Joshua Tang
11136 Magnolia dr
Cleveland, OH 44106-3903

Dr. Robin Evans
Director of Professional & Technical Communication
Case Western Reserve University
Guilford House 403
11112 Bellflower Road
Cleveland, OH 44106

December 06, 2014

Dear Dr. Robin Evans,

Enclosed you will find a proposal for a “*3D Scanner for Android Devices*”. As 3D models are becoming ubiquitous to modern life, most are still painstakingly created by artists, designers and engineers. This proposal demonstrates how we could have a simple, low-cost and effective way to create 3D models from the real world. This could accelerate game development and rapid prototyping and could potentially lead to new more intriguing facets of 3D modeling.

Our proposed method of getting 3D model generation to the masses is by creating an simple and easy to use Android app which let's the user take a video of an object and create a 3D model from that. Using very recent technological breakthroughs let's us generate the 3D model in real-time.

Producing this app is fairly low-buget, with a total budget of $1500, however it is a considerable time investment.

One section of the proposal was omitted: the Anticipated involvement of the faculty member/investor/client,” section was omitted because this is a project that I am both able and willing to undertake entirely on my own. The only outside help that I will need will be getting access to the CWRU HPC (High Powered Computer Cluster) which I already have access to.

I would like to thank Raul Mur Artal for his helpful answers to my questions when I was implementing ORB-SLAM. If you have any questions or comments, I can be contacted via e-mail at jlt81@case.edu or by phone at (216) 800-5350. Thank you for taking the time to review my proposal.

Thank you for your consideration,

Joshua Lee Tang
CWRU Department of Electrical Engineering and Computer Science
Class of 2016

3D Scanner For Android Devices

Submitted To : Dr. Robin Evans
Submitted By: Joshua Lee Tang
December 06, 2015

Abstract: 3D reconstruction, the process used to turn objects and scenes from the real world into 3D models has become very accurate in the past five years, and can now be executed with a high level of precision. The ability to create 3D scans is very practical for artists, designers, engineers and the maker community, as it permits rapid prototyping and replicating the world around us with ease. This process is currently available to the general public by using expensive attachments to their smartphones and tablets. This extra hardware is unnecessary to 3D scanning, which really only requires a camera. We aim to create an Android app that will be easy to use and accessible to anyone to create 3D models from the world around them using only their smartphone. The app uses feature based monocular SLAM in order to make a semi-dense probability mapping of a video taken by the smartphone. This initial mapping will be made denser, triangulated and saved as a 3D model. Funding for this project will be minimal, around $1500, but I will request access to the CWRU HPC to speed up how fast data can be processed and accelerate development.

1. Project Description

As technology has progressed over the last century, 3D models have become an ever bigger part of our lives. They are present in our art, our entertainment, our engineering processes and are becoming present in an increasing number of fields. Currently, the vast majority of 3D models are painstakingly created by artists, engineers and designers from scratch. A significant portion of 3D models represent real world objects and environment. With the advent of smartphones and advances in computer vision, we can now create 3D models quickly and effectively from a video of the object. My project proposes to create a simple and accessible smartphone app which will allow users to scan objects in the real world and create 3D models of those objects. The app will allow users to utilize a sort of “copy-paste” methodology on physical objects. The app will also let users not only make models of objects that they encounter, but also let them modify those objects and make molds for those objects.

Currently all commercially available software that exists to create 3D models of objects on smartphones is very slow and clunky(see 123D Catch). The app we are proposing to make will be much more user friendly and much faster as it will create the model on the phone itself rather than sending the data out to a server to be computed. We will also be the first publicly available app to permit the users to directly edit the models resulting for the scan and create a mold for said models. In terms of research, in October, after I had proposed this project. Microsoft Research published the Mobile Fusion [14] paper which behaves similarly to our proposed app. While this is unfortunate, our app will being using a different methodology to obtain the 3D models and as such it will be interesting to compare and contrast our results with those of Microsoft's Oxford research team.

My app, will be extracting ORB features[5] from the video it is taking in order to perform monocular ORB-SLAM[6] on the scene that is being presented to it. Using the keyframes selected from the SLAM algorithm in order to create a semi-dense 3D mapping of the depth of each point from the frames [2]. The resulting mapping will be turned into a mesh which will then be available to view, edit and export to 3D printers, game engines, virtual worlds, anything really. These scans can then be used in a variety of projects including video games, figurines, souvenirs, house tours and engineering prototypes, either as a final product or a base to be embellished upon.

2. Background information

2.1 Camera calibration

Cameras are not perfect devices, however most of the algorithms we use to assume a perfect pinhole camera. This can be corrected by properly calibrating the camera. This is traditionally done by using a known image (generally a chessboard, a set of symmetrical dots or a set of asymmetrical dots) and calculation how far off the camera is when looking at the image from different angles. While this method is generally precise, it is rather clunky, slow and annoying for a layman to use as they would need to print out the calibration image and take a number of pictures of them while the app is in calibration mode. Since this would be a hassle for the average user, we have elected to automate the process of calibrating a camera. Several techniques exist to do this, including a technique that is capable of determining the calibration parameters by using a single frame [17]. Some methodologies have even been attempted to solve the particular problem of how a smartphone's camera's focus changes[15]. However, many of these techniques are not generalized enough, requiring specific conditions to such as straight perpendicular lines and other Manhattan image constraints to be present in the image. The app will need a more generalized technique in order to be more robust and usable in any situation without having any restrictions of where or how to calibrate the device. We decided to create our own more generalized camera calibration module using machine learning to learn camera calibration from a dataset that we have previously created for Micheal Lewicki's Natural Perceptions Lab to train a neural network to identify the camera's calibration during runtime.

2.2 Feature extraction

In order for our algorithm to work, we are going to need to be able to rapidly extract features from the scene in order for us to perform the SLAM algorithm. The features that are picked are going to need to be both scale, translation and rotation invariant, which is to say that they will remain the same regardless of the image's scale rotation or position. Looking through the literature, we find that the initial influential paper in this domain is the paper on SIFT [6], which describes exactly the kind of features we are looking for based on those feature's mathematical properties. SIFT is a little bit of a hack since the selected mathematical features that are picked were hand picked by researchers rather than being completely based on vision. Despite this fact SIFT works quite well. The main issue with using SIFT is that it requires a lot of computations and as such is very slow. Next, we look at SURF[7], which is an accelerated form of SIFT. SURF possesses the speed we are looking for, however it is less precise than SIFT and will break if sufficient noise is present in the scene. Which finally brings us to ORB[5] which is a combination of FAST and BRIEF. It has the speed and precision we are looking for and the only downside to using it is the fact that it breaks when exposed to pure rotations and like the other feature types, it doesn't track moving objects. I've decide to go with ORB for the project specially since it is the feature used in the ORB-SLAM dense probability mapping[1][2].

2.3 Tracking and Mapping

Now we are going to need to track the invariant features we've extracted from the scene in order keep track of where they are even when they are obstructed by other objects. Additionally we need to be able to map these features to a 3 dimensional space, giving each of them a place in the physical space. All of the algorithms we will now be looking at functino by taking multiple keyframes throughout time, extracting some sort of feature from these keyframes and uses the information from the other keyframes to create a map. I took note of the work done in the Parallel Tracking and Mapping paper [8] which uses the FAST features and demonstrates how running multiple parallel processes to track and map all of the points is better than any method that exist prior to it and is very useful to helping a virtual environment interact with the real environment. The LSD-SLAM algorithm [11] get's closer to what we are looking for by creating a dense mapping of of it's environment which makes it good for our purposes, however, the algorithm's precision leaves something to be desired. We then come to DTAM[9], DPPTAM[10] and ORB-SLAM[1][3], all of which suit our purposes rather well in term of precision and speed. In the end I chose ORB-SLAM because it seemed like the easiest to implement in a relatively short time period and uses less processing power than the other two while still giving us great results. In addition we know that ORB-SLAM has been successfully been implemented on phones before [16] which, confirms the feasibility of this project, even though that example did not implement the semi-dense probability mapping [2] that we will be using.

2.4 Pose Estimation, Relocalisation and Auto-calibratio*n*

In order to get a good mapping of a scene, we are going to need to know precisely where are camera is. To do this, we are relying on good relocalisation [4] to tell us when our pose estimation is wrong and to allow us to find the correct pose by looking at the points we have already successfully mapped. This is a crucial part of both Tracking and Mapping the scene [3]. In addition, after some initial testing, we know that even for the same model of android phone, the camera is not necessarily calibrated the same and thus we need to calibrate the algorithm to the needs of the camera. While we could let the user calibrate the camera themselves using the method OpenCV has for calibrating cameras, this would be obnoxious to our users and would make the app less user friendly. Instead we can automate the calibration on the phone automatically[15], which should make using the app much easier.

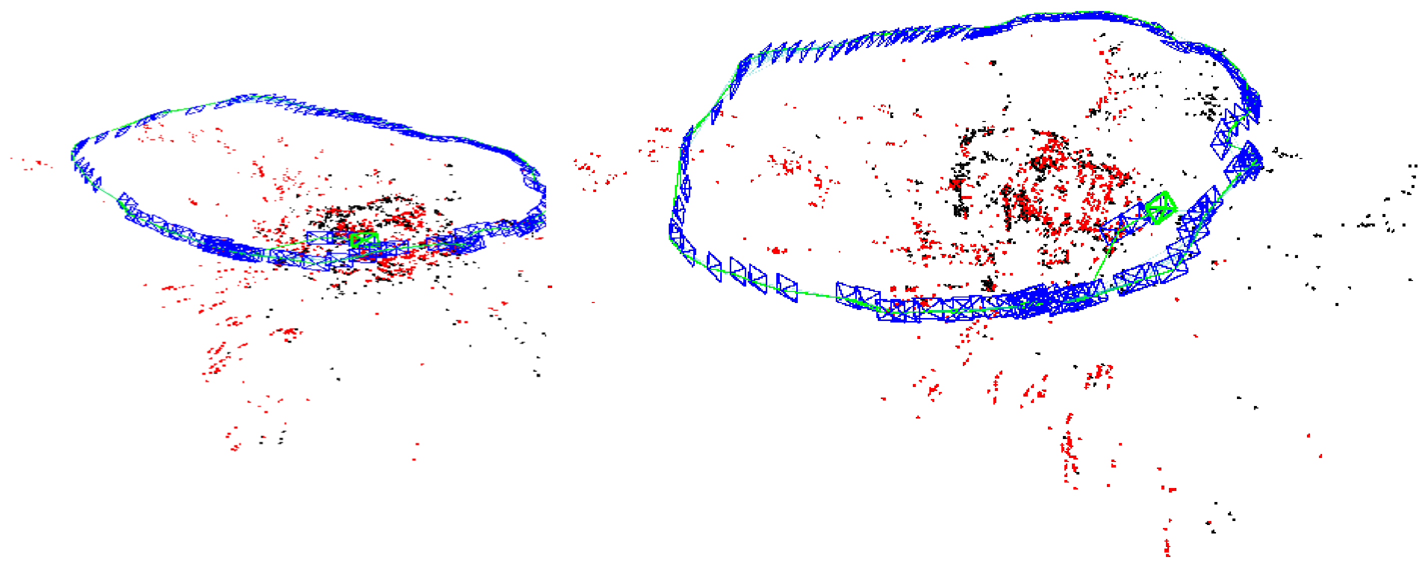
2.5 Creating and Smoothing a 3D Mesh

Finally, once we have the dense probability mapping of an object, we are going to need to turn it into a mesh so that we can edit it. There are many different ways to do this but using spherical parameterization [13] seems like a good idea due to fact our point cloud won't be particularly dense. Other options [12] are available to us, but currently it seems like spherical parameterization will be our choice for this project. Editing the resulting model and displaying it should then be easily feasible. We should save the model as an STL or .obj file so that it is easier to share, print or use in other projects.

3. Plan of work

3.1 Proof of concept

Preliminary work on this project started in September of 2015. From September to December, we checked we created a basic Android app that allowed us to calibrate the devices camera using traditional methods, that could take video and could send said video to a server. The server we developed in Ubuntu 14.04 runs ROS Indigo in order to run an implementation of ORB-SLAM. This entire setup let's us calibrate the Android device's camera, take a video, send the video to the server and generate a sparse mapping of the video. This proof of concept separates the app development from the reconstruction algorithms that are being run server side. This is very practical as it allows us to write an implementation of all of the systems that will be in the final app on the server where it will be easier for us to debug and find out what challenges face us when developing the app.



We implemented the proof of concept first by creating an Ubuntu 14.04 server and installing ROS Indigo on it. We used a combination of the code from the ORB-SLAM paper[1] and our code to write an implementation of ORB-SLAM. We then created a controller program which was set to receive the video from any source and turn it into a RosBag (which was the required filetype to run our implementation of ORB-SLAM). We then proceeded to write an app that let's the user calibrate the camera the traditional way and let's the user take videos, send them to the server and see the resulting 3D models. The probability semi-dense mapping modules still needs to be fully implemented. The core of the module has been implemented but development has stopped due to finals. The module is based on [2] and utilizes our existing implementation of ORB-SLAM,

We plan on finishing the semi-dense probability mapping module on the server by the third week of December and have the entire proof of concept operational by the second week of January.

3.2 Auto-calibration

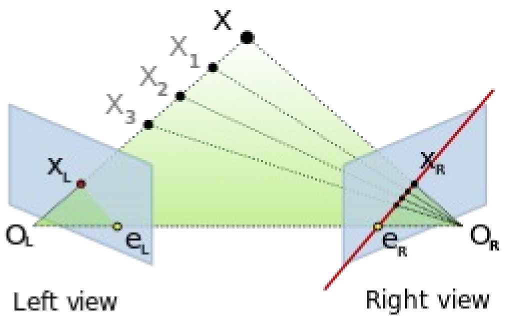
While we were working on the proof of concept, we also developped a machine learning solution to calibrating a camera using CWRU's HPC to speed up how quickly we could train our algorithm. The algorithm uses a neural network trained on the motion parallax dataset from CWRU's Natural Perceptions Lab. The network checked where epipolar geometry predicted where points would be compared to where they actually were and used the difference between those 2 locations to change the calibration parameters until it converged on a network that was good at predicting a variety of different calibration parameters based solely on a few seconds of video from that camera. Once the artificial neural network has been properly trained, we will select the best instance of it and use that to create an auto calibration module for our app. This should be done by the end of the first week of the spring semester of 2016.

3.3 ORB-SLAM

While we can't reuse much of the code from the proof of concept, we can use it's codebase to inspire both the structure and the content of our implementation of ORB-SLAM. While the fucntionality will be the same, we will need to have the code run natively on android, removing it's reliance on ROS and perhaps even on OpenCV if we want to avoid having to make people download the OpenCV Manager app. We will write the implementation in C++ porting over parts of the OpenCV library (since it's source is available under a BSD license) so that all of the code is contained in the resulting library. We can then port that library over to Android and run it all natively. This should be accomplished by the sixth week of classes in the spring of 2016.

3.4 Semi-dense probability mapping

Converting the code from the proof of concept into for the semi-dense mapping module. The module uses the ORB-SLAM module extensively and is run simultaneously to ORB-SLAM. It uses the keyframes from ORB-SLAM to determine the probable depth of each pixel in each image that was used to create the mapping for ORB-SLAM and uses the other images to verify the accuracy of it's estimations. The module is expected to be finished by the eight week of classes in spring 2016.



3.5 Creating a dense mapping

The point cloud resulting from the semi-dense probability mapping is very accurate but doesn't shoe the full picture. We need to create a module that takes the skeleton provided by the semi-dense mapping and uses it to create a denser mapping of the scene. Making another pass on all of the keyframe from ORB-SLAM while using the semi-dense skeleton let's us refine our search for point at appropriate depths to complete the skeleton. We expect this module to be finished by the tenth week of classes in 2016.

3.6 Creating and Saving a 3D model

Now that we have a dense point cloud, we need to get a 3D model from it, so we need to crate a module that triangulates the dense mapping into a basic polygon based 3D model which will then be smoothed and saved in the STL or .obj formats. The STL and .obj formats were chosen because while there is no standard file-type for 3D models both of these file-types are very commonly used and are generally the file-types used by 3D printers. The app should be able to successfully generate 3D models by the twelfth week of classes in 2016.

4 Project Timeline

week 0 will be the 3rd week of december 2015

Week 0	Week 4	Week 10	Week 12	Week 14	Week 16
Proof of concept	Auto-calibration	ORB-SLAM	Semi-dense probability mapping	Densifying the point cloud	Creating and saving a 3D model

5 Qualifications

I am a senior at CWRU pursuing an Bachelors of Science in Computer Science with a minor in Artificial Intelligence. I have been doing research in the field of structure from motion for Professor Lewicki's Natural Perceptions Lab for the past year and a half. I have taken several graduate level classes covering various aspects of computer vision, machine learning and other AI topics. Additionally I have been writing Android apps for the past three years. Additionally, I have created multiple projects in the past two years that involved either some form of image processing or the use of OpenCV. I believe my experience in the field of computer vision, specially my experience with structure from motion makes me very qualified to develop this app.

6 Budget

The budget for this project will be minimal as most of this project will be accomplished by investing my personal time into it. There exist two elements that could make this project cost anything at all. CWRU could theoretically charge me for my use of the HPC, but as I am a member of Dr. Lewicki's lab, this should not be a problem as I already have my own account on the HPC. I will need to have a few different Android devices so that I can test the app's functionality on multiple different devices. The mean cost of an Android capable of running the app is around $500 so I will be asking for $1500 in order to have at least 4 drastically different devices including my own to test on, assuring that the app works on different devices.

References

[1] R. Mur-Artal, J. Montiel and J. Tardos, 'ORB-SLAM: A Versatile and Accurate Monocular SLAM System', *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147-1163, 2015.

[2] R. Mur-Artal and J. Tardos, 'Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM', Robotics: Science and Systems. Rome, Italy, July 2015.

[3] R. Mur-Artal and J. Tardos, 'ORB-SLAM: Tracking and Mapping Recognizable Features',Robotics: Science and Systems (RSS) Workshop on Multi VIew Geometry in RObotics (MVIGRO), Berkeley, USA, July 2014.

[4] R. Mur-Artal and J. Tardos, 'Fast Relocalisation and Loop Closing in Keyframe-Based SLAM.', IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, June 2014.

[5] Rublee, Ethan, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 'ORB: an efficient alternative to SIFT or SURF.' In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2564-2571. IEEE, 2011.

[6] D. Lowe, 'Distinctive Image Features from Scale-Invariant Keypoints', *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[7] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, 'Speeded-Up Robust Features (SURF)', *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346-359, 2008.

[8] Klein, Georg, and David Murray. "Parallel tracking and mapping for small AR workspaces." In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225-234. IEEE, 2007.

[9] Newcombe, Richard A., Steven J. Lovegrove, and Andrew J. Davison. "DTAM: Dense tracking and mapping in real-time." In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2320-2327. IEEE, 2011.

[10] Concha, Alejo, and Javier Civera. "DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence." In *IEEE/RSJ international conference on intelligent robots and systems*. 2015.

[11] Engel, Jakob, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-scale direct monocular SLAM." In *Computer Vision–ECCV 2014*, pp. 834-849. Springer International Publishing, 2014.

[12] Fabio, Remondino. "From point cloud to surface: the modeling and visualization problem." *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 34, no. 5 (2003): W10.

[13] Zwicker, M., and C. Gotsman. "Meshing point clouds using spherical parameterization." In *Proc. Eurographics Symp. Point-Based Graphics*. 2004.

[14] P. Ondruska, P. Kohli and S. Izadi, 'MobileFusion: Real-Time Volumetric Surface Reconstruction and Dense Tracking on Mobile Phones', *IEEE Trans. Visual. Comput. Graphics*, vol. 21, no. 11, pp. 1251-1258, 2015.

[15] Saponaro, Philip, and Chandra Kambhamettu. "Towards Auto-calibration of Smart Phones Using Orientation Sensors." In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pp. 20-26. IEEE, 2013.

[16] Shridhar, Mohit, and Kai-Yuan Neo. "Monocular SLAM for Real-Time Applications on Mobile Platforms." 2015.

[17] Deutscher, Jonathan, Michael Isard, and John MacCormick. "Automatic camera calibration from a single manhattan image." In *Computer Vision—ECCV 2002*, pp. 175-188. Springer Berlin Heidelberg, 2002.