Tampereen yliopisto

Aleksi Suonsivu

RGBD SLAM BASED 3D OBJECT RECONSTRUCTION AND TRACKING

Using Google ARCore

Bachelor's thesis
Faculty of Information Technology and Communication Sciences
July 2020

TIIVISTELMÄ

Aleksi Suonsivu: RGBD SLAM based 3D object reconstruction and tracking
Kandidaatintyö
Tampereen yliopisto
Signaalinkäsittely
Kesäkuu 2020

Lisätyn todellisuuden sekä virtuaalitodellisuuden suosion merkittävä kasvu tekniikan ja viihteen aloilla on luonut uusia haasteita sekä mahdollisuuksia. Googlen ARCoren ja Applen ARKitin kaltaiset kehitysalustat ovat mahdollistaneet lisätyn sekä virtuaalisen todellisuuden sovellusten kehityksen suuren kasvun. Erityisesti lisätyn todellisuuden sovelluksissa merkittäväksi osa-alueeksi on muodostunut kolmiulotteinen rekonstruktio ja ympäristön havainnointi. Tämän työn tavoite on tutkia ARCoren ominaisuuksia sekä analysoida onko sen avulla mahdollista toteuttaa kolmiulotteinen rekonstruktio kappaleesta tai tilasta käyttäen tavallista älypuhelimen kameraa.

Työ jakautuu viiteen kappaleeseen ja työn alkupäässä käydään läpi lyhyesti syitä tälle tutkimukselle ja aiheeseen liittyviä aikaisempia töitä sekä menetelmiä. Toinen kappale käsittelee ARCoren ominaisuuksia ja toimintaperiaatteita. Kolmannessa kappaleessa käsitellään konenäön ja erityisesti lisätyn todellisuuden aloille merkittäviä menetelmiä, joihin ARCoren toiminta perustuu. Kappaleessa kerrotaan, miten kolmiulotteinen rakenne saadaan selville skenaariosta liikkeen, syvyyssensorin tai usean kameran avulla. Menetelmistä kerrotaan lyhyesti niiden toimintaperiaatteet sekä käyttötarkoitukset.

Työn loppupäässä esitellään kaksi erillistä sovellusta, jotka hyödyntävät ARCorea kolmiulotteista rekonstruktiota tehdessä ja näiden sovellusten tuottamat tulokset. Testausvaiheessa näitä kahta sovellusta testataan älypuhelimella ja lopussa vertaillaan näitä saatuja tuloksia keskenään, sekä pohditaan mahdollisia ARCoren tulevia käyttötarkoituksia sekä ominaisuuksia. Kolmiulotteinen rekonstruktio on siis mahdollista toteuttaa ARCoren keräämän datan avulla, mutta ARCore itsessään ei suoraan tue rekonstruktiota, vaan sen lisäksi tarvitaan erillisiä sovelluksia itse rekonstruktion suorittamiseen.

Avainsanat: lisätty todellisuus, 3D, rekonstruktio, ARCore, virtuaalitodellisuus

ABSTRACT

The significant growth in the popularity of augmented reality and virtual reality in the fields of technology and entertainment has created new challenges and opportunities. Development platforms like Google's ARCore and Apple's ARKit have enabled increased growth in augmented as well as virtual reality application development. Especially in augmented reality applications, three-dimensional reconstruction and environmental observation have become an important area. The aim of this work is to study the features of ARCore and to analyze whether it is possible to carry out a three-dimensional reconstruction of an object or scene using a standard smartphone camera.

The work is divided into five sections, and at the beginning of the work, the reasons for this research and previous work and methods related to this topic are briefly reviewed. The second section discusses the features and operating principles of ARCore. The third section discusses the important methods for machine vision and especially augmented reality, on which ARCore operates. The section explains how to find out the three-dimensional structure of a scenario using motion, a depth sensor, or multiple cameras. The methods are briefly described in terms of their operating principles and uses.

At the end of the work, two separate applications that utilize ARCore in three-dimensional reconstruction and the results produced by these applications are presented. In the testing phase, the two applications will be tested on a smartphone, and at the end, these results will be compared with each other, as well as possible future uses and features of ARCore will be considered. Thus, it is possible to implement a three-dimensional reconstruction with the help of data collected by ARCore, but ARCore itself does not directly support the reconstruction. Separate applications are needed to perform the texture meshing and visualization.

Keywords: augmented reality, ARCore, image processing, android, 3D reconstruction

ALKUSANAT

This thesis was written as a part of Bachelor of Science program in spring 2020.

Special thanks to my supervisor Joni Kämäräinen for his patience and guidance. And also thanks to my close ones for their support.

Tampere 22.7.2020.

Aleksi Suonsivu

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

3D	three-dimensional
AR	augmented reality
VR	virtual reality
SLAM	simultaneous localization and mapping
POSE	position and rotation
COM	concurrent odometry and mapping
SDK	software development kit
API	application programming interface
RGBD	red-green-blue-depth
SFM	structure from motion
VO	visual odometry
VIO	visual inertial odometry
SIFT	scale-invariant feature transform
IMU	inertial measurement unit

1. INTRODUCTION

In recent years, Augmented Reality (AR) has grown in popularity due to the rising availability of high-end smartphones and overall improvement of computing power. AR applications are slowly becoming a part of our everyday life, and computer created worlds are no longer utopic concepts of the future. In this thesis we will study possible ways to create a 3D reconstruction on a standard smartphone.

Structure of this study is as follows. Chapter 2 focuses on Google's ARCore and its features. In chapter 3 we take look at fundamental theory behind problems like *motion tracking* and *reconstruction.* Chapter 4 presents two applications that use ARCore to solve this problem and the testing setup. In chapter 5 we go through the conclusions and possible future applications of ARCore.

1.1 Motivation

With augmented and virtual reality (VR) gaining more popularity every day, the field of computer science has really started to focus on these two things. It has been realized that AR applications can have many profitable uses for entertainment, engineering and healthcare. One reason for the rising amount of new AR applications might be publishing of Google's ARCore and Apple's ARKit. With these two frameworks, the development of AR applications has become easier and more efficient for the developers. In Figure 1 is shown so called *virtuality continuum* which describes the fusion of AR and VR to our daily environment [1].

First smartphone AR application implementations date back to 2004 [2]. Especially in the field of entertainment and video games, AR has improved its status quite a lot.

Briefly, mobile AR applications are capable of two things, tracking the rotation and location of the device in real-time and have a basic understanding of its surrounding scene. This makes collisions and interactions between virtual objects and the scene possible.

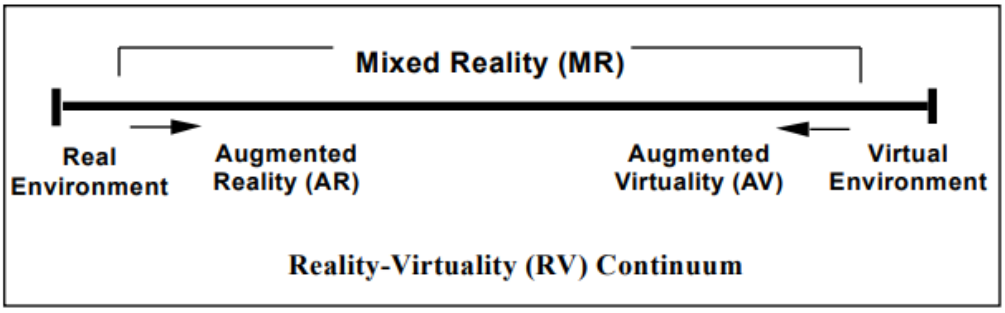

Figure 1. Virtuality Continuum. [1]

1.2 Related Work

The field of study of 3D reconstruction relies heavily on methods like Simultaneous Localization and mapping (SLAM) and image-based modeling. In 2007 Klein and Murray proposed a method of estimating camera pose (position and rotation) in unknown scene [3]. Their proposed method consists of two parallel tasks using dual-core computers. Problem is split for one thread to handle the tracking of the handheld device's motion and second thread to produce a 3D map of the feature points, observed in video frames. Later in 2013 Bylow *et al.* published their paper where they proposed a method based on Signed Distance Functions where they utilized depth maps gained from the scene using RGBD sensor. In Figure 2 is described the outcome of this particular method. [4]

Figure 2. 3D reconstruction using Signed Distance Functions. [4]

2. GOOGLE ARCORE

This chapter focuses on Google's platform ARCore and its features and how those could be used to perform a 3D reconstruction with a smartphone.

2.1 Overview of ARCore

ARCore is a platform published by Google in March 2018, which is used for building AR applications. ARCore offers APIs for different operating systems and development platforms. ARCore supports both Android and iOS operating systems. ARCore supports all Apple's devices that support Apple's own ARKit and have iOS version 11.0 or higher [5].

ARCore is a great platform for app and mobile-game development, since it has a large community with lots of tutorials and sample projects to get started with. ARCore is currently mostly used for interaction between real-world and virtual objects, which can be used in AR applications.

Currently ARCore only supports plane detections and feature point tracking, therefore it does not support 3D reconstruction directly. To understand how we can achieve our goal of performing the reconstruction on our smartphone, we need to familiarize ourselves with studies of 3D reconstructions field.

To solve this reconstruction problem, we can use ARCore's ability to detect and track feature points and planes. Using our smartphone's camera's visual data and data from our device's IMU unit, we can store and track *point clouds.* These point clouds can later be plotted into 3D representation. Later, *texture meshing* can be done with some external software, because ARCore itself does not support that.

2.2 Fundamental features of ARCore

ARCore's three fundamental abilities are *motion tracking, environmental understanding* and *light estimation.* The working principle of ARCore uses a method called *concurrent odometry and mapping* (COM) for locating the position of the device, relative to the world around it. COM is used to detect and track distinct feature points, which are then used to calculate the motion of the device. The visual information obtained with COM is then combined with inertial measurement data. This data is collected with device's *inertial measurement unit* (IMU). The combined visual and inertial data is then used to estimate the *position* and *orientation* of the device relative to the world,

and it is called *Visual-Inertial Odometry*. Figure 3 shows the pipeline of Visual-Inertial Odometry. [8]

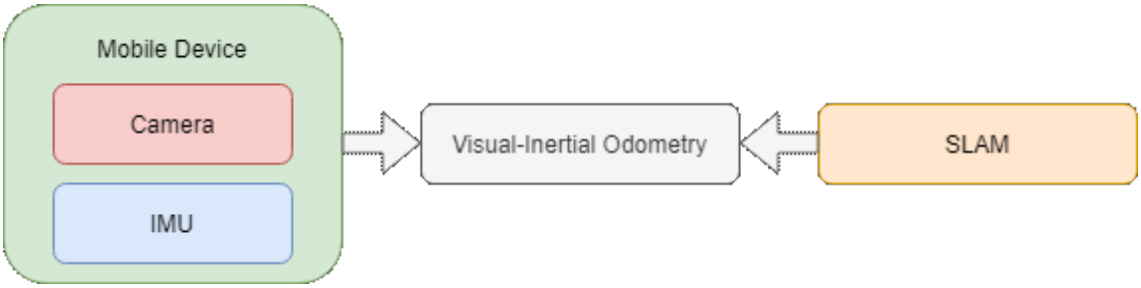

Figure 3. The process that computes IMU data and images for motion tracking is called Visual-Inertial Odometry. [6]

For environmental understanding ARCore looks for clusters of feature points that appear on flat horizontal and vertical surfaces. These surfaces are then stored as *planes* for the user to use in their app. ARCore uses the feature points of these surfaces for *plane detection,* so the surfaces need to have some sort of texture, so ARCore can detect points when moving. For example, completely white floor does not give ARCore enough information about the scene. [6]

ARCore uses *light estimation* for creating more realistic AR objects, by adjusting the virtual elements lighting to the perceived lighting of the surrounding world. By filtering the intensity of a subset of pixel data for each frame, we can calculate the light estimation. After computing the estimation, the returned values are used to adjust the rendering style of placed virtual objects or planes. [6] To create these virtual objects, ARCore provides a method to create these objects called *anchors.* Anchors position are tracked, and they are used as key-points for placing stationary objects [7]. Anchors can be placed on detected planes, for example a wall. Anchors track the movement of the device and its pose and use that data to calculate anchor's relative pose to the camera.

As we can see, ARCore is capable of handling many important tasks for creating an AR application and it provides great platform for app development. Figure 4 shows the fundamental pipeline of mobile devices processing. [6]

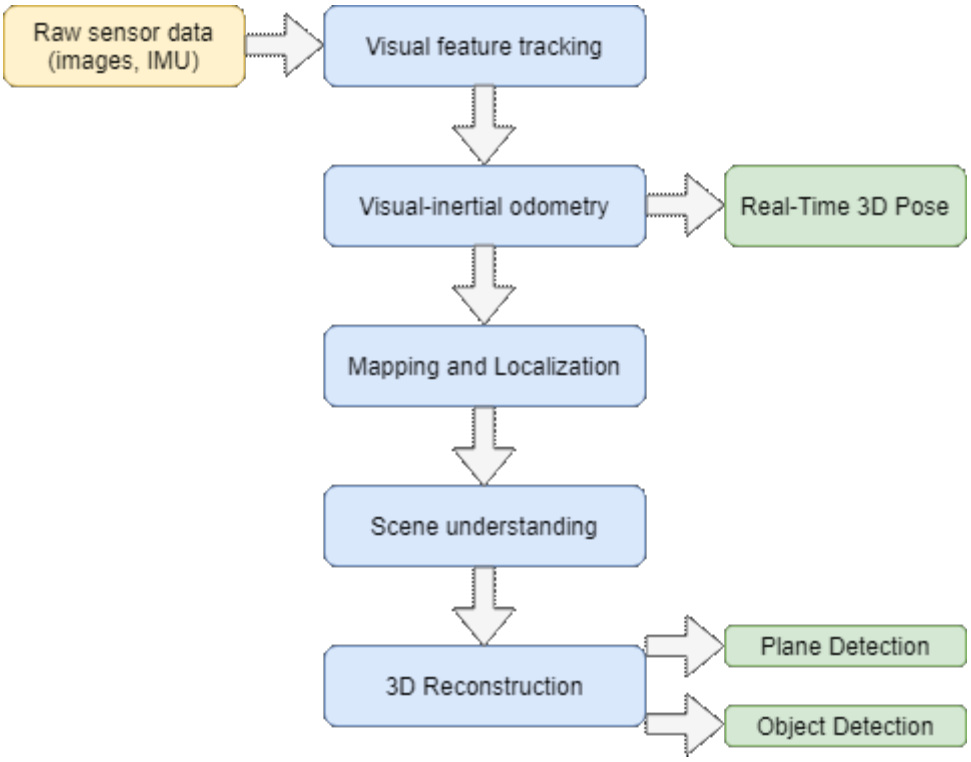

Figure 4. Mobile processing pipeline. [6]

The usage of ARCore is free and Google offers a large documentation for ARCore, and it is available online. [8] Google offers own documentations and *software development kits* (SDK) for Android, iOS, Unity and Unreal.

3. FUNDAMENTAL CONCEPTS

This chapter focuses on the fundamental concepts behind ARCore's features. We go through Structure from Motion and its theory and then we go through Visual Odometry and Visual-Inertial Odometry.

3.1 Structure from Motion

In the field of computer vision *Structure from Motion* (SfM) has been one of the most studied areas of research for the last 30 years. The quantity of studies has improved SfM methods, and SfM applications have reached commercial application level. SfM problems can be split into two scenarios: single-view and stereo-view geometries.

3.1.1 Single-view geometry

Single-view geometry needs to utilize mathematical applications to obtain projected 3D coordinates of the image into 2D camera frame. [9]

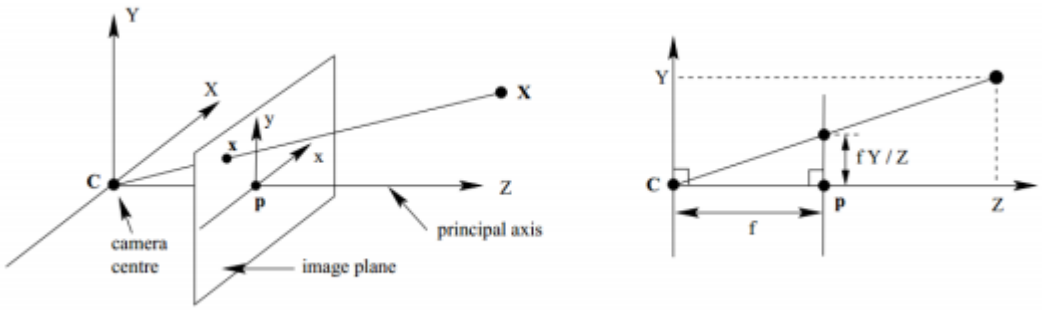

Figure 5. Geometry between camera centre and the image plane. [9]

The relationships of images and 3D structure of points can be calculated using three simple transformations of the projected image. First the ground coordinates need to be converted to camera coordinates as follow:

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1)$$

Where coordinates with index C represent camera coordinates, R being the rotation and T translation matrix to the origin of real-world coordinate system. [9]

After that the coordinates need to be transformed from 2D form to 3D form using following formula with *pinhole camera* assumption that the *focal length* $f = 1$.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} \tag{2}$$

Third transformation is to transform camera plane coordinates to images coordinate system where K is camera's calibration matrix as follows:

$$K = \begin{bmatrix} a_u & s & u_0 \\ 0 & a_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Where a_u and a_v are scale factors and s and scaling factor of sloping. According to Robertson and Cipolla [12], if $\boldsymbol{u}_i = [u_i \quad v_i]^T$ is the measured image position of 3D point $[X_i \quad Y_i \quad Z_i]^T$, the projection matrix P can be expressed as:

$$u_i = \frac{p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}} \tag{4}$$

$$v_i = \frac{p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}} \tag{5}$$

The parameters p_{11}, $p_{12} \ldots p_{34}$, in equations (4) and (5) are the unknowns in the projection matrix. These parameters need to be solved to obtain camera's pose $\boldsymbol{u}_i$ and structure $[X_i \quad Y_i \quad Z_i]^T$. [12]

$$\begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{pmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \mathbf{0}. \tag{6}$$

To obtain cameras pose and scenes structure from image data, we need at least six points from two overlapping pictures [9]. This formula represents the *pinhole camera* scenario with single-view geometry. The other form of SfM's geometry is multi-view, also known as stereo-view geometry.

3.1.2 Multi-view geometry

Multi-view geometry, also known as stereo-view geometry or *Epipolar Geometry* consists of scene with two or more cameras. Epipolar Geometry defines the projectory geometry between cameras and it is independent of the scene structure, as it depends on cameras pose and internal parameters. [10]

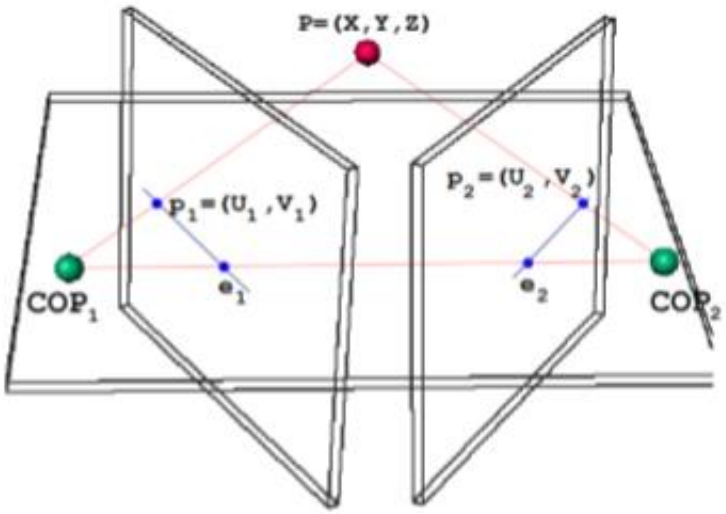

Figure 6. Epipolar geometry of a scene. [10]

Figure 6 displays the epipolar geometry of a scene, where P is a corresponding point, and COP_1 and COP_2 are the centers of projections in each camera. P, COP_1 and COP_2 form a plane called *epipolar plane.* An *epipolar line* connects both COP_1 and COP_2, and points e_1 and e_2 which are located at the intersection of epipolar line and the image plane, are called *epipoles*. Relationship between points p_1 and p_2 can be solved by formula, where F is the *Fundamental Matrix*, which represents the algebraic relation of epipolar geometry [9]:

$$p_1{}^t * F * p_2 = 0 \tag{7}$$

Hartley et. al. proposed that with eight corresponding points from multi-view scene, it is possible to solve matrix F, and then solve the intrinsic and extrinsic parameters of the scene and the reconstruction can be done. Reconstruction algorithm can be simplified into three steps [10]:

1. Compute the fundamental matrix F, with point correspondences found from the images.
2. Compute cameras intrinsic matrices, with the fundamental matrix.
3. For each correspondence, compute the 3D coordinates of the point projected to the two points.

3.1.3 Pipeline of Structure from Motion

According to Rossi et. al [11], SfM reconstruction's pipeline can be divided into 5 steps. The SfM's pipeline is summarized in the figure 5.

1. Feature extraction

The feature keypoints of the scene are extracted using feature detection algorithm, for example: SIFT (Scale-Invariant Feature Transform). SIFT is the most common feature detection algorithm as it can detect features even if the images scale, contrast, rotation or translation change. Other common methods are SURF, ORB and KAZE.

2. Feature matching

This step utilizes two algorithms, one to keep track of the extracted keypoints and one to match these features across the frames.

3. Bundle adjustment

Bundle adjustment methods are used to estimate the pose and structure, when creating the projective two-view geometry.

4. Generating the point-cloud

This step consists of producing the point-cloud from our extracted 3D points. Dense point-cloud can be achieved by obtaining sparse cloud and then refining cameras pose and repeating the process to get denser point-cloud.

5. Visualization

Last step is to visualize the point-cloud. This can be done in different ways, easiest being just plotting the 3D points according to their coordinates. More accurate visualizations can be achieved with using *meshing*, where an algorithm is used to create a surface between on our 3D point-cloud.

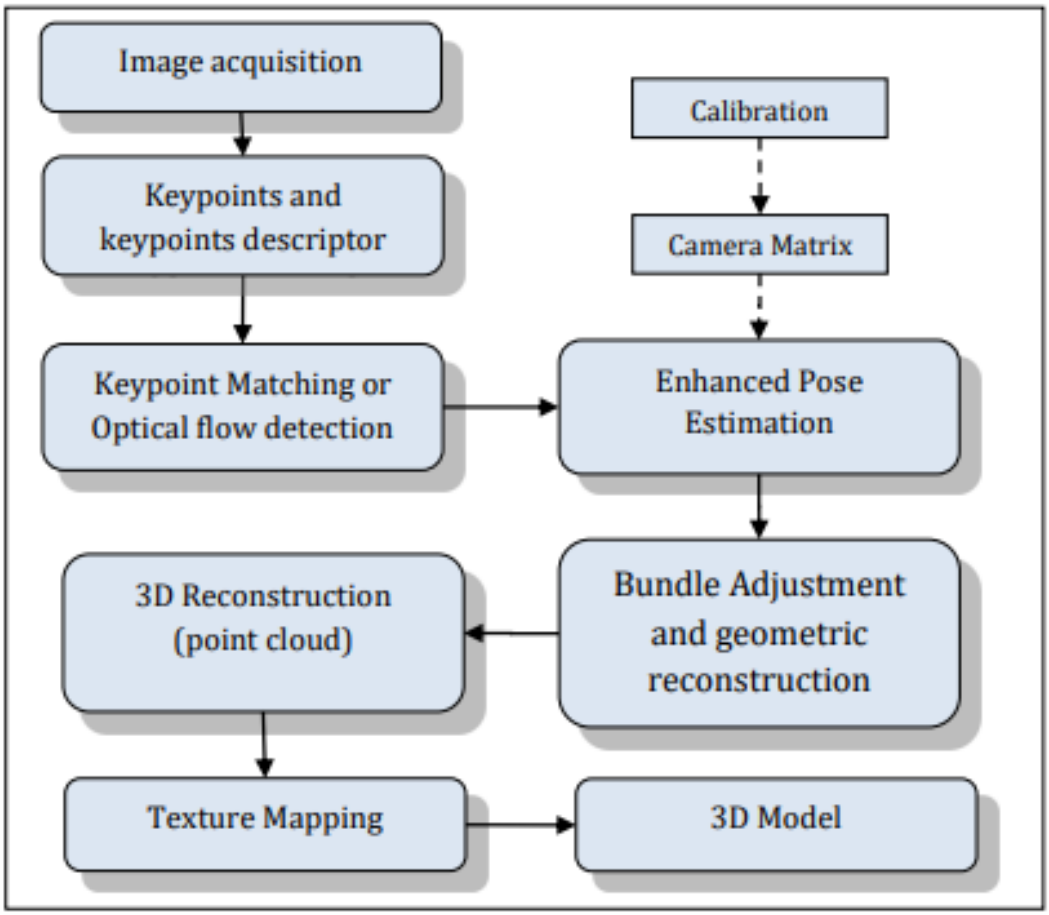

Figure 7. SfM pipeline. [9]

3.2 Visual Odometry

Visual Odometry (VO) means estimating the pose of the device using only data gained from images collected by monocular or stereo camera system. Existing VO methods can be separated into three classes: feature-based methods, direct methods and hybrid methods.

Direct methods work by estimating camera's motion by minimizing the photometric error over all pixels. Most VO methods are feature-based and work by detecting keypoints and matching them between frames. [12] Hybrid methods combine elements of these two prior methods, and its feature-correspondence is an implicit result of direct motion estimation [13].

Below is presented a simplified pseudo-algorithm for most existing approaches to visual odometry [14].

1. Acquire image data from camera sensor.
2. Image correction: applying image processing to reduce lens distortion etc.
3. Feature detection: define the points of interest and match them across frames and construct optical flow field.
4. Check flow field vector for error and remove outliers.
5. Estimate the camera motion from the optical flow.

3.2.1 Visual-Inertial Odometry

Method where the visual data gained from device's camera and the inertial data collected by device's IMU are combined, is called *Visual-Inertial Odometry* (VIO). IMU is the most commonly used measurement unit in mobile devices used to track the movement of the device. Most IMU's consist of accelerometers, gyroscopes.

Gyroscope measures device's rotational velocity and accelerometer measures the acceleration of the device relative to the state of free-fall. This means that device that is in free-fall has relative acceleration of $0\ m/s^2$ and when still, the acceleration is $9.81\ m/s^2$ upwards. Accelerometers measure the acceleration on only one axis, therefore for maximum information multiple units are needed [15]. Both accelerometer and gyroscope are sensitive to noise, bias and other factors that affect the accuracy of each unit.

For combining the visual data and the inertial data, following parameters need to be known, in order to precisely use the IMU's data [15]:

- Camera's calibration and distortion parameters.
- The position offset of IMU and camera.
- The delay between IMU and camera.
- The camera's capturing rate.
- Random bias factors.

VIO is the method that most of the modern AR applications including ARCore rely on, since the IMUs' prices have fallen, and the quality has improved at the same time. VIO enables us to obtain most accurate information about device's pose relative to the scene, since the visual and inertial data are combined.

IMPLEMENTATION AND TESTING

This chapter focuses on implementing and testing the above-mentioned methods and theory. For testing the 3D reconstruction on a smartphone, we are using Android Operating System device, Huawei P30 Pro smartphone and two different public applications. Goal in this chapter is to perform a 3D reconstruction and analyze its performance and possible usage.

3.3 Application 1

First application demonstrates how the 3D-pointcloud can be obtained from the scene in real-time using ARCore's functions. The application utilizes code from Silidragos [16] and it is ran in Android Studio. The application has separate stand-by and scanning modes, and when the scanning mode is turned on, our device seeks for feature points and stores them as temporary point cluster. It also tracks the pose of our device to create a large point cloud from the obtained clusters. Once the scanning is finished the point cloud is stored to JSON-file (Java Script object notation) in the form of

$$Keypoint = \{[X, Y, Z], \{[R, G, B]\}$$

where the *X, Y* and *Z* are the coordinates of the point's position and the *R, G* and *B* are the components of color in form of RGB (Red, Green, Blue).

The phone's pose is tracked with ARCore's class Pose, which is used to describe the transformation from object's local coordinate space to the world coordinate space. The Pose class offers built in methods to extract information about the pose, like rotation and translation.

3.4 Application 2

Second application that we use to test ARCore's features is a 3D Scanner published by L. Vonásek [17]. Scanner is used to scan an object or scene with a mobile device, to create an 3D model of it. Scanning happens in real-time and the app meshes a texture on top of already scanned areas. Texture rendering is done using OpenGL renderer. After the scanning is done, the data is processed, and the final 3D model

is produced. In figure 3. is shown the pipeline of Vonásek's implementation. [15]

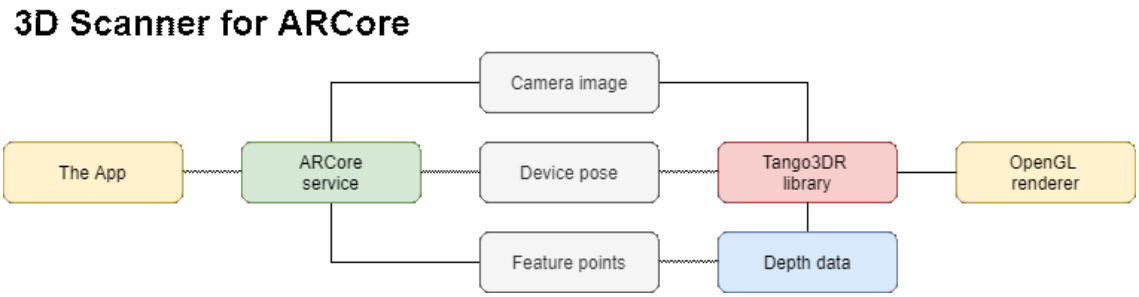

Figure 8. The pipeline of Vonásek's 3D Scanner for ARCore. [18]

User has 4 options for capturing resolution: 2 cm, 4 cm, 8 cm and Face scan. 2 cm is meant for interior scanning, 4 cm for exterior and 8 cm for fast scanning. User can also select if the depth sensor is used for the scanning. This 3D scanner uses either the depth sensor of the device or *passive depth sensing* to obtain the structure of the scene. Passive depth sensing calculates the depth component using *visual-inertial odometry* and it is used if the device does not have a separate depth sensor. For tracking camera's pose the scanner uses ARCore's built in *Pose*-method. In figure x is shown a picture of the 3D model that the app creates from a room.

This application can be used to model everything from indoors scene to single objects and the results are shown in the next chapter.

3.5 Results

Silidragos' low-resolution scanner can be used to store feature points from certain scene, and it gives as a coarse representation of the scene. This model does not give you any small details, but rather just the overall structure. It could not provide a representation of a single object, since its resolution is too small. In Figure 9 is shown the data collected by the application, plotted in Matlab.

For more specific representations the data could be plotted using software like Unity. Choosing different methods for plotting can change the quality of the model.

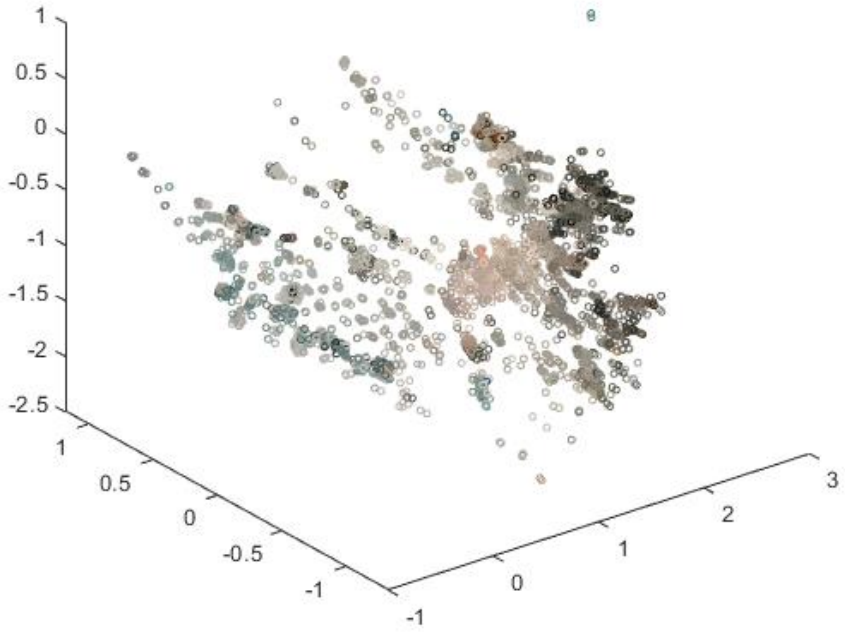

Figure 9. Coarse representation of a scene with Low-Resolution scanner. [16]

Vonásek's 3D scanner's performance and usability were acceptable, as the process happened in real-time and final rendering was fast. The quality of the model when using the most accurate resolution, was usable for models of a larger scene or a room. As we can see from Figure 10, the model is quite accurate overall, but it has few flaws in there too.

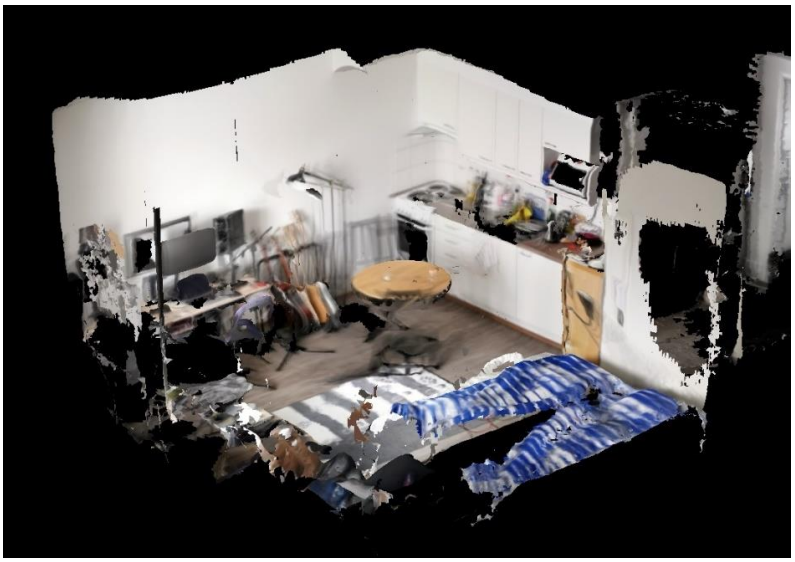

Figure 10. *3D model of a small apartment using Vonásek's 3D scanner using 2 cm resolution.*

Basic idea of the scene can be obtained from the model, but smaller details appear blurry and distorted. When creating a model of a single object, the resolution matters more, 2 cm having the best quality and 8 cm worst. The higher the resolution was, more sensitive it was for slight movement creating artifacts and distortion as can be seen from Figure 11. Over all this applications accuracy is satisfactory, but for certain tasks, models would need some post processing.

Figure 11. *3D model of an acoustic guitar using Vonásek's 3D scanner using 2 cm, 4 cm and 8 cm resolution.*

These results give us a basic idea of what ARCore is capable of. It provides us easy and efficient methods to obtain information of the given scene. The built-in libraries make it easy for developers to develop their own AR applications. However, ARCore might not be the best choice for creating an 3D scanner for reconstruction, because it mainly focuses on the interactions between virtual game objects and planes of the scene. ARCore performs the plane and feature point detections well, but the resolution for feature point detection could be better.

The benefits of ARCore are its usability and community, as they provide lots of information for developers and users. This makes it easy for anyone to start experiencing with ARCore development.

5. CONCLUSIONS

In this study we went through ARCore's fundamental concepts and the theory behind them. Structure from Motion is used to obtain the structure of the scene using only visual data obtained from the device's camera and solving the structure by calculating the correspondences between frames. Visual-Inertial Odometry is a method where the visual data and the inertial data are combined to solve the structure of the 3D scene.

We tested ARCore's features with two applications that use ARCore to perform a 3D reconstruction. First one was a simple point-cloud scanner, that can be used to obtain a coarse estimation of the scene's structure. Second application was used to create a 3D model.

ARCore does help a lot with creating the 3D reconstruction with its ability to detect feature points efficiently and track the pose of our device, but currently it can not perform the reconstruction alone. With ARCore's built-in features we can obtain the point-cloud and 3D structure of our object or scene, but the texturing needs to be done on a third-party application like OpenGL renderer.

In conclusion, ARCore might not be the best tool for reconstruction problems, because it focuses mostly on virtual objects and interacting with them. However, it offers great tools for obtaining information from the scene, but for handling the data, you need to program yourself or use third party applications to visualize this data. 3D reconstruction might become a built-in feature of ARCore later in the future, since it is a very desired feature among the ARCore community.

SOURCES

[1] Milgram, Paul & Takemura, Haruo & Utsumi, Akira & Kishino, Fumio. (1994). Augmented reality: A class of displays on the reality-virtuality continuum. Telemanipulator and Telepresence Technologies. 2351. 10.1117/12.197321.

[2] Victor Adrian Prisacariu, Olaf Kahler, David W. Murray, and Ian D. Reid. Realtime 3d tracking and reconstruction on mobile phones. IEEE Transactions on Visualization and Computer Graphics, 21:557–570, 2015

[3] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, 2007, pp. 225-234.

[4] Bylow, Erik & Sturm, Jürgen & Kerl, Christian & Kahl, Fredrik & Cremers, Daniel. (2013). Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions. 10.15607/RSS.2013.IX.035.

[5] Google – ARCore Overview: https://developers.google.com/ar/discover

[6] M. Martinez, Smartphone 3D Reconstruction for Physical Interactions in Augmented Reality, Dissertation, University of Dublin, Trinity College

[7] Googe – ARCore Concepts: https://developers.google.com/ar/discover/concepts

[8] Google – ARCore Documentation: https://developers.google.com/ar/reference

[9] Laksono, Dany. (2016). CloudSfM: 3D Reconstruction using Structure from Motion Web Service. 10.13140/RG.2.2.34441.29289.

[10] Hartley, R., Zisserman, A., 2003. Multiple view geometry in computer vision. Cambridge university press, Cambridge, UK

[11] Rossi, A.J., Rhody, H., Salvaggio, C., Walvoord, D.J., 2012. Abstracted workflow framework with a structure from motion application, in: Image Processing Workshop (WNYIPW), 2012 Western New York. IEEE, pp. 9–12.

[12] Robertson, D.P., Cipolla, R., 2008. Structure from motion, in: Practical Image Processing and Computer Vision. Wiley, Blackwell

[13] Feature-based visual odometry prior for real-time semi-dense stereo SLAM, N. Krombach, D. Droeschel, S. Houben, S. Behnke

[14] Cheng, Yang & Maimone, Mark & Matthies, Larry. (2006). Visual odometry on the Mars Exploration Rovers - A tool to ensure accurate driving and science imaging. Robotics & Automation Magazine, IEEE. 13. 54 - 62. 10.1109/MRA.2006.1638016.

[15] Shelley, M.A., Monocular Visual Inertial Odometry on a Mobile Device

[16] Silidragos: https://github.com/silidragos/colorfulcoding-projects/tree/master/1-ARCore%20Scanner

[17] L. Vonásek, https://play.google.com/store/apps/details?id=com.lvonasek.arcore3dscanner&hl=fi

[18] L. Vonásek, https://github.com/lvonasek/tango/wiki/3D-Scanner-for-ARcore