# 3D Reconstruction Using Tango

Quentin Gautier, Steven Lee, Sungwook Son, Jin Yu

June 9, 2016

Abstract

Reconstructing models in three dimensions change the way humans visualize. Archaeologists use visualization to get further knowledge on their findings. Many archeologists today still use the old method of pencil and paper. This project allows archaeologists to use Google Tango tablet to reconstruct their findings while providing real time mapping and feedback. The Google Tango is a portable tablet used in all types of Augmented Reality and Virtual Reality applications. Our implementation of On-Board SLAM generates a 3D reconstruction that efficiently uses both CPU and GPU to map while doing calculation. After real field testing in Guatemala, we fixed drift and loss issues found on previous implementation.

## 1. Introduction

The goal of this project is to implement technological advancements in current field of art and archeology for further visualization. This visualization not only allows artists and archeologists to detail their findings, but also allows them to build a highly scaled model with much deeper depth and detail. Even today, many of the artists and archeologists are limited from many of these modern resources. These artists and archeologists use an out-dated method of drawing a 3D model on paper with pencil. This use of long established method, when compared to the use of modern technology, falls behind in many qualities; as the size of mapping gets larger, visualization through the use of traditional method gets obscure. Thus, with bigger findings, this 3D reconstruction provides better accuracy and efficiency. This project aims to provide artists and archeologists with a mobile system that can reconstruct an underground environment, with real-time partial or full feedback.

The 3D reconstruction Google Tango team is composed of three members and one supervisor. In our team we have Steven Lee as SLAM specialist, Jin Woong Yu Tango specialist, and Sungwook Son as Tango/SLAM jack-of-all trades. Our team leader, Quentin Gautier, is a current member of Engineers for Exploration organization at University of California, San Diego. This organization is involved in many engineering solutions that goes beyond current technology. In collaboration with this organization, we put an effort to develop 3D reconstruction capability on a portable device, which will be used to reconstruct the caves of Guatemala.

The main piece of hardware is the Google Tango handheld device that comes with an abundance of sensors including but not limited to 3D depth sensors, gyroscope, accelerometer, and motion tracking camera.  It has a NVIDIA mobile processor that doubles as a GPU. Project Tango can be easily obtained with a price of $500. Its superb processing ability and sensors allows us to reconstruct the area in 3D. Tango is very light and portable that it could be used anywhere. For this project we have used project Tango to map the caves in Guatemala, but the possibilities are infinite. There are already two devices in our possession.

We tried two different implementations of Simultaneous Localization and Mapping (SLAM). There is not one SLAM implementation to fit all requirements, and moreover, we want it to match the Google Tango capabilities. Through various research, we came down to RTAB and On-Board slam. The advantage of implementing SLAM algorithms for 3D reconstruction over any other method is that as the mapping begins, highly efficient SLAM algorithm also calculates the position relative to the world and objects around them. Moreover, implementing SLAM algorithms on Google Tango tablet  attempted to fix drift and loss issues, found on previous Kinfu application.

However, we were only able to to get the On-Board slam working on Google Tango. When Quentin took this implementation to Guatemala for real field testing, this algorithm was only able to hold such amount. When mapping bigger findings, this application implemented with On-Board slam algorithm, crashed after mapping at most few minutes. Despite this, when we did our testing, we were able to walk around the second floor of the CSE building without much problem. One problem we encountered was processing issue. As the scale gets larger, because Google Tango has limited power, the algorithm slows down over time. Moreover, when trying to extract the .obj file after successfully mapping, this process took a while.

One issue we encountered while mapping with Google Tango tablet was limited battery life. Because this application does so much computation, it drains the battery really fast. As the SLAM algorithm further develops, the battery may be an impediment for a larger mapping.

# 2 Technical Material

## *Our Approach*

The Google Tango is an Android device with APIs available in Java, C, and Unity.  The Tango device has out of the box motion tracking and depth sensing capabilities due to its abundance of sensors.  Some of the key sensors include the motion sensor camera and 3D depth sensor.  It uses a NVIDIA Tegra processor with CUDA support for graphics computing applications.

The initial goal of the project was to improve upon an existing application that had 3D mapping capabilities.  By scanning an area with the Tango's camera and depth sensors, the Tango could save point cloud information to be later examined as a model file.  The application could then be used by Archaeologists to map their findings; more specifically Guatemalan caves the Engineers for Exploration organization at UCSD would be exploring.  The Tango would have to possibly map long and winding tunnels with rocky surfaces.  The existing application did not have drift correction so over long periods of time, the motion tracking would lose its position in the world.  Any 3D mapped models would not be accurate due to this drift.  The solution was to integrate a simultaneous localization and mapping algorithm (SLAM) that would correct for this drift.

The existing application required a great deal of hardware level programming due to its graphics processing demands.  The Android OS which uses the Java language has support for hardware level tinkering through the Java Native Interface (JNI) using the C language.  Applications created with such support are called Java Native Applications (JNA).  This added a great many new difficulties to our team which was unfamiliar with JNAs.  Google provides C, Java, and Unity APIs.  We addressed our weakness by developing in the Unity environment.

Unity is a very popular 3D game development program. Due to its ease of use, it has found a place among triple A companies and independent developers alike. Some notable products created with Unity include Blizzard's Hearthstone and the Kerbal Space Program. Unity supports many different types of platforms as well including our target of Android. The game creation process exploits the object orientated nature of video games. Game objects have a variety of different components attached to them to define their behaviors. Components can range from Sprite Renderers, 3D Models, Transforms, Rigidbodies, Scripts, etc. Scripts for Unity can be written in C# or Javascript. We decided to go with the Unity favorite of C#. Scripts allow even more complex behaviors to be programmed into the Game Objects.

## *3D Mapping*

The 3D mapping of our application involves two separate features.

1. The ability to record and later export data involving the Tango and the recorded environment
2. The ability to give real time feedback to the user on what has and has not been recorded

The data to record includes pose data, camera images, depth data, and model data. The pose data is the orientation of the Tango device in space at a given moment in time. The camera images are the images seen by the rgb camera. The depth data is the sensor data from the depth camera which is taken a rate of 5 Hz (much slower than the actual camera data). The model data is the actual environment represented as a mesh. The Tango comes with convenient API functions that are callbacks to receive pose data, camera images, and depth data. The model data cannot be captured as simply and is created through a mesh renderer.

The mesh renderer is entirely contained in a mesh rendering script. The mesh is represented as a multitude of in-engine Game Objects with shader and transform components. The shader gives the mesh the color of the object being represented. The transform determines the position and orientation of the mesh in the application world. The transform is extremely important as an accurate transform determines the accuracy of an exported model.

The mesh rendering processing goes through quite a few steps. The mesh renderer subscribes to the Tango's depth sensor callback. The Tango feeds the mesh renderer depth data allowing the renderer to find flat surfaces. Each surface is then transformed into a grid in which a square on that grid can then be meshed. Depending on the resolution determined beforehand (we found 5 cms to be efficient), the grid will have differently sized squares. A point cloud is generated for the purpose of coloring each mesh to better represent what is being mapped.

To improve the user experience, the computationally heavy mesh rendering is done on a separate thread. When indices are seen that require meshing, they are pushed into a queue data structure to await their turn. This backlog is processed for a small portion of the computation cycle every cycle. Of course this means that mapping a brand new area creates a surge in meshes that need to be processed. By keeping the Tango focused on a previously rendered area or by pausing the rendering process through the GUI, the backlog can be processed much more quickly.

## *SLAM*

Many of the SLAM algorithms we researched including ORB-SLAM, RGBD SLAM, and LSD SLAM cannot be compiled through the Unity's build process. This meant that we had to rely on Tango's onboard SLAM. The onboard SLAM named "Area Learning" by Google, functions very similarly to these other SLAM algorithms. The Tango is constantly viewing and remembering specific landmarks in its environment. As the user takes the Tango away from the starting location and back, the Tango will remember landmarks from the starting location. It then realizes that it has formed a loop and the loop can be “closed” through recalculating positional data. The Tango stores the pose data from when it began recording and when a loop is detected it compares the current pose data to the starting pose data. Both poses should technically be the same as the Tango has simply returned to the position it has started in but mistakes in position accumulate over time creating drift. The difference between the pose the Tango started with and the pose the Tango ended up with can be used to recalculate all the poses along the Tango’s route.

*Weaknesses*

The current implementation for Archaeological recordings has some weaknesses that need to be addressed. The application is not built as a JNA thus it is not using 100% of its hardware capabilities. This can be seen in the drops in frame rate and backlog processing rate of the application.

The application uses the onboard SLAM algorithm which cannot be examined or tuned by developers.

The application is built in Unity which is limited in its compiling and building procedures.

*Strengths*

The Unity engine is simple to learn and powerful to use. By using the Unity APIs, we did not have to spend manpower and resources learning JNA.

The Tango is light and portable weighing in at 0.82 lb (370 g).

The application maps quickly and accurately for its given hardware.

This subsection describes how to run the 3D reconstruction system. The following will demonstrate requirements and instructions to setup and build On-Board SLAM on Google Tango tablet.

2.1.1 Requirements

Components:

- Google’s Project Tango
- Laptop

2.1.2 Set Up Instructions

Environment used for this build

- Windows 10 / Mac OS X
- Android Studio 2.1
- Unity version 3.4.1.f
- Google USB Driver

Setup

1. Install Windows 10. Windows 10 is a very common operating system among PCs. Usually Windows 10 comes with the computer itself. However, if Windows 10 is not available, register and download. Download:[https://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?ws=94fd1fe3-eb9b-e011-969d-0030487d8897&vsro=8](https://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?ws=94fd1fe3-eb9b-e011-969d-0030487d8897&vsro=8)

2. Install Android Studio 2.1. Android Studio is a IDE(Integrated Development Environment) for android.
   Download: https://developer.android.com/studio/index.html
3. Install Unity (3.4.1.f) via Unity's Download Assistant. Unity is a tool for 3D game engine. Google Tango has Unity support for AR(Augmented Reality) and VR(Virtual Reality). Make sure to select "Android Build Support" and "Microsoft Visual Studio Tools for Unity" is recommended.
   Download: http://unity3d.com/get-unity/download
4. Install Tango Unity API. Tango Unity API allows Unity to use APIs specifically for Tango.
   Download: https://developers.google.com/tango/downloads#tango_sdk_files
5. Install Google USB Driver. Google USB Driver is a driver to let the Windows computer to recognize android devices such as Tango.
   Download: https://developer.android.com/studio/run/win-usb.html

### 2.1.3 Building App

1. Add the Android SDK path to Unity by selecting Unity -> Preferences -> External Tools
2. Import the Tango SDK into Unity, select Assets -> Import Package -> Custom Package, then select TangoSDK
3. Clone https://github.com/ssl024/TangoMapper.git
4. Build apk

### 2.1.4 Running App

1. Transfer the apk to Tango
2. Using File Viewer app install the apk
3. Hold Tango still when opening because of the calibration.
4. Type in appropriate values for the field
5. The app will start mapping the area
6. If desired to export, press export and view file from path written on step 4

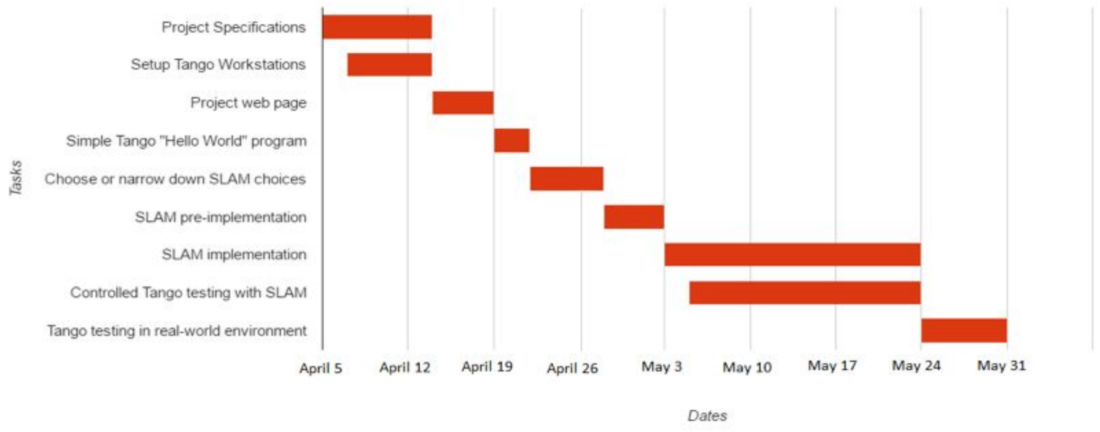
## 2.2 System Usage

# 3 Milestones

Initial milestone

| Week | Objective/Milestones | Status | Artifact/Comments |
|---|---|---|---|
| 3 | Complete Project Specifications<br>Setup Tango workstation | Comp<br>Comp | Specifications<br>Setup Instruction |
| 4 | Complete Project web page. | Comp<br>Comp | BitBucket Wiki<br>Video Demo - Hello World |

|  |  |  |  |
|---|---|---|---|
|  | Run simple Tango program running such as “Hello World” |  |  |
| 5 | Oral Update.<br>Understand Tango functionality. | Comp<br>Comp | Presentation Slides<br>Video Demo - point cloud demo |
| 6 | SLAM pre-implementation. | Comp | Video Demo - mesh rendering |
| 7 | Milestone Report.<br>SLAM implementation complete. | Comp<br>Comp | This<br>Video Demo - SLAM implementation |
| 8 | Tango testing in a controlled environment | Comp |  |
| 9 | Tango testing in real-world | Comp<br>Comp |  |
| 10 | Catch slips.<br>Final presentation<br>Final Video | Comp<br>Comp<br>Comp |  |



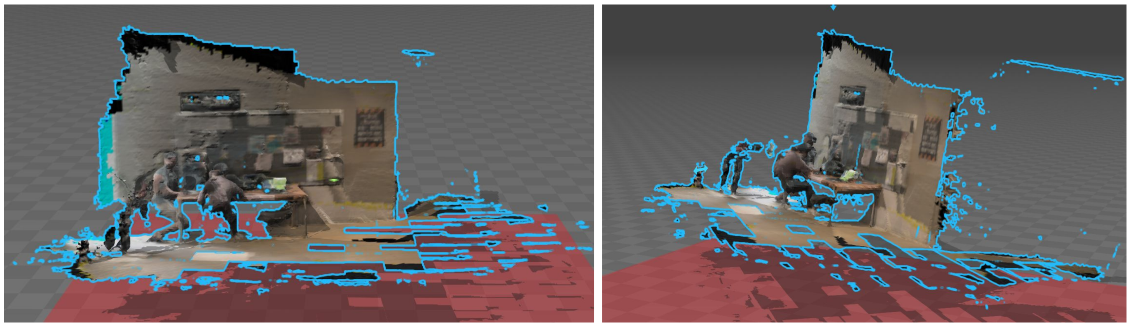
By the end of initial milestone (week 5),

- Complete Project Specifications and setup Tango workstations.
    - To set up Tango workstations for use and download NDK and SDK,
    - https://developers.google.com/tango/apis/c/#install_the_ndk
- Complete Project web page and a simple Tango task such as “Hello World” program
    - After successfully setting up Tango workstation, we tested multiple simple programs

- Understand Tango functionality.  SLAM research
  - After looking at multiple SLAM algorithms on https://openslam.org/ we came down to 4 different algorithms: On-Board, RGBD, RTAB, ORB

By end of second milestone (week 7),
- SLAM pre-implementation
  - While finalizing which SLAM implementation best fits our purpose, we got mesh rendering implemented on Tango using Unity.
- SLAM implementation complete
  - We couldn't finish this until beginning of week 8.

By end of final milestone (week 10),
- Controlled Tango testing with SLAM
  - After implementing On-Board SLAM, we mapped a small area second floor of CSE building



- Tango testing in real-world SLAM

Notes:
- New Objectives:
  - Controlled Tango testing with SLAM.
  - Using ORB-SLAM with Unity API for mesh rendering.

Issues encountered:

At first, we worked on two different implementation of SLAM algorithm: RTAB and On-Board. On our first development of RTAB SLAM algorithm, we were able cross-compile all dependencies and successfully got the APK file to put in the Google Tango tablet. However, after updating some features and pulling from the repository, cross-compiling failed and failed to reproduce the APK.

Also, downloading the newest version of all softwares and sources are not the best idea. While implementing On-Board SLAM, we had lots of dependency issue. Staying with prior version makes life easier.

## 4 Conclusion

The 3D Reconstruction project using Google Tango tablet shows how modern artists and archeologists can use this device and application to effectively map their findings. Compared to their previous mapping, which relied on hand-drawn structures, this project will definitely be an impetus for many artists and archeologists to move on forward. Although this project has limitations to mapping big findings, it can still map much more efficiently and shows high details due to use of depth sensor. Moreover, with traditional method of drawing, there are limitations to how much mapping can be done. Use of this 3D reconstruction application allows the user to extract their mapping and save it as an .obj file which then can be viewed on computer through 3D viewer, and can even be printed. Because we do not have statistics to this project, we cannot say how effective the use of this project will be compared to other algorithms or previously used methods to map the findings.

Improvements can definitely be made. There are many different simultaneous localization application mapping algorithms and it is hard to end up with one perfect algorithm. Because our use of On-Board SLAM algorithm, there are limitations, as we cannot tweak it as much for variability. One way to achieve better solution would be to research more different SLAM algorithm and find out which algorithm is good for long distance and short distance. Because archeologists like to map long distance caves, On-Board SLAM was probably not the best idea. Our prior implementation of Real Time Appearance Based Mapping showed high details but this algorithm is probably not the best idea for long distance mapping, as we found out that the efficiency of this algorithm degraded over time.

Compared to our other team, Team Kinect, the use of Google Tango was much portable. However, due to its small size, there were computational limits. For Team Kinect, they used Microsoft Kinect which has a camera and multiple sensors. Moreover, the computation is done on a laptop using its central processing unit (CPU) and graphics processing unit (GPU). For testing, Quentin Gautier used a laptop with i7 processor high end graphics processing unit, enabling smooth rendering and producing rich quality reconstruction. Although Kinect team produced better results during the test at Guatemala, its portability was not so great. For 3D reconstruction done with Google Tango tablet, all computation and rendering was done on the tablet while 3D reconstruction done with Kinect had to carry around power source for kinect and a laptop for rendering. Overall, although not perfect, both team produced a full working 3D reconstruction application. Thus, we believe our project was successful.

## References

https://openslam.org/
http://wiki.ros.org/rtabmap
https://developers.google.com/tango/hardware/tablet
https://unity3d.com/unity/editor
https://developers.google.com/tango/apis/c/#install_the_ndk