison between them.

Keywords: augmented reality, augmented reality glasses, kinect, transformation, optical flow, markerless

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

In this chapter I present the motivation, objectives and overview of this work.

INTRODUCTION

1.1 MOTIVATION

Augmented reality has benefited from the progresses of multimedia and virtual reality, making feasible new forms of interaction between humans and machines. Differently from virtual reality, that takes the user to a virtual environment, the augmented reality keeps the user on his physical environment and takes the virtual environment to the user's space, allowing the interaction with the virtual world, in a more natural manner and without needing training or adaptation (Tori; Kirner; Siscoutto, 2006). Many times, this interaction means merging virtual images with images captured from a real environment.

One of the greatest challenges on the field of augmented reality is to determine, in real time, which virtual image to be displayed, in which position and how it should be represented. In order to obtain an integration illusion between real objects and virtual objects, the generated object should be aligned with the three-dimensional position and orientation of the real objects (Placitelli; Gallo, 2011). This can be achieved by estimating the camera position.

On many situations, fiducial markers[1] are used (often this is due to the real time requirements of the augmented reality applications) (Azuma, 1997) and are drawn in a way that they can be easily identified. Those markers need to be placed on the target scene and can achieve great results using just a few computational resources. Figure 1.1 shows the usage of a fiducial marker and a three-dimensional object being projected over it.

However, besides requiring human interference on the scene, there situations where the usage of fiducial markers is not possible, feasible or comfortable for the target model. That is the case, for example, of medical applications on which this model is a patient. It's also possible to cite other limitations of fiducial markers, like, for example, occlusion (a virtual image could be not projected if the marker is not completely visible) and

[1]A fiducial marker or fiducial is an object placed in the field of view of an imaging system which appears in the image produced, for use as a point of reference or a measure.

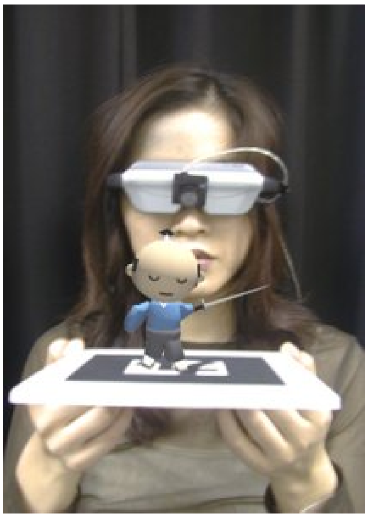

Figure 1.1 Fiducial marker used to represent a three-dimensional model over it (Artoolkit, 1999)

illumination (the intensity of light reflected by the marker could make it hard to be identified). Less common, there are approaches that replace fiducial markers (Carmigniani et al., 2011) (Gallo; Ciampi, 2009) by GPS, gyroscopes, accelerometers, cameras, among others (Azuma, 1997) (Azuma et al., 2001). These approaches have the advantage of not requiring human interference on the scene (to put a marker or to move it around).

Depending on the way that a user sees the mixed world, augmented reality can be classified on two ways. When the user sees the mixed world pointing his eyes straight to the real positions with optical scene or video, this augmented reality is called *immersive* or of *direct vision*. On the other hand, when the user sees the mixed world by some device, like a monitor screen or projector, not aligned with the real positions, this augmented reality is *non-immersive* or of *indirect vision* (Tori; Kirner; Siscoutto, 2006).

This work proposes a multi-view environment for augmented reality, of direct vision, composed by two Kinects (Microsoft, 2010) and augmented reality glasses, that allows an observer visualize, in real time, virtual images merged with real images from the target model, transformed to his viewpoint. In this approach, it's not intended to use any fiducial marker. Instead, it will be used a geometric approach based on the data captured by each Kinect. This proposed environment can be used, for example, on the medical field (real situations, education and training) or in other situation where a multi-view environment for markerless augmented reality is applicable.

1.2 OBJECTIVES

The environment proposed on this work aims to contribute to augmented reality applications where virtual images need to be merged with real images in real time, without using fiducial markers, and considering the viewing angle of the observer and the position of the target object.

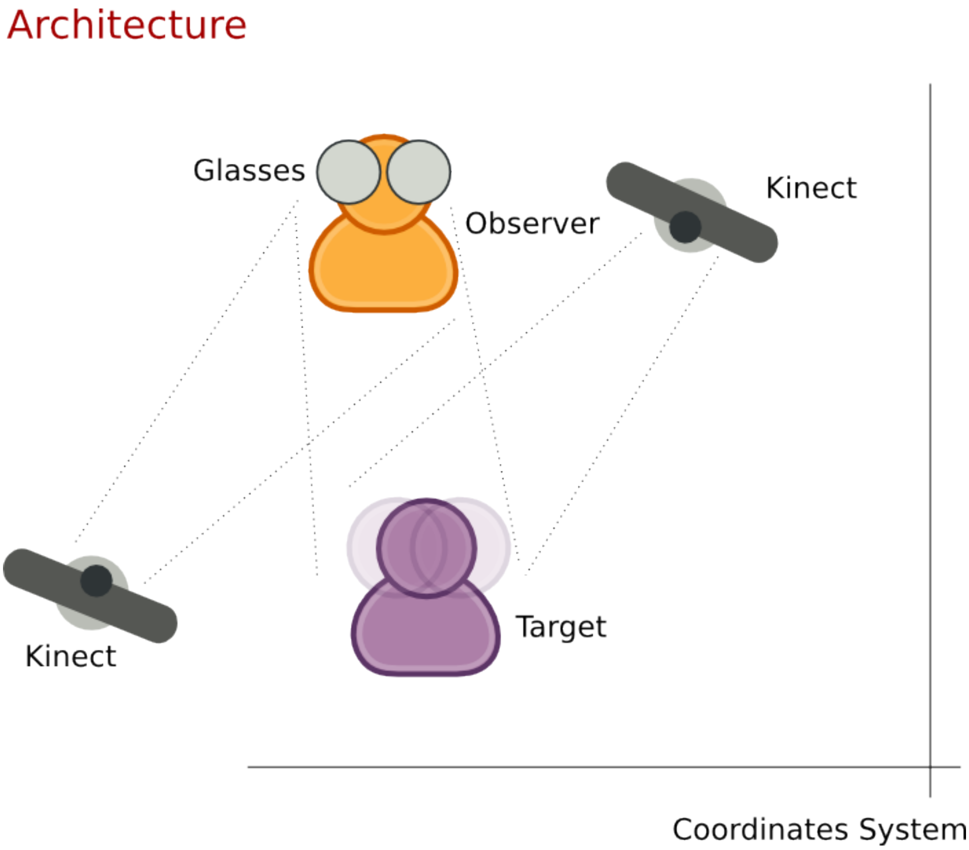

Figure 1.2 Global vision

1.2.1 Features

Based on the study of related works, it was defined the following features that define the scope of the first version of the augmented reality environment, graphically represented on Figure 1.2.1:

- There are two main elements on the environment, the observer and the target model;
- Observer and target model are positioned one in front of the other, with some restriction of minimum and maximum distance;
- The observer sees the combination of a real image with a pre-defined virtual image over the target model, through the augmented reality glasses;
- The observer and target model can move their heads, in a way that the virtual image adapts itself in real time to fit the new viewing angles;
- No fiducial marker is used.

The elements presented on this environment are the following:

- Observer - He is the user of the system that wears a pair of augmented reality glasses and is positioned in front of the target model (which can be a human being, for example, or a static object). On a medical context, the observer would be a medical specialist, responsible for observing the patient and responsible for analyzing the combination of the real image (part of the patient body) with a virtual image (from magnetic resonance imaging or computed tomography). The observer is able to move his head.

- Augmented reality glasses - The augmented reality glasses are wore by the observer and have two cameras. The images captured from the target model will be merged with virtual images and displayed on the two lenses of these glasses. On one of the approaches implemented by this work, the observer motion is determined from sensors present on the glasses: accelerometer, magnetometer and gyroscopes. Based on sensor data, it's possible to determine the variation on the orientation of the glasses, and so it's possible to know how much the observer has moved his head. This calculation returns values that define motion on longitudinal axis, vertical axis and lateral axis. The virtual image should be reprojected in real time according to those movements.

- Target model - The target model (a human being, for example), is placed in front of the observer and doesn't use any kind of fiducial marker. The main goal is that the virtual image is placed over the target model. In order to calculate where and how the virtual image should be displayed, it's necessary to identify the position and orientation of the real object relative to the observer. This is done based on two sensors placed on the environment, where the first one captures data from the observer and the second one captures data from the target model.

- Sensors - Two sensors are placed on the environment and capture data from the observer and the target model (one for each). The information captured by the target model's sensor contains its model, that will be merged with the virtual image.

Each device presented on this multi-view (glasses and sensors) environment has its own coordinate system, but all information must be converted to a global coordinate system.

Since there are two sensors and one pair of augmented reality glasses, another objective is to implement two different approaches: one that uses the augmented reality glasses to determine the observer's pose (based on data from accelerometer and magnetometer) and another one that uses a second Kinect device to determine the observer's pose based on a reconstruction of his model.

1.3 THESIS OVERVIEW

The next chapter "Conceptual primer" describes some basic theory needed. It also describes the hardwares that are used by this work and how to calibrate them. After that, the "Related work" chapter presents some works related to this one, divided by subject.

The third chapter, "Solution architecture", after the theory and related works were presented, explains the steps performed in order to implement the objectives of this work. The results of this implementation are presented on the chapter later, "Results", and finally the conclusions about those results are presented on the last chapter, "Conclusions", where possible future works are also listed.

Chapter 2

In this chapter I present the main concepts behind this work.

CONCEPTUAL PRIMER

2.1 AUGMENTED REALITY

Augmented reality was born on the decade of 1990, to merge a virtual image or virtual environment with a real image or real environment. But only from the 2000s it became more popular, due to lower costs of hardware and software devices, and ready to be used on tangible and multimodal (voice, touch, gesture, etc.) applications (Kawashima et al., 2001). It can be considered the mixing of real and virtual worlds at some point of the continuum reality-virtuality, that connects completely virtual environments to completely real environments (Milgram et al., 1994), like shown on Figure 2.1. It can also be considered a system that completes the real world with virtual objects, in a way that they seem to exist on the same space, respecting the following features:

- Real objects are mixed with virtual objects;
- Execution is interactive and on real time;
- Virtual and real objects are aligned;
- Applicable to all human sensory systems, including auditory, olfactory and somatosensory (Azuma et al., 2001).

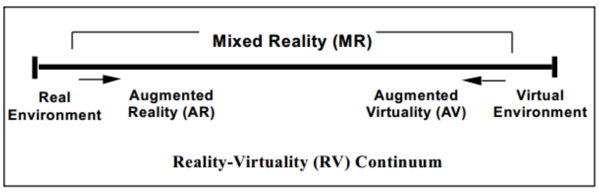

Figure 2.1 The augmented reality is localized between the extremes of the reality-virtuality continuum

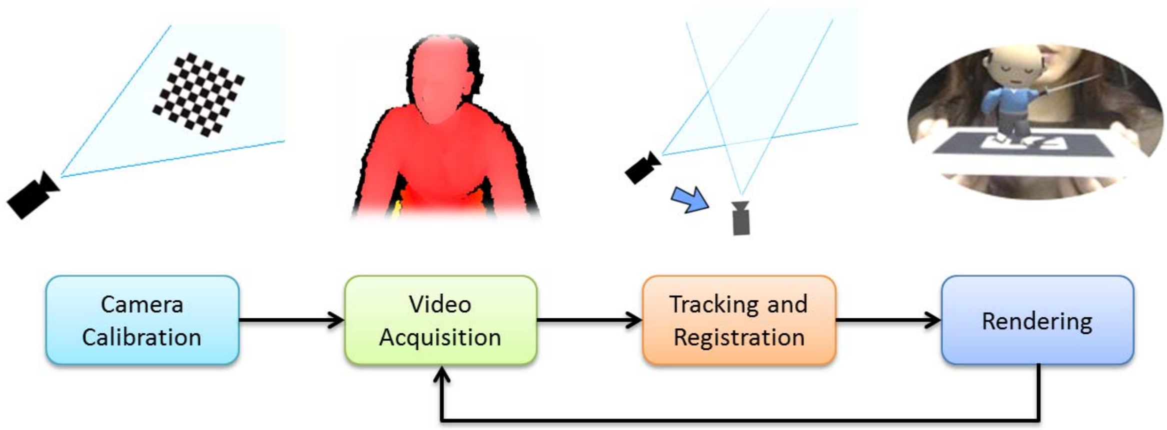

Figure 2.2 The typical pipeline of an augmented reality application (Placitelli; Gallo, 2011)

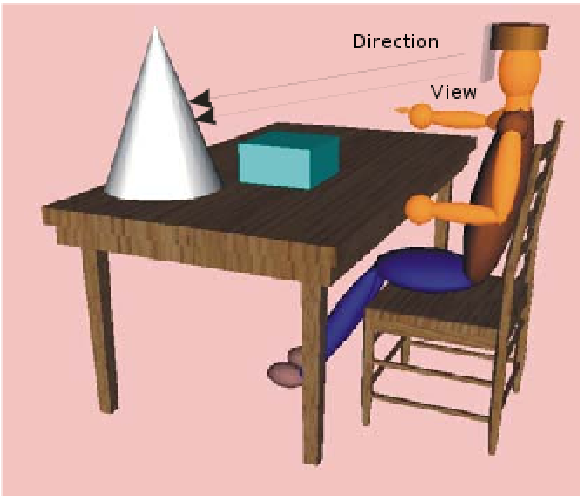

Figure 2.3 Immersive augmented reality

A typical augmented reality system, that uses a single RGB video feed for tracking and displaying, has basically four steps on its pipeline, as shown on Figure 2.1: camera calibration, video acquisition, tracking and registration, and rendering.

From a human-computer interaction perspective, the augmented reality can be considered a new way of interaction between humans and computers, and, on this aspect, it can be classified as direct vision or indirect vision.

2.1.1 Direct or indirect vision

Augmented reality is classified as *direct vision* or *immersive* when the user sees the mixed world pointing his eyes straight to the real position of the objects of interest, like shown on Figure 2.1.1.

On the direct vision, images from the real world can be seen with the naked eye or brought by video, while the generated virtual images can be projected on the eyes, on

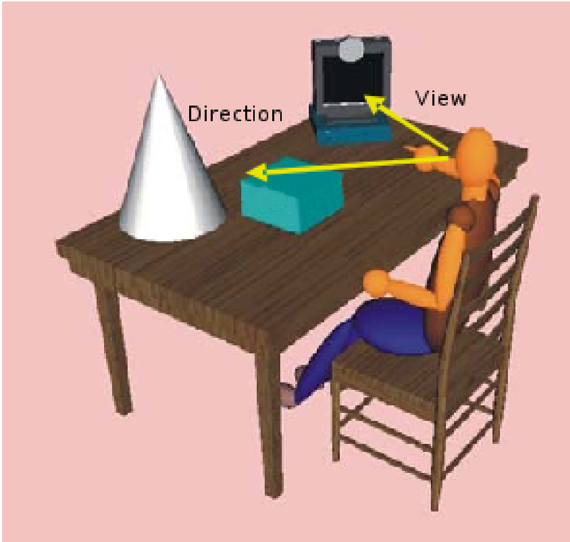

Figure 2.4 Non-immersive augmented reality

the real scenario or mixed with the real world video. Immersive augmented reality can be implemented with optical helmets, for example (Tori; Kirner; Siscoutto, 2006).

On the other hand, non-immersive augmented reality (indirect vision), happens when the user sees the mixed world through other output device, like a monitor screen, for example, not aligned with the real positions, as shown on Figure 2.1.1. On this kind of vision, real and virtual images are merged and displayed as video to the user. It can be achieved by using cameras and projectors (Tori; Kirner; Siscoutto, 2006).

2.1.2 Markers

In order to identify the position where a virtual image should be rendered, two main approaches can be applied by augmented reality applications: one is to use fiducial markers, the other is not use them.

2.1.2.1 Fiducial markers Fiducial markers are often implemented as square white cards with a black symbol printed (or drawn) on it, easy to be recognized, working like a barcode or QR code. Computer vision techniques are used to calculate the position of the real camera and its orientation relative to the markers, in a way that virtual objects can be projected over them.

Fiducial markers can assume other shapes besides a square card. The Figure 2.1.2.1 shows an example of a fiducial marker used on medical applications, similar to a sticker, which is fixed on the patient's skin. On these applications, augmented reality can be used for visualization and training on surgeries. It's possible to collect patient's data in real time, by using non-invasive sensors as the ones used for magnetic resonance imaging and computed tomography. This dataset can be merged in real time with the real image of the patient (Azuma, 1997).

Figure 2.5 Example of fiducial marker used by medical applications

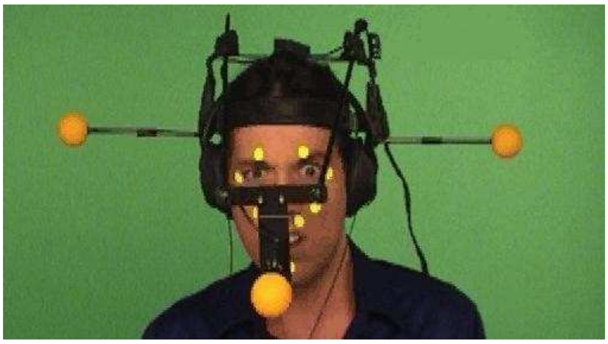

Figure 2.6 Example of fiducial marker for motion capture

On motion capture systems, other kinds of markers can also be used, like, for example, the one described on (Schoo; Mukundan, 2006) and represented on Figure 2.1.2.1. In order to estimate the pose, it uses a single camera and three spherical fiducial markers, which are also reflexive. Two of those markers are placed at the left and right sides of the actor, perpendicular to his ears. The third marker is placed on the same height as the other two, but in front of the actor. The markers have a color close to orange for high reflectance (in order to be easily recognized) and are supported by a structure made of carbon fiber (which is light).

Disadvantages of fiducial markers are listed on (Dolz, 2012), for example, they are invasive (need to be placed on the scene or fixed on the target object), they have limited interaction (can't be moved around so much because it needs to still be visible and recognizable) and need to be printed or drawn before using and stored for future usage.

2.1.2.2 Markerless Markerless augmented reality means that fiducial markers are not used on the scene. Instead, features from the target scene need to be used in order to estimate the camera pose and objects orientations.

Target detection based on computer vision has been extensively studied and successfully applied on augmented reality applications. On related computer vision literature, geometric primitives can be used for pose estimation, on most cases, points, segments, lines, edges of edge points, cones, cylinders, or a combination of two or more of those features.

Markerless augmented reality has the advantage of using parts of the real environment as targets and can even extract information from the environment to be used by the augmented reality system (Dolz, 2012).

Avoiding markers leads to a much more effective augmented reality experience, but requires the implementation of several image processing or sensor function techniques, resulting in more complex algorithms and in higher computational resources (Shumaker, 2011).

2.2 CAMERAS

Many tasks on augmented reality deal with an imaging device. Usually this imaging device is a camera, which performs a mapping from a 3D world to a 2D image (Hanning, 2011). The problem called *camera calibration* is the one that handles the determination of the parameters for this mapping.

2.2.1 Calibration

Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images. Much work has been done, by both the photogrammetry community and the computer vision community. Those techniques can be divided roughly into two categories: photogrammetric calibration (performed by observing a calibration object whose geometry in a 3-D space is known with very good precision) and self-calibration (does not use a calibration object, just move the camera in a scene and get its parameters) (Zhang, 2000).

Camera calibration means to determine the camera model parameters which fit best to the observed behavior of the actual camera (Hanning, 2011). So, it's necessary to measure the distance of an observation to a given camera model. The determination of the optimal camera mapping concerning each of these distance functions defines a non-linear optimization problem, and as such, it depends on the initial value.

Ordinary cameras are very often modelled as pinhole cameras. A pinhole is an imaginary wall with a tiny hole in the center that blocks all rays except the ones that pass through this tiny center hole (Bradski; Kaehler, 2008). Unfortunately, a real pinhole is not a good way to make images because it does not gather enough light for rapid exposure. This is why human eyes and cameras use lenses to gather more light than what would be available at a single point. This way, the simple geometry of the pinhole camera model is not enough and it also introduces distortion of the lens itself.

The camera coordinate system (CCS) is a Cartesian coordinate system defined by the principal plane: The x-axis and y-axis of the CCS determine the principal plane, the z-axis is given by the optical axis. The optical center of the lens determines the origin (0, 0, 0) of the CCS. Thus, in the camera coordinate system, the principal plane becomes $z = 0$ (Hanning, 2011).

Camera calibration is usually split up into two distinct parts, the intrinsic and extrinsic parameters, which will be covered on the following section.

2.2.2 Parameters

The process of camera calibration gives us both a model of the camera's geometry and a distortion model of the lens. Both compose the intrinsic parameters of the camera.

2.2.2.1 Intrinsic parameters The intrinsic parameters are those that describe the internal geometry of the camera and consist of the focal length, the location of image center in pixel space, the pixel size in the horizontal and vertical directions and radial and tangential distortions (Malik, 2002). These values are needed to help to describe imperfections in the lens of the camera and give a mapping from camera reference frame to the image plane. The intrinsic parameters depend only on the camera itself, and so they just need to be found once, regardless the environment changes or not (no need for recalibration due to environmental changes). The location of the image center and the pixel size make it possible to link image coordinates (x_{im}, y_{im}) in pixels, to the respective coordinates (x, y) in the camera coordinate system (Malik, 2002):

$$x = -(x_{im} - o_x)s_x \ y = -(y_{im} - o_y)s_y$$

Where (o_x, o_y) define the pixel coordinates of the principal point and (s_x, s_y) define the size of pixels in the horizontal and vertical directions, respectively.

2.2.2.2 Extrinsic parameters The extrinsic parameters, on the other hand, represent the viewpoint of the camera by a rigid transformation, which describes its position and orientation (Bajramovic, 2010). This part of the model is independent of the camera itself. The according parameters are called extrinsic, as they describe the relation between the camera and the world.

According to (Tillapaugh; Engineering, 2008), the extrinsic parameters are those that are dependent on the environment. To relate an object's coordinate system to the world's coordinate system, a translation and a rotation matrix are needed. Therefore, the extrinsic parameters consist of these two matrices, so that a mapping from the world coordinate system to the camera reference frame can be found. Since the extrinsic parameters for the camera explain how the camera relates to the environment, if the camera changes position, the parameters have to be recalculated (differently from the intrinsic parameters as explained on the previous section).

The relation between world coordinate system and the camera reference frame can be described by the equation:

$$X_c = R_c \times X + T_c$$

Where X is a 3x1 vector that represents a point on the world coordinate space, R_c is the extrinsic rotation matrix, T_c is the extrinsic translation matrix and X_c is a 3x1 vector in the camera reference frame. The rotation matrix is build from the combination of three single-axis rotation matrices:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & sin(\theta) \\ 0 & -sin(\theta) & cos(\theta) \end{bmatrix}$$

$$R_y(\phi) = \begin{bmatrix} cos(\phi) & 0 & -sin(\phi) \\ 0 & 1 & 0 \\ sin(\phi) & 0 & cos(\phi) \end{bmatrix}$$

$$R_z(\omega) = \begin{bmatrix} cos(\omega) & sin(\omega) & 0 \\ -sin(\omega) & cos(\omega) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The rotation matrix R has the property that its inverse is its transpose, hence $R^T R = RR^T = I$, where I is the identity matrix consisting of 1s along the diagonal and 0s everywhere else.

The translation vector T is how it's possible to represent a shift from one coordinate system to another system whose origin is displaced to another location; in other words, the translation vector is just the offset from the origin of the first coordinate system to the origin of the second coordinate system (Bradski; Kaehler, 2008). So, in order to shift from a coordinate system centered on an object to one centered at the camera, the appropriate translation vector is simply $T = origin_{object} - origin_{camera}$. Thus, a point in the object (or world) coordinate frame P_o has coordinates P_c in the camera coordinate frame: $P_c = R(P_o - T)$.

The extrinsic parameters can be represented by a final matrix called *homogeneous matrix*. It looks as follows:

$$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

So, in order to convert from a world origin coordinate to a camera orientation coordinate system, only a single homogeneous matrix needs to be used.

2.2.3 Calibration approaches

There are multiple ways to calibrate cameras. The most common way to perform the calibration is to have multiple points that have known relationship to each other in the world's coordinate system captured in an image from the camera (Tillapaugh; Engineering, 2008). As explained previously, for the intrinsic parameters it does not matter how the points are related to the world coordinates, it's only important how they relate to each other. Such parameters consist of the horizontal and vertical focal lengths and the principal point of the camera and the skew. The skew is generally assumed to be zero (Furht, 2011) for many cameras.

Various techniques were created to obtain accurate mappings, including vanishing points for orthogonal directions and calibration from rotation purely (Medioni; Kang, 2004),

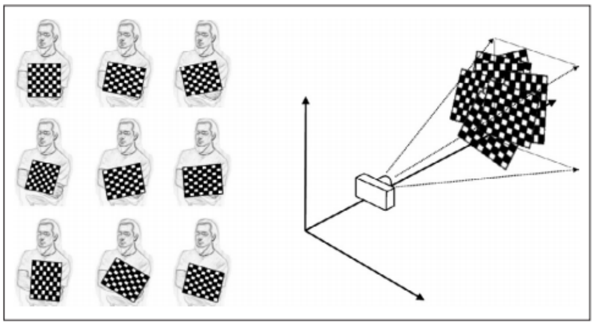

Figure 2.7 Images of a chessboard being held at various orientations (left) provide enough information to completely solve for the locations of those images in global coordinates (relative to the camera) and the camera intrinsics (Bradski; Kaehler, 2008)

features extraction which can be lines or points, single-image or multiple-images configurations, different camera models, linear and non-linear algorithms, among many others (Armangué; Salvi; Batlle, 2000) (Clarke; Fryer, 1998). Self-calibration is indeed feasible with simple image correspondences among frames, yet the participation of control points produces more robust calibration results, in closer agreement with object space constraints (Douskos; Kalisperakis; Karras, 2007).

The works by (Tsai, 1986) and (Zhang, 2000) are extensively referenced and propose closed form solutions for the estimation of intrinsic and extrinsic parameters using 3D and 2D calibration patterns respectively. Camera calibration is a much discussed topic but the lack of robust algorithms for features detection makes harder the construction of automatic calibration process (Laureano; Paiva; Silva, 2013). Calibration pattern recognition is a hard task, where the lighting problems and high level of ambiguities are the principal challenges. For this reason, the algorithms often require user intervention for a reliable detection of the calibration points.

There are some tools available for automatic camera calibration. Two of the most popular ones are The Bouguet MatLab Toolbox (Bouguet, 2008) and the OpenCV library (Bradski, 2000). The former is an application that asks the user to define four extreme points that represent the area where an algorithm searches for the calibration points, given the number of rows and columns of the pattern. The latter is a very popular computer vision library that offers an automatic way to detect chessboard patterns in images by the *findChessboardCorners()* function. The method performs successive morphological operators until a number of black and white contours are identified, subsequently the corners of the contours make up the calibration point set. The pattern is recognized only if all rectangles are identified. Figure 2.2.3 shows how a person can use a chessboard in order to calibrate a camera using the OpenCV library.

As stated by (Arca; Casiraghi; Lombardi, 2005), the well-known algorithms for chessboard detection proposed by (Tsai, 1986) and (Zhang, 2000) can achieve good calibration results since both of them are based on an initialization procedure that requires the precise corners positions of a calibration pattern (a chessboard, for example). Although the pattern to be detected is generally a simple object, the detection of its corners at sub-pixel precision is a very difficult task to be solved under uncontrolled acquisition conditions. At the state of art, most of the presented methods determine the positions of the chessboard corners by means of a two step processing scheme. At first the corners are detected with pixel precision, by means of edge detection methods; to increase the detection accuracy, the second step modifies their positions by the interpolation of the found borders. Those methods rarely reach a high accuracy, since the interpolation provides just a guess about the precise corners positions. Some authors try to improve the accuracy in the later steps of camera calibration, which increases computational cost, while others, like (Arca; Casiraghi; Lombardi, 2005), does not make any assumption on the orientation and scale of the chessboard and try to first find the chessboard, then determine the size of the squares and later on find the corners by using a simple statistical model.

2.2.4 Multi-view

Calibrating a multi-camera system accordingly means estimating the intrinsic and extrinsic parameters of all cameras (Bajramovic, 2010). Actually it is mainly concerned with estimating extrinsic parameters, since most approaches first compute intrinsic parameters individually for each camera. Extrinsic parameters are subsequently estimated given the intrinsic parameters.

The calibration task is usually formulated such that the world coordinate system may be chosen arbitrarily, which amounts to calibrate up to a rigid transformation of the whole system. Furthermore, if no metric measurements are used, the scale of the multi-camera system cannot be determined. This is the case if only correspondences between the cameras are used as input (of the extrinsic calibration) (Bajramovic, 2010).

According to (Hanning, 2011), the calibration of a stereo camera system is more than calibrating two camera separately: An additional constraint for the stereo camera system can be applied, since a calibration target is observed by both cameras.

So, in theory, calibration of a multi-camera system is possible by individually calibrating intrinsic and extrinsic parameters of all cameras involved with respect to the same world coordinate system using classical calibration methods. This just requires that all cameras observe a common calibration object or that a calibration object is moved to precisely known positions such that every camera can observe it. The first approach imposes very strict limitations on the allowable relative camera poses (relative orientation and position) or requires a very large calibration object, which is generally complicated (Chen; Davis, 2000).

Once cameras are calibrated, they are ready to register objects from a mixed reality scene. In general, traditional methods can be classified into three types: sensor-based methods, image-based methods and hybrid methods (Shumaker, 2011). The first two will be covered in the next sections; the third, which is also applied by this work, won't be covered here since it's basically a combination of the first two methods.

2.3 SENSOR-BASED REGISTRATION APPROACH

Three-dimensional (3D) reconstruction of an environment is an important problem that has received much attention in past decades. It can be defined as the process of capturing the shape and appearance of real objects (Pati, 2012). Recovering 3D surface structure of an object has been a central issue in computer vision, however due to high precision requirements of the registration process, no sensing device, alone, in the past, has achieved the results required by most augmented reality applications (Vallino, 1998). But on the last years, with the launch of inexpensive RGBD sensors (notably the Kinect), applications based on sensor registration became cheaper and more feasible. In the past, most augmented reality applications relied on sensors, such as magnetic, mechanical or inertial, but on the next sections only the Kinect will be covered.

2.3.1 Kinect

Kinect (Microsoft, 2010) is a new and widely-available sensor platform that incorporates a structured light based depth sensor. It can be classified as an RGB-D camera, since it comes with not only an RGB camera, but also a depth sensor. While the quality of this depth map is generally remarkable given the cost of the device, a number of challenges still remains (Newcombe et al., 2011): the depth images can contain a number of "holes", when depth reading was not possible, due to certain materials or scene objects that don't reflect infra-red (IR) light, very thin or small structures or surfaces at glancing incident angles. Data can also be missed if the device is moved fast.

The depth sensing system of Kinect consists of two parts: the IR laser emitter (which creates a known noisy pattern of structured IR light) and the IR camera. The depth sensing works on a principle of structured light. There's a known pseudo-random pattern of dots being pushed out from the camera. These dots are recorded by the IR camera and then compared to a known pattern (Kramer et al., 2012). The disturbances are known to be variations in the surface and can be detected as closer or further away. The fact that light matters brings three main problems:

- The wavelength must be constant (Kinect handles it internally)
- Ambient light can cause issues
- Distante is limited by the emitter strength

Within the sensor, there is a small heater/cooler that keeps the laser diode at a constant temperature. This ensures that the output wavelength remains as constant as possible, given variations on power and temperature. Ambient light is the bane of structured light sensors, but there are measures to mitigate this. One is an IR-pass filter that prevents stray IR in other ranges from blinding the sensor. However, Kinect doesn't work well in places with strong sunlight. Sunlight's wide band IR has enough power on Kinect's IR sensor range in order to blind it (Kramer et al., 2012).

The RGB camera collects 30 frames per second of actual real time events at a 640x480 resolution (Jean, 2012). The Kinect also has the option to switch the camera to high resolution, running at 15 frames per second (fps), which in reality is more like 10 fps at 1280x1024 pixels. Of course, the former is reduced slightly to match the depth camera; 640x480 pixels and 1280x1024 pixels are the outputs that are sent over USB. The camera itself possesses an excellent set of features including automatic white balancing, black reference, flicker avoidance, color saturation, and defect correction (Kramer et al., 2012).

The Kinect is calibrated at factory and has built in numbers for converting the disparity values to depth values, and also for mapping 3D points to the color camera. However, the calibration from factory uses a simple model where depth values are fast enough to be computed, but its accuracy can be improved by using other calibration methods (Jedvert, 2013).

One of those method is the one by (Burrus, 2014). It's a same-automatic way to calibrate the Kinect depth sensor and the RGB output to enable a mapping between them. It's based on a standard stereo calibration technique but the main difficult comes from the depth image that is not able to detect patterns on a flat surface. So, the pattern is created using depth difference. On the first step, the color camera intrinsics are calibrated using standard chessboard recognition. In order to get the intrinsic parameters for the depth camera, the corners of the chessboard on the depth image are extracted and stored (by hand). The raw depth values are integers between 0 and 2047, which can be transformed in meters using some fixed internal values from Kinect, as shown on the following algorithm:

function RAW_DEPTH_TO_METERS(raw_depth)
  if $raw\_depth \leq 2046$ **then return** $1.0/(raw\_depth * -0.0030711016 + 3.3309495161)$
  elsereturn 0
  end if
end function

Stereo calibration can also be performed by just selecting the corners of the chessboard on the color images. Depth pixels can be mapped to color pixels by undistorting RGB and depth images using the estimated distortion coefficients and then using a formula to project each pixel of the depth camera to metric 3D space.

All the information gathered by the RGB camera and the IR camera are available in two data streams provided by Kinect. One data stream makes it possible to build an RGB map and the other data stream makes it possible to build a depth map. Both maps without any calibration are useless, so it's necessary to perform a calibration called *depth registration* in order to get the correspondences between a pixel on the RGB map with another pixel on the depth map, which can be done by software but using built-in data stored on Kinect's firmware.

2.3.2 Registration

The depth maps generated by the Kinect contain discrete range measurements of the physical scene, thus this data can be reprojected as a set of discrete 3D points (or point cloud). Point clouds are sets of data containing collections of 3D vertices, often derived from an observation of a real world scene (Price, 2012). At a first glance, point clouds can be divided into two main classes: unorganized or organized. If the dataset is organized, it can be directly indexed or searched in a tree-basis in order to find an specific point and also its neighbors. This is very important because most of the point clouds operations depends on being able to compute properties of a point based on its neighborhood (or k-closest points). So, an unorganized point cloud should be first parsed into a tree structure in order to allow faster computations. This tree structure can be, for example, an array, an octree or a KD-tree.

Once a point cloud is organized, operations can be performed over it. Measure error is expected in any conversion from the real world to a digital structure. As stated previously, variations on the surface or scene objects materials or shapes, lighting, edges, among others, can lead to noisy point clouds, giving an inaccurate result. The next sections will cover three common operations over point clouds.

2.3.2.1 Filtering So, filtering is one of the first tasks to be performed over an early fetched point cloud. There are many filtering algorithms for this, but one of most powerful ones is the statistical outlier filter, which can solve these irregularities by performing a statistical analysis on the neighborhood of each point, and by discarding the ones that don't meet an specific criteria. In the one implemented by the Point Cloud Library (Library, 2013c), the outliers removal is based on the computation of the distribution of points to neighbors distances in the input dataset. For each point, the mean distance from it to all the neighbors is computed, and by assuming that the resulted distribution is Gaussian (with a mean and standard deviation), all points whose mean distances are outside the interval defined by the global distances mean and standard deviation can be considered outliers and removed from the dataset.

2.3.2.2 Normal estimation The normal vector, often simply called the "normal" to a surface is a vector perpendicular to it (Weisstein, 2014). Given a geometric surface, this is easy to be determined, however, this is not straightforward for point clouds, where estimation is necessary. Normal estimation at a point is an important task in order to determine whether points are part of the same object. There are many different methods for normal estimation, and one of the simplest ones is the one that approximates this problem by the problem of estimating the normal of a plane tangent to the surface, which becomes a problem of estimating the least-square plane fitting (Library, 2013a). Therefore the problem is reduced to an analysis of eigenvalues and eigenvectors of a covariance matrix generated from the nearest neighbors of the target point. Another simple method, given a point and its neighborhood, is to just average the normals created from the cross products of the vectors to all of the other points on the neighborhood (Price, 2012).

2.3.2.3 Segmentation Segmentation is the process that separates a set of points into the different objects that they represent. For example, if the point cloud is acquired from a room, the segmentation would determine which sub-sets of points from this point cloud represent a chair, a desk, and any other object present in the captured room. Approaches for this include derivative computation of the normal field (in order to determine when one object ends and another starts) or even pattern matching.

On Kinect, these operations can be performed in the following way:

- Each point P at position (m, n) on the point cloud can be computed by $P_{m,n}(t) = D_{m,n}(t) \times K_{m,n}$, where D is the depth array given by Kinect and K is the intrinsic field-of-view tensor;
- Normal vector at any point (m, n) is an approximation of $(P_{m+1,n} - P_{m,n}) \times (P_{m,n+1} - P_{m,n})$;
- Segmentation can be done based on color values (from RGB map) or depth values (from depth map)

Two consecutive point clouds, scanned at different times, can be aligned in order to calculate the transformation of the captured object.

2.3.3 Alignment

There also many methods for aligning two point clouds, but the Iterative Closest Point (ICP) (Besl; Mckay, 1992) is far the most implemented one, proposed on 1992 for the registration of 3D shapes, but it's also used to reconstruct surfaces of different scans, gather optimal path planning, and so on. The main goal of the algorithm is to minimize the difference between two point clouds, one called the *target* and another one called the *source*. The target point cloud is kept fixed, while the source point cloud is transformed in order to best match the fixed target point cloud. The algorithm iteratively revises the transformation (composed by rotation and translation) in order to minimize the difference between the source point cloud and the target point cloud, until it reaches a local minimum.

On this context, the interest is on applying the algorithm to point sets, however it was originally designed to also work with six other structures: line segment sets, implicit curves, parametric curves, triangle sets, implicit surfaces and parametric surfaces (Hajnal; Hill, 2014).

The algorithm has only two stages, and then iterates. On the first stage, the closest target point for each source point is identified. The second stage tries to find the least square rigid body transformation that relates these two point sets. The algorithm then iterates to redetermine the closest point set and continues until it reaches the local minimum match between the source and target surfaces, as determined by some threshold.

Regardless the representation of the source surface P, it is first converted to a set of points p_i. The target data remains in its original representation. As explained, the first stage means identifying the closest point on the target surface T for each point p_i on the source surface P. This is the point x in T where the distance between p_i and x is minimum:

$$d(p_i, x) = min_{x \in T} \|x - p_i\|$$

All the closest points (one for each p_i) are returned as new a set q_i. A least squares registration between the points p_i and q_i can be performed and so the set of source points

p_i can be transformed to a set of points p'_i using the rigid body transformations (composed by a rotation and a translation) that was calculated, and then the closest points can be identified again. The algorithm stops when the change in mean square error between two iterations goes below a threshold. Since the algorithm iterates to a local minimum closest to the starting point, it may not find the best solution, that's why the original authors propose to start the algorithm multiple times and then choose the minimum of the minima obtained (Besl; Mckay, 1992).

The algorithm can be optimized by storing the solutions at each iteration, for example. Actually many variations were proposed to the original version, mainly focused on performance improvements, like using KD-trees (Zhang, 1994), Lie group representation (Dong et al., 2014) or expectation maximization estimation for point set registration with noise (Liu et al., 2014).

2.3.4 Reconstruction

KinectFusion (Izadi et al., 2011) (Newcombe et al., 2011) is probably the most famous algorithm for reconstructing objects using the Kinect device. It is the state-of-art algorithm for real-time reconstruction and rendering of a real world scene, and is implemented on the PCL (Library, 2013b) library under the name of KinFu. In order for this implementation to work, a powerful processor and a graphics card with a CUDA-enabled Nvidia GPU are required, so it can run at at least 30 frames per second.

The algorithm was originally developed by Microsoft Research in 2011. It allows a user to reconstruct a three-dimensional scene in real-time and with good level of details by just moving the Kinect sensor around the real scene.

The input for the algorithm is a temporal sequence of depth maps as returned by the Kinect device. Since the original algorithm just uses depth maps and doesn't use any color information from the RGB sensor, it can in theory work in a completely dark environment. It runs in real-time, so it proceeds by using a frame per time as provided by the depth sensor. A surface representation is extracted from the current depth frame and a global model is refined by first aligning and then merging the new surface with it. The global model is obtained as a prediction of the global surface that is being reconstructed and refined at each iteration (Pirovano, 2012).

So, at each new frame, the new depth map obtained from the depth sensor is used as input for the algorithm. This depth map must be converted into a three-dimensional point cloud with vertex and normal information. This is done at different, layered resolutions, resulting in a number of images with different levels of details, which is originally defined in the algorithm as three. This is called *multi-resolution pyramid* (see Figure 2.3.4), on which the lower resolution layers are obtained by sub-sampling the higher resolution ones.

The depth map is converted into a three-dimensional point cloud by back-projection. In order to do that, it's used the internal calibration matrix that is stored in Kinect, that matches depth values to actual 3D coordinates. The normal of each point is estimated through cross-product of two vectors: the vector joining the chosen point and the one above it, and the vector that pass through the chosen point and the one at its right. In other words, since the points are arranged as depth pixels, if point (x, y, z) is picked up,

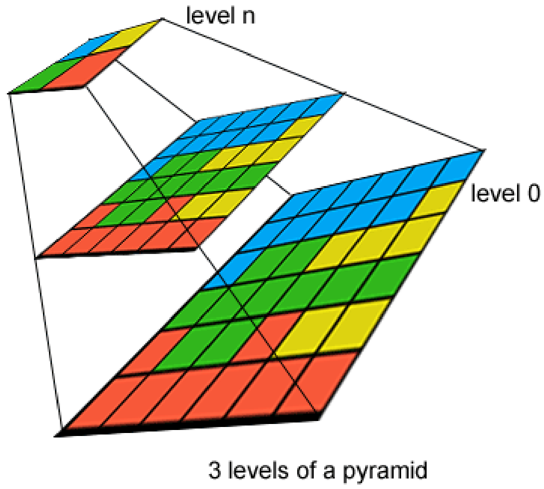

Figure 2.8 A multi-resolution pyramid with three levels (Pirovano, 2012)

points $(x, y+1, z)$ and $(x+1, y, z)$ will be used for normal estimation, where x and y are image coordinates and z is the depth value at these coordinates. At this point, there is a point cloud with normal and vertex information for each point with three distinct levels of details. This cloud is considered ordered since the points are arranged according to the pixels on the depth map.

The next stage is the alignment. The first point cloud is related to the model that was captured. From the second step on, the new point cloud is aligned to the current model and them merged to it in order to produce more refined model through iterative steps. The alignment is performed through the iterative closest point algorithm (ICP). A point-to-point approach is used here: for each point in the current point cloud, the closest one on the previous (global) point cloud is considered, and the distance between them is calculated. The final transformation at the step is the one that minimizes the total error between all point pairs. It's important to notice that KinectFusion doesn't use the standard ICP algorithm since it's too slow to be applied on a real-time context. The difference here is that is assumes that the changes between the current cloud and the previous one are small, which makes sense if the algorithm runs in real-time (for example, at 30 frames per second) and if the Kinect is moved slowly. The modified ICP back-projects the two clouds onto the camera image frame of the model and so they are considered to match if they fall on the same pixel. To make it even faster, the ICP iterations are performed at three resolutions. After a match is found, the modified ICP algorithm calculates the error between two points in a match by using a point-to-plane metric. Some iterations later, the modified ICP will generate a six degrees of freedom transformation matrix, composed by a rotation and a translation, which aligns the source point cloud with the target point cloud.

After that, comes the reconstruction itself. Once the transformation is found and this the current pose of the camera can be estimated, the new cloud is ready to be merged with the current model. In order to avoid losing details, the raw depth data is used instead of the filtered one. This step is based on a Truncated Signed Distance Function (TSDF). What this function does is extract the surface of objects in the scene and assign negative numbers to pixels inside objects or inside an area not yet measured, and positive numbers to pixels outside the surface. For pixels on the surface, this number is zero, for pixels outside the surface, this value is greater for pixels far from the surface and lower for pixels near the surface. TSDF is computed for the new cloud and merged to the current model's TSDF.

On the last step, the resulting TSDF is used to reconstruct the surface of the refined model, which is achieved through ray casting. It is performed from the global camera focal point by intersecting the zero level set of the TSDF. The refined ray-casted model will be used for the next step of the ICP alignment. The end result of the algorithm is a 3D surface representing the acquired scene.

Since the ICP make assumptions about small changes between two point clouds, sudden movements can make it converge to a wrong match, and then break the tracking and consequently the reconstruction. On the other hand, the algorithm is very robust for presence of dynamic objects on the scene, given enough iterations of it. (for example, if a desk is being captured on a room and someone walks in, this person will be removed from the reconstruction). This is due to the weighted average used during merging the TSDFs (Pirovano, 2012).

2.4 VISION-BASED REGISTRATION APPROACH

Registration based on sensors relies solely upon geometric information, for example, the spatial coordinates of the points in two clouds. As a result, environments that don't provide high geometric texture (for example, plane surfaces like a wall) can cause sensor tracking to fail. For such cases, a vision-based registration approach can be applied, by using the RGB information from the images (Peasley; Birchfield, 2013).

2.4.1 Optical flow

Optical flow is a pattern that describes the apparent motion of objects, edges or surfaces on a visual scene caused by the relative motion between that object and an observer (a camera or the human eye) (Burton; Radford, 1978). This concept was first introduced by the psychology in order to describe the visual stimulus provided to animals moving on the world (Gibson, 1950).

A fundamental problem in the processing of image sequences is the measure of the optical flow (or image velocity). The main objective is to compute an approximation to the 2D motion field - a projection of the 3D velocities of surface points onto the imaging surface - from spatiotemporal patterns of image density. Once computed, the measurements of image velocity can be used for many purposes, including object tracking (Barron; Fleet; Beauchemin, 1994).

The field of optical flow estimation is making big progresses as evidenced by the increasing accuracy of current methods on the Middlebury optical flow benchmark (Scharstein; Szeliski, 2002). After almost 30 years of research, these methods have obtained an impressive level of reliability and accuracy, by combining a data term that assumes constancy of some image property with a spatial term that models how the flow is expected to vary across the image (Sun; Roth; Black, 2014).

Differential methods belong to the most widely used techniques for optic flow computation in image sequences. They can be classified into local methods such as Lucas-Kanade technique or Bigun's structure tensor method, and into global methods such as the Horn/Schunck approach and its extensions. Often local methods are more robust under noise, while global techniques yield dense flow fields (Bruhn; Weickert; Schnörr, 2005).

Despite their differences, many of these techniques can be viewed conceptually in terms of three stages of processing (Barron; Fleet; Beauchemin, 1994):

- Pre-filtering or smoothing with low-pass/band-pass filter in order to extract signal structure of interest and to enhance the signal-to-noise ratio;
- The extraction of basic measurements, such as spatiotemporal derivatives (to measure normal components of velocity) or local correlation surfaces;
- The integration of these measurements to produce a 2D flow field, which often involves assumptions about the smoothness of the underlying flow field.

One of the most popular methods for computing the optical flow of an image is the Lucas-Kanade algorithm, which will be explained on the following section.

2.4.2 Lucas-Kanade algorithm

The Lucas-Kanade (Lucas; Kanade, 1981) algorithm is a differential method for computing the optical flow of an image (Peasley; Birchfield, 2013). The goal of Lucas-Kanade is to align a template image $T(x)$ to an input image $I(x)$ where $x = (x, y)^T$ is a column vector containing the pixel coordinates. If the Lucas-Kanade algorithm is being used to compute optical flow or to track an image from time $t = 1$ to time $t = 2$, the template $T(x)$ is an extracted sub-region (a window) of the image at $t = 1$ and $I(x)$ is the image at $t = 2$. Applications of the Lucas-Kanade algorithm range from optical flow and tracking to layered motion, mosaic construction and face coding. Numerous variations has been made to the original algorithm (Baker; Matthews, 2004).

The basic idea of the Lucas-Kanade algorithm is based on three assumptions (Bradski; Kaehler, 2008):

- Brightness constancy: a pixel from the image of an object in the scene does not change in appearance as it (possibly) moves from frame to frame. For grayscale images (although Lucas-Kanade can also be done in color) this means that it's assumed that the brightness of a pixel does not change as it is tracked from frame to frame;
- Temporal persistence: the image motion of a surface patch changes slowly in time. In practice, this means the temporal increments are fast enough relative to the scale of motion in the image that the object does not move much from frame to frame;

- Spatial coherence: neighboring points in a scene belong to the same surface, have similar motion, and project to nearby points on the image plane.

Mathematically, the goal of the Lucas-Kanade is to minimize the sum of squared error between the two images, the template T and the image I warped back onto the coordinate frame of the template (Baker; Matthews, 2004):

$$\sum_x [I(W(x;p)) - T(x)]^2$$

Where $W(x;p)$ denote the parametrized set of allowed warps, where $p = (p_1, ..., p_n)^T$ is a vector of parameters. The warp $W(w;p)$ takes the pixel x in the coordinate frame of the template T and maps it to the sub-pixel location $W(x;p)$ in the coordinate frame of the image I. In this context of optical flow computation, the warps $W(w;p)$ are the translations:

$$W(w;p) = \begin{matrix} x + p_1 \\ y + p_2 \end{matrix}$$

Where the vector of parameters $p = (p_1, p_2)^T$ is then the optical flow.

Warping I back to compute $I(W(x;p))$ requires interpolating the image I at the sub-pixel locations $W(x;p)$. The minimization is performed with respect to p and the sum is performed over all of the pixels x in the template image $T(x)$. Minimizing the expression is a non-linear optimization task even if $W(w;p)$ is linear in p because the pixel values $I(x)$ are, in general, non-linear in x. In fact, the pixel values $I(x)$ are essentially unrelated to the pixel coordinates x. To optimize the expression, the Lucas-Kanade algorithm assumes that a current estimate of p is known and then iteratively solves for increments to the parameters Δp, so the expression is (approximately) minimized:

$$\sum_x [I(W(x;p + \Delta p)) - T(x)]^2$$

With respect to Δp, and then the parameters are updated:

$$p \leftarrow p + \Delta p$$

These steps are iterated until the estimates for the parameters p converge.

The result of the algorithm is a set of optical flow vectors distributed over the image which give an estimation idea of the movement of objects in the scene, although some of those vectors will be erroneous (Rojas, 2014).

The Lucas-Kanade method can be applied in a sparse context because it relies only on local information that is derived from some small window surrounding each of the points of interest. The disadvantage of using small local windows in Lucas-Kanade is that large motions can move points outside of the local window and thus become impossible for the algorithm to find, which led to the development of the "pyramidal" Lucas-Kanade algorithm (Bouguet, 2000), one of many approaches that have been used to improve the convergence rate and reduce the likelihood of falling into a local minimum.

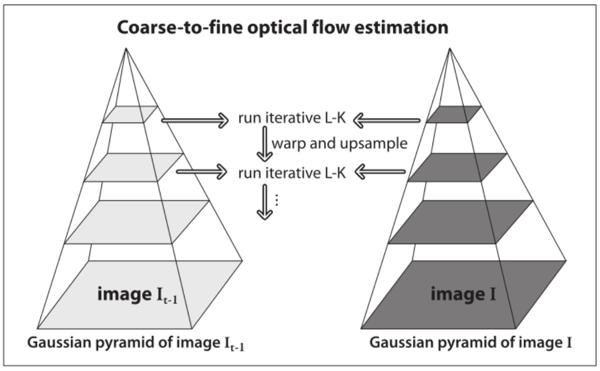

Figure 2.9 Pyramid Lucas-Kanade optical flow: running optical flow at the top of the pyramid first mitigates the problems caused by violating our assumptions of small and coherent motion; the motion estimate from the preceding level is taken as the starting point for estimating motion at the next layer down (Bradski; Kaehler, 2008)

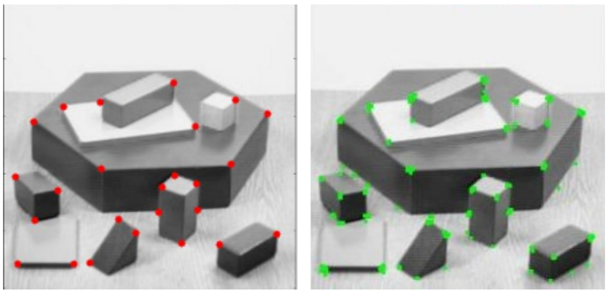

Figure 2.10 The same input image being used for features identification by Shi-Tomasi algorithm (red dots) and by Harris Corner Detector (green dots)

One component in many algorithms is a coarse-to-fine strategy. The most common approach is to build image pyramids by repeated blurring and downsampling. So, optical flow is first computed on the top level (fewest pixels) and then upsampled and used to initialize the estimate at the next level (Baker et al., 2011), as shown on Figure 2.4.2. Computation at the higher levels in the pyramid involves far fewer unknowns and so is far faster. The initialization at each level from the previous level also means that far fewer interactions are required at each level. Due to this, pyramid algorithms are usually faster than a single solution at the bottom level.

The features to be tracked by the Lucas-Kanade algorithm can be manually defined or identified from an algorithm, like the popular one proposed by Shi and Tomasi (Shi; Tomasi, 1994), called "good features to track", and implemented by the OpenCV library (Bradski, 2000). The idea of this algorithm is a modification of the Harris Corner Detection (Harris; Stephens, 1988), which basically replaces Harris' scoring function. Figure 2.4.2 shows the feature points on the same input image as identified by both algorithms.

So, the Lucas-Kanade algorithm, applied over a good set of feature points, is an efficient method for obtaining optical flow information at interesting points in an image and works for moderate object speeds.

Chapter 3

In this chapter I present some related works. Since it hasn't been found any work that uses a multi-view markerless augmented reality environment based on hybrid reconstruction, this chapter is divided into subjects related to the this work: usage of multiple Kinects, optical flow, markerless augmented reality, medical applications, multi-view environment, reconstruction, etc.

RELATED WORKS

3.1 AUGMENTED REALITY APPLICATIONS FOR MEDICINE

On the field of augmented reality, one of the first approaches was to track known patterns, that's how much fiducial markers work. The traditional ARToolKit (Artoolkit, 1999) is probably the most famous of them and augments virtual objects onto black and white square markers. Its advantage is simplicity and performance, since it works also with modest computers and commodity cameras. However, it doesn't support any kind of occlusion, since its fitting process is based on a four-lines approach that requires the marker to be fully visible by the camera.

Augmented reality has been widely implemented on medicine on the last years, for training, simulation, therapy and so on. Many works on this field are based on fiducial markers in order to calculate the camera's real position in relation to the marker's real position, like the system called ARBioMed (Bucioli; Jr.; Cardoso, 2008). The idea of this system is to represent a virtual, three-dimensional heart over a marker placed on the target user's chest. This virtual heart has its pulsation simulated according to a received signal. From this system, the operator user can define, through a user interface, a fixed pulse rate to be simulated on the heart. In order to set the size of the heart based on the dimensions of the user chest, two other fiducial markers are used, as shown on Figure 3.1. This architecture makes the implementation simpler, since it is based on ARToolKit (Artoolkit, 1999), a popular and straightforward framework for augmented reality based on fiducial markers. However, the markers pollute the augmented reality scene and offers a less comfortable experience to the target user, since the markers are invasive.

Another work on the medical field that is based on the traditional square fiducial markers is the motion capture system presented on (Damasceno; Lamounier; Cardoso, 2012), whose goal is to follow up physiotherapy exercises performed by patients. As shown on

Figure 3.1 ARBioMed running, with three fiducial markers (Bucioli; Jr.; Cardoso, 2008)

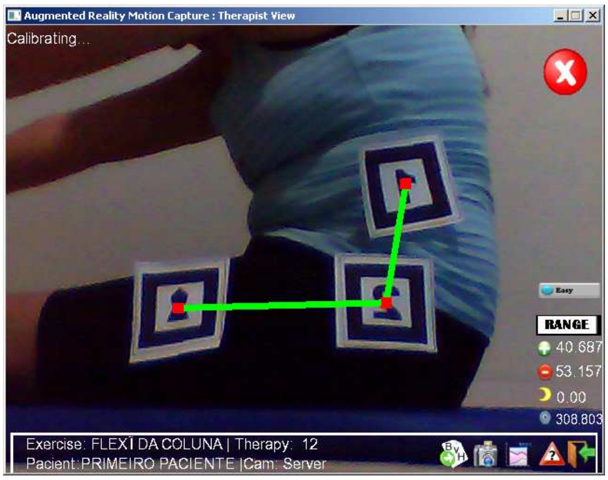

Figure 3.2 User under augmented virtual therapy (Damasceno; Lamounier; Cardoso, 2012)

Figure 3.1, fiducial markers are fixed on parts of the patient's body where motion will be captured. Based on trigonometric evaluation of the markers' position, it's possible to establish the range of motion that was executed.

Augmented reality applications on medicine are also applied for training, like the ANGELS system (Quero et al., 2014). The main objective of this project is to test the utility of augmented reality to train in prevention of risks at work in the health care sector. After conducting a requirements analysis in order to obtain a general framework for the priority risks in safety at work within the health care, a pilot study was carried out and preliminary results showed a good opinion of the users as well as favorable usability of the system.

Still on training, the work at (Vera et al., 2014) implemented augmented reality telementoring platform to overlay the instruments of a mentor onto the trainee's laparoscopic monitor, for surgical training. The study concluded that the designed platform was a more

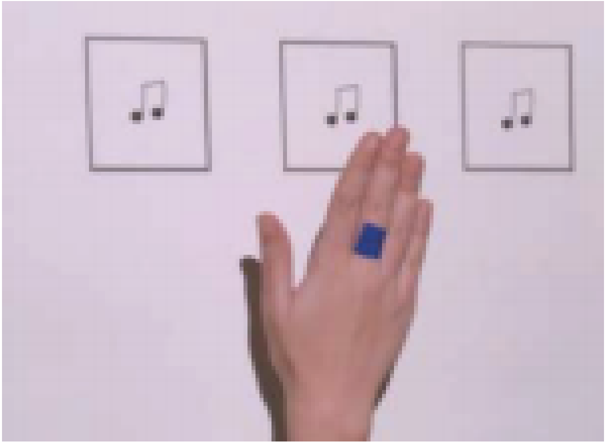

Figure 3.3 Patient with stroke performing an exercise to reach melody boxes (Hondori et al., 2013)

effective training technique in teaching laparoscopic skills to medical students compared to traditional methods, since it reduced the number of failed attempts and resulted in faster suture times.

On the therapy side, there is a work that developed a low-cost spatial augmented reality system that allows individuals with stroke to practice hand and arm movement exercises at home or at a clinic with minimal support of the therapist (Hondori et al., 2013). This setup uses a color marker, fixed on the patient's hand, to track his movement and check how accurate he performs. After that, a score is displayed to him. There are setups for some exercises, and one of them is displayed on Figure 3.1.

This study of related works on the field of augmented reality applied on medicine showed that most of the works are still based on fiducial markers. Works related to markerless augmented reality approaches will be studied on the next section.

3.2 MARKERLESS AUGMENTED REALITY

On the last years, alternatives to the traditional fiducial markers started to emerge. One of the common problems related to fiducial markers is that the marker needs to be located on the field of view of the camera. A possible solution for this is presented on (Lee et al., 2011), that is able to track a camera based on plane tracking algorithm. This technique has been applied mainly due to its simple geometry. Results also show that 3D objects can be used on augmented reality applications with primitive-based modeling (Kim; Lepetit; Woo, 2010).

One of the first works to propose an algorithm for markerless object tracking was (Comport; Marchand; Chaumette, 2003), that suggests a tracking algorithm based on 3D models to calculate the distance between the camera and the objects. Based on this calculation, objects can be placed on the scene. Though this method seemed to be robust on handling occlusion and luminosity issues, which are weak points of the fiducial markers, it still had some limitations. This is an example of an application based on models, that already uses geometric 3D data to identify where to place a virtual image on the real scene.

Other alternatives to fiducial markers are proposed on (Wuest; Vial; Stricker, 2005), that uses an adaptive edge detector, (Ferrari; Tuytelaars; Gool, 2001), that renders planar textures over planar fragments previously identified, and on (Klein; Murray, 2007), which proposed a multi-thread system to track a camera and to compute a three-dimensional map of the available marks, using a hand-held camera.

The system designed on (Zhang; Lew, 2012) allows any object from the environment to be used as a marker, instead of using a fiducial marker. Besides that, the system was designed to work with low-contrast surfaces (like human hand marks). In order to place a three-dimensional object, the system uses salient points from the environment combined with a local texture, which turns the detection much more stable.

Like that work, other works show that feature detection on the target scene can achieve good performance. Descriptors are usually computed from key-points of the scene, so, they don't reach great performance when dealing with objects without texture and/or with few key-points on their surfaces (Bay et al., 2008).

On the other hand, object recognition based on depth focuses on geometric properties of the target, like curves or edges. An object can be identified based on its depth map.

3.3 SENSORS

Recently augmented reality works use sensors in order to capture scene data and use them for pose estimation, replacing the need for markers. Such sensors can capture real images from the scene in order to identify natural marks or features, or can even capture three-dimensional objects to be used as geometric markers.

The study on (Lee et al., 2012) proposes a cranial augmented reality system which performs image-to-patient registration using only natural facial features. The hardware for this project is composed by three calibrated cameras and no fiducial markers. Two of the cameras are mounted together in order to form a stereo-vision system, while the third camera moves freely and captures real-time images for displaying with the virtual image. The patient's head is first reconstructed by stereo vision, while another surface is reconstructed from computed tomography images. An algorithm based on the iterative closest point (ICP) is then used to register the two facial data, which transfers the facial information from the computer tomography images to the physical space. The reconstruction, as expected, comes with noisy data, so the conventional ICP would suffer from the local-minimum problem of not reaching the best match. Therefore, the authors also propose an improvement to the ICP algorithm in order to increase robustness.

An interesting approach that uses the human hand as a marker is proposed by (Lee; Hollerer, 2007). On the calibration stage, an algorithm based on computer vision detects the user's hand (by checking the contours of the fingers) and use them as a reference pattern, thus providing a virtual camera with six degrees of freedom on the user's hand, that's where virtual objects will be projected. On the next stage, the user can move his hand arbitrarily and so the virtual object projected on his hand will mode accordingly too. This is an example of augmented reality application that uses image analysis to identify where to project the virtual object.

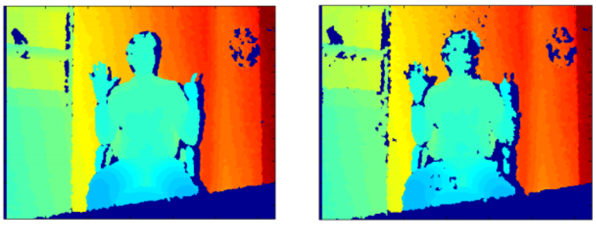

Figure 3.4 Difference between using a single Kinect and more than one Kinects

Another system that treats the user hand as a marker is presented on (Santos; Lamounier; Cardoso, 2011). On this implementation, the user can interact with menus and three-dimensional objects without using any fiducial marker. The system is activated from a specific motion performed by the user's hand, which is captured by a Kinect device. The interaction means selecting and controlling a set of virtual objects and buttons that are displayed in front of the user, which wears a pair of augmented reality glasses.

3.4 MULTIPLE KINECTS

More than one Kinect is needed when the use case is capturing a large volume, full 3D reconstruction, tracking or super resolution. However, when more than one Kinect device is used, new challenges come in. Data captured from the overlapping region between two or more Kinects usually have poor quality due to the interference, since the depth sensing technology limits the use of multiple Kinects (Or-el, 2013). This noticed from the depth maps, which suffer major quality degradation in overlapping areas. In places of depth uncertainty, Kinect will place zero values, and so, holes will be created. The difference between using a single Kinect and multiple Kinects is represented on Figure 3.4.

On the other hand, multiple Kinects can result on better data quality for large objects or regions since each device can be responsible for capturing an specific part of an object or scene, resulting in more details for each component. Similar approach is implemented by the work at (Tong et al., 2012), which aims to make a full three-dimensional scan of the human body by using three Kinects carefully positioned in order to avoid overlapping (and consequently, interference) between them as much as possible, as shown on Figure 3.4.

The main reason that made the authors decide for using multiple Kinects was because in order to capture a full human body, a single Kinect should be positioned around 3 meters away from the body, and the resolution would be very low, since little geometry information is captured on the depth map. Even using multiple frames to enhance the final resolution, the result wouldn't be acceptable.

Not only multiple Kinects, but multi-camera environments are used for three-dimensional reconstruction, since a single camera can just capture a two-dimensional projection of the scene. Not many works were found with regards to calibrating multiple Kinects. The most notable way of calibrating many RGB cameras in relation to a common reference is by processing images captured from a chessboard. The approach at (Bouguet, 2008) became popular by providing a straightforward way to find chessboard corners and that's

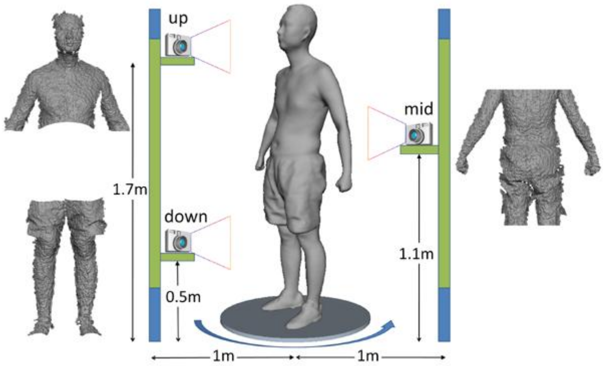

Figure 3.5 Three Kinect devices being used for a full scan of the human body with minimum overlapping (Tong et al., 2012)

still largely used.

When multiple Kinects are present on a multi-camera configuration, the challenge is to develop methods that simultaneously calibrate the RGB sensor and depth sensors using a suitable calibration pattern. An approach to that is discussed on (Berger et al., 2011), which uses four Kinect devices for motion capture. This work investigates on reducing or mitigating the detrimental effects of multiple active light emitters, thereby allowing motion capture from all angles. The calibration approach used by the authors is to still use a chessboard pattern to calibrate the RGB and depth sensors simultaneously, but not a simple board printed on paper, instead they use a binary pattern consisting of diffuse and mirroring patches. The reflective patches act as mirrors and deflect the IR pattern to infinity while the diffuse patterns reflect the IR light and provide depth values in the captured image, which are finally used for calibration in the Matlab calibration toolbox (Bouguet, 2008). Many other studies for multiple Kinects are discussed on (Schröder et al., 2011).

Besides calibrating the depth and RGB maps simultaneously, for some cases it's also important to calibrate one Kinect in relation to the other and so be able to determine the relative position between them. On (Jedvert, 2013), among other works, it's used a calibration process introduced by (C.; Kannala; Heikkil, 2012), which not only explains the approach, but also releases a toolbox to perform the calibration, which is represented on Figure 3.4.

That algorithm simultaneously calibrates the color camera and the depth camera, as well as the relative position between them. The color camera intrinsics is based on the following model: a 3D point is transformed into image coordinates and a disparity value from the disparity map at these coordinates is transformed into a 3D point.

3.5 HYBRID-BASED TRACKING

Since sensor-based methods rely solely on geometric information, sometimes it can fail for planar or smooth surfaces. On such cases, the results can be improved by combining geometric data with visual data. Different methods can be employed to detect specific

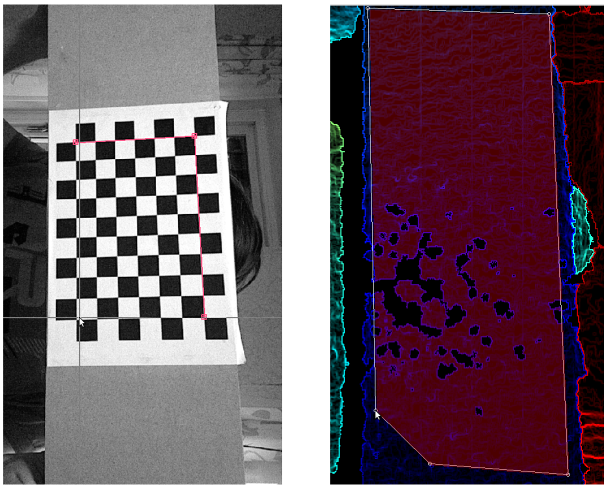

Figure 3.6 Calibrating Kinect using toolbox from (C.; Kannala; Heikkil, 2012)

features or objects in the scene, such as point detection, background modeling, image segmentation or classifiers based on supervised learning.

One approach to use visual information to improve geometric mapping is discussed on (Henry et al., 2010), which combines visual features with shape-based alignment. This paper introduces RGB-D mapping, a framework that can generate dense 3D maps of indoor environments despite the limited depth precision and field of view provided by RGB-D cameras. The core of this framework is a variant of the ICP algorithm, called RGBD-ICP, which uses the rich information contained in RGB-D data. First, the framework performs an initial estimate of the 3D camera transformation by applying RANSAC[1] to SIFT[2] feature matches with depth values. The modified ICP algorithm combines these visual feature associations with dense point associations, and then the final transformations are used to create a pose graph, which at a final post-processing step is optimized in order to achieve a consistent map. Later on, the same authors released another paper (Henry et al., 2012) where they avoid the expensive ICP step when enough features are identified on the image. Another work, at (Engelhard et al., 2011), follows a similar approach but uses SURF (Bay et al., 2008) instead of SIFT features.

The closest to what is done on this work is the work by (Peasley; Birchfield, 2013), which improves the KinectFusion algorithm by using the Lucas-Kanade algorithm in cases of low geometric features.

An application of the hybrid-based tracking is also noticed on (Giovanni et al., 2012). It's a virtual try-on system that employs one Kinect sensor and one high definition camera. It covers the calibration of the HD camera in relation to the Kinect sensor, which is based on the OpenCV implementation of a standard chessboard calibration (Bouguet, 2008). On this work, the Kinect is used for skeleton tracking and height estimation, and the HD camera is used for capturing a better video from the user.

[1]Random sample consensus - an iterative method to estimate the parameters of a mathematical model

[2]Scale invariant feature transform - an algorithm in computer vision to detect and describe local features in images

Chapter 4

In this chapter I explain in details the steps performed in order to implement the objective of this work.

SOLUTION ARCHITECTURE

In order to achieve the objectives of this work, it was necessary to implement some softwares and auxiliary codes, all of them based on other open source code or open source libraries. All the code implement on this work was released under open licenses and are available on public repositories

4.1 ENVIRONMENT

The multi-view environment proposed by this work is composed by two Kinects and one pair of augmented reality glasses. Since each Kinect needs its own USB hub (not only an USB port), and the glasses themselves need other two USB ports, it was not possible to use a single computer to

4.2 CALIBRATION

4.2.1 Augmented reality glasses calibration

4.2.2 Initial calibration between Kinects

4.2.3 Initial calibration between Kinect and glasses

4.3 COMMUNICATION

Network, sockets, etc.

4.4 TRANSFORMATIONS

4.5 METHOD 1: GLASSES ACCELEROMETER AND ONE KINECT

Cover Lucas-Kanade algorithm, etc.

4.6 METHOD 2: TWO KINECTS

Talk about performance of two Kinfus fighting for a single GPU.

4.7 HYBRID APPROACH

When optical flow has just a few feature points, we switch to the second Kinect.

Chapter

5

In this chapter I present the results of the procedure explained in the previous chapter.

RESULT

5.1 SCOPE

Talk about error propagation.

5.2 ANALYSIS

Talk about performance and alignment results.

5.3 COMPARISON

Compare methods 1 and 2 with regards to performance and quality.

Chapter 6

In this chapter I discuss the conclusions of this work and list some possibilities of future works.

CONCLUSIONS

6.1 FUTURE WORK

BIBLIOGRAPHY

Arca, S.; Casiraghi, E.; Lombardi, G. *CORNER LOCALIZATION IN CHESSBOARDS FOR CAMERA CALIBRATION*. 2005.

Armangué, X.; Salvi, J.; Batlle, J. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognition*, v. 35, p. 1617–1635, 2000.

Artoolkit. *ARToolkit*. 1999. Available at: <http://www.hitl.washington.edu/artoolkit>.

Azuma, R. et al. Recent advances in augmented reality. *IEEE Comput. Graph. Appl.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 21, n. 6, p. 34–47, nov. 2001. ISSN 0272-1716. Available at: <http://dx.doi.org/10.1109/38.963459>.

Azuma, R. T. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, v. 6, n. 4, p. 355–385, ago. 1997.

Bajramovic, F. *Self-Calibration of Multi-Camera Systems*. [S.l.]: Logos Verlag Berlin, 2010. ISBN 9783832527365.

Baker, S.; Matthews, I. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, Kluwer Academic Publishers, v. 56, n. 3, p. 221–255, 2004. ISSN 0920-5691.

Baker, S. et al. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, Springer US, v. 92, n. 1, p. 1–31, 2011. ISSN 0920-5691. Available at: <http://dx.doi.org/10.1007/s11263-010-0390-2>.

Barron, J.; Fleet, D.; Beauchemin, S. Performance of optical flow techniques. *International Journal of Computer Vision*, Kluwer Academic Publishers, v. 12, n. 1, p. 43–77, 1994. ISSN 0920-5691.

Bay, H. et al. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, Elsevier Science Inc., New York, NY, USA, v. 110, n. 3, p. 346–359, jun. 2008. ISSN 1077-3142. Available at: <http://dx.doi.org/10.1016/j.cviu.2007.09.014>.

Berger, K. et al. Markerless motion capture using multiple color-depth sensors. In: *VMV*. [S.l.: s.n.], 2011. p. 317–324.

Besl, P.; Mckay, N. D. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, v. 14, n. 2, p. 239–256, Feb 1992. ISSN 0162-8828.

Bouguet, J. Y. *Camera calibration toolbox for Matlab*. 2008. Available at: <http://www.vision.caltech.edu/bouguetj/calib_doc/.>.

Bouguet, J. yves. Pyramidal implementation of the lucas kanade feature tracker. *Intel*, 2000.

Bradski, G. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000.

Bradski, G.; Kaehler, A. *Learning OpenCV: Computer Vision with the OpenCV Library*. [S.l.]: O'Reilly Media, 2008. ISBN 9780596554040.

Bruhn, A.; Weickert, J.; Schnörr, C. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, Kluwer Academic Publishers, v. 61, n. 3, p. 211–231, 2005. ISSN 0920-5691.

Bucioli, A.; Jr., E. L.; Cardoso, A. A utilização da realidade aumentada no tratamento e simulação de sinais cardiológicos com biofeedback em tempo real. In: . [S.l.]: V Workshop de Realidade Virtual e Aumentada, 2008.

Burrus, N. *Kinect Calibration*. 2014. Available at: <http://nicolas.burrus.name/index.php/Research/KinectCalibration>.

Burton, A.; Radford, J. *Thinking in Perspective: Critical Essays in the Study of Thought Processes*. [S.l.]: Methuen, 1978. (Psychology in progress). ISBN 9780416858402.

C., D. H.; Kannala, J.; Heikkil, J. Joint depth and color camera calibration with distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE Computer Society, Los Alamitos, CA, USA, v. 34, n. 10, p. 2058–2064, 2012. ISSN 0162-8828.

Carmigniani, J. et al. Augmented reality technologies, systems and applications. *Multimedia Tools Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 51, n. 1, p. 341–377, jan. 2011. ISSN 1380-7501. Available at: <http://dx.doi.org/10.1007/s11042-010-0660-6>.

Chen, X.; Davis, J. Wide area camera calibration using virtual calibration objects. In: *In Proceedings of CVPR*. [S.l.: s.n.], 2000. p. 520–527.

Clarke, T.; Fryer, J. The development of camera calibration methods and models. *The Photogrammetric Record*, Blackwell Publishers Ltd., v. 16, n. 91, p. 51–66, 1998. Available at: <http://dx.doi.org/10.1111/0031-868x.00113>.

Comport, A.; Marchand, E.; Chaumette, F. A real-time tracker for markerless augmented reality. In: *ACM/IEEE Int. Symp. on Mixed and Augmented Reality, ISMAR'03*. Tokyo, Japan: [s.n.], 2003. p. 36–45.

Damasceno, E.; Lamounier, E.; Cardoso, A. Uma avaliação heurística sobre um sistema de captura de movimentos em realidade aumentada. In: . [S.l.]: Journal of Health Informatics, 2012. v. 4, n. 3, p. 87–94.

Dolz, J. *Markerless Augmented Reality*. 2012. Available at: <http://www.arlab.com/blog/markerless-augmented-reality>.

Dong, J. et al. Lietricp: An improvement of trimmed iterative closest point algorithm. *Neurocomputing*, v. 140, n. 0, p. 67 – 76, 2014. ISSN 0925-2312.

Douskos, V.; Kalisperakis, I.; Karras, G. *AUTOMATIC CALIBRATION OF DIGITAL CAMERAS USING PLANAR CHESS-BOARD PATTERNS*. 2007.

Engelhard, N. et al. Real-time 3-d visual slam with a hand-held camera. In: *in Proc. RGB-D Workshop 3-D Perception*. [S.l.: s.n.], 2011.

Ferrari, V.; Tuytelaars, T.; Gool, L. J. V. Markerless augmented reality with a real-time affine region tracker. In: *ISAR*. [S.l.]: IEEE Computer Society, 2001. p. 87–96. ISBN 0-7695-1375-1.

Furht, B. *Handbook of Augmented Reality*. [S.l.]: Springer, 2011. (SpringerLink : Bücher). ISBN 9781461400646.

Gallo, L.; Ciampi, M. Wii remote-enhanced hand-computer interaction for 3d medical image analysis. In: *Proceedings of International conference on the Current Trends in Information Technology*. Los Alamitos, CA, USA: IEEE Computer Society, 2009. (CTIT '09), p. 85–90. ISBN 978-1-4244-5755-7.

Gibson, J. *The perception of the visual world*. [S.l.]: Houghton Mifflin, 1950.

Giovanni, S. et al. Virtual try-on using kinect and hd camera. In: Kallmann, M.; Bekris, K. E. (Ed.). *MIG*. [S.l.]: Springer, 2012. (Lecture Notes in Computer Science, v. 7660), p. 55–65. ISBN 978-3-642-34709-2.

Hajnal, J.; Hill, D. *Medical Image Registration*. Taylor & Francis, 2014. (Biomedical Engineering). ISBN 9781420042474. Available at: <http://books.google.com.br/books?id=2dtQNsk-qBQC>.

Hanning, T. *High Precision Camera Calibration*. [S.l.]: Vieweg+Teubner Verlag / Springer Fachmedien Wiesbaden GmbH, Wiesbaden, 2011. (Vieweg + Teubner research). ISBN 9783834898302.

Harris, C.; Stephens, M. A combined corner and edge detector. In: *In Proc. of Fourth Alvey Vision Conference*. [S.l.: s.n.], 1988. p. 147–151.

Henry, P. et al. Rgb-d mapping: Using depth cameras for dense 3D modeling of indoor environments. In: *Proc. of the International Symposium on Experimental Robotics (ISER)*. [S.l.: s.n.], 2010.

Henry, P. et al. Rgb-d mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments. *International Journal of Robotics Research (IJRR)*, v. 31, n. 5, p. 647–663, April 2012.

Hondori, H. M. et al. A spatial augmented reality rehab system for post-stroke hand rehabilitation. In: *MMVR*. [S.l.: s.n.], 2013. p. 279–285.

Izadi, S. et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: ACM. *Proceedings of the 24th annual ACM symposium on User interface software and technology*. [S.l.], 2011. p. 559–568.

Jean, J. S. *Kinect Hacks: Tips & Tools for Motion and Pattern Detection*. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2012. ISBN 1449315208, 9781449315207.

Jedvert, M. *3D Head Scanner*. Thesis (Mestrado) — Chalmers University of Technology, Goteborg, Sweden, 2013.

Kawashima, T. et al. Magic paddle: A tangible augmented reality interface for object manipulation. In: *Proceedings of ISMR 2001*. [S.l.: s.n.], 2001. p. 194–195.

Kim, K.; Lepetit, V.; Woo, W. Keyframe-based modeling and tracking of multiple 3d objects. In: *ISMAR*. [S.l.]: IEEE, 2010. p. 193–198.

Klein, G.; Murray, D. Parallel tracking and mapping for small AR workspaces. In: *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*. Nara, Japan: [s.n.], 2007.

Kramer, J. et al. *Hacking the Kinect*. 1st. ed. Berkely, CA, USA: Apress, 2012. ISBN 1430238674, 9781430238676.

Laureano, G. T.; Paiva, M. S. V. de; Silva, A. S. da. *Topological Detection of Chessboard Pattern for Camera Calibration*. 2013.

Lee, A. et al. Markerless augmented reality system based on planar object tracking. In: *Frontiers of Computer Vision (FCV), 2011 17th Korea-Japan Joint Workshop on*. [S.l.: s.n.], 2011. p. 1–4.

Lee, J.-D. et al. Medical augment reality using a markerless registration framework. *Expert Systems with Applications*, v. 39, n. 5, p. 5286 – 5294, 2012. ISSN 0957-4174.

Lee, T.; Hollerer, T. Handy ar: Markerless inspection of augmented reality objects using fingertip tracking. In: *Wearable Computers, 2007 11th IEEE International Symposium on*. [S.l.: s.n.], 2007. p. 83–90.

Library, P. C. *Estimating surface normals on a point cloud*. 2013. Available at: <http://pointclouds.org/documentation/tutorials/normal_estimation.php>.

Library, P. C. *Point Cloud Library*. 2013. Available at: <http://pointclouds.org/>.

Library, P. C. *Removing outliers using a StatisticalOutlierRemoval filter*. 2013. Available at: <http://pointclouds.org/documentation/tutorials/statistical_outlier.php>.

Liu, J. et al. Probability iterative closest point algorithm for position estimation. In: *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. [S.l.: s.n.], 2014. p. 458–463.

Lucas, B. D.; Kanade, T. An iterative image registration technique with an application to stereo vision. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981. (IJCAI 81), p. 674–679. Available at: <http://dl.acm.org/citation.cfm?id=1623264.1623280>.

Malik, S. *Robust Registration of Virtual Objects for Real-Time Augmented Reality*. Thesis (Mestrado) — Carleton University, Ontario, Canada, 2002.

Medioni, G.; Kang, S. B. *Emerging Topics in Computer Vision*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004. ISBN 0131013661.

Microsoft. *Microsoft Kinect for Windows*. 2010. Available at: <http://www.microsoft.com/en-us/kinectforwindows>.

Milgram, P. et al. Augmented reality: A class of displays on the reality-virtuality continuum. In: . [S.l.: s.n.], 1994. p. 282–292.

Newcombe, R. A. et al. Kinectfusion: Real-time dense surface mapping and tracking. In: *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*. Washington, DC, USA: IEEE Computer Society, 2011. (ISMAR '11), p. 127–136. ISBN 978-1-4577-2183-0. Available at: <http://dx.doi.org/10.1109/ISMAR.2011.6092378>.

Or-el, R. Working with multiple kinects. In: . [S.l.]: Seminar in Advanced Topics in Computer Vision, 2013.

Pati, U. C. *3-D Surface Geometry and Reconstruction: Developing Concepts and Applications: Developing Concepts and Applications*. [S.l.]: Information Science Reference, 2012. (Premier reference source). ISBN 9781466601147.

Peasley, B.; Birchfield, S. Replacing projective data association with lucas-kanade for kinectfusion. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. [S.l.: s.n.], 2013. p. 638–645. ISSN 1050-4729.

Pirovano, M. *Kinfu - an open source implementation of Kinect Fusion + case study: implementing a 3D scanner with PCL*. 2012.

Placitelli, A. P.; Gallo, L. Low-cost augmented reality systems via 3d point cloud sensors. In: Yétongnon, K.; Chbeir, R.; Dipanda, A. (Ed.). *SITIS*. [S.l.]: IEEE Computer Society, 2011. p. 188–192. ISBN 978-0-7695-4635-3.

Price, A. *Kinect-based object reconstruction*. Thesis (Mestrado) — University of Alabama, Tuscaloosa, Alabama, 2012.

Quero, S. et al. Preliminary data of an augmented reality training program for safety at work in hospitals. In: *INTED2014 Proceedings*. [S.l.]: IATED, 2014. (8th International Technology, Education and Development Conference), p. 1568–1573. ISBN 978-84-616-8412-0. ISSN 2340-1079.

Rojas, R. *Lucas-Kanade in a nutsheGll*. 2014.

Santos, E. S.; Lamounier, E. A.; Cardoso, A. Interaction in augmented reality environments using kinect. In: *Symposium on Virtual Reality*. [S.l.: s.n.], 2011.

Scharstein, D.; Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, Kluwer Academic Publishers, Hingham, MA, USA, v. 47, n. 1-3, p. 7–42, abr. 2002. ISSN 0920-5691. Available at: <http://dx.doi.org/10.1023/A:1014573219977>.

Schoo, M.; Mukundan, R. Animating highly constrained deformable head/face models using motion capture. 2006.

Schröder, Y. et al. Multiple kinect studies. *Computer Graphics*, v. 2, n. 4, p. 6, 2011.

Shi, J.; Tomasi, C. Good features to track. In: . [S.l.: s.n.], 1994. p. 593–600.

Shumaker, R. *Virtual and Mixed Reality - New Trends, Part I: International Conference, Virtual and Mixed Reality 2011, Held as Part of HCI International 2011, Orlando, FL, USA, July 9-14, 2011, Proceedings*. [S.l.]: Springer, 2011. (Information Systems and Applications, incl. Internet/Web, and HCI). ISBN 9783642220203.

Sun, D.; Roth, S.; Black, M. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, Springer US, v. 106, n. 2, p. 115–137, 2014. ISSN 0920-5691. Available at: <http://dx.doi.org/10.1007/s11263-013-0644-x>.

Tillapaugh, B.; Engineering, R. I. of T. C. *Indirect Camera Calibration in a Medical Environment*. [S.l.]: Rochester Institute of Technology, 2008. ISBN 9780549934479.

Tong, J. et al. Scanning 3d full human bodies using kinects. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Virtual Reality)*, v. 18, n. 4, p. 643–650, 2012.

Tori, R.; Kirner, C.; Siscoutto, R. *Fundamentos e tecnologia de realidade virtual e aumentada*. [S.l.]: Editora SBC, 2006. ISBN 9788576690689.

Tsai, R. Y. An efficient and accurate camera calibration technique for 3D machine vision. In: *Proc. Conf. Computer Vision and Pattern Recognition*. Miami: [s.n.], 1986. p. 364–374.

Vallino, J. R. *Interactive Augmented Reality*. Rochester, NY, USA, 1998.

Vera, A. et al. Augmented reality telementoring (art) platform: a randomized controlled trial to assess the efficacy of a new surgical education technology. *Surgical Endoscopy*, Springer US, v. 28, n. 12, p. 3467–3472, 2014. ISSN 0930-2794. Available at: <http://dx.doi.org/10.1007/s00464-014-3625-4>.

Weisstein, E. W. *Normal vector, from MathWorld – A Wolfram Web Resource*. 2014. Available at: <http://mathworld.wolfram.com/NormalVector.html>.

Wuest, H.; Vial, F.; Stricker, D. Adaptive line tracking with multiple hypotheses for augmented reality. In: *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*. Washington, DC, USA: IEEE Computer Society, 2005. (ISMAR '05), p. 62–69. ISBN 0-7695-2459-1. Available at: <http://dx.doi.org/10.1109/ISMAR.2005.8>.

Zhang, Q.; Lew, M. S. The leiden augmented reality system (lars). In: Fusiello, A.; Murino, V.; Cucchiara, R. (Ed.). *ECCV Workshops (3)*. [S.l.]: Springer, 2012. (Lecture Notes in Computer Science, v. 7585), p. 639–642. ISBN 978-3-642-33884-7.

Zhang, Z. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vision*, Kluwer Academic Publishers, Hingham, MA, USA, v. 13, n. 2, p. 119–152, out. 1994. ISSN 0920-5691. Available at: <http://dx.doi.org/10.1007/BF01427149>.

Zhang, Z. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 22, n. 11, p. 1330–1334, 2000.