Project title: *PTAM on the iPhone*

Project report for Course: *Numerical Geometry of Non-rigid Shapes, Spring Semester 2010*

Lecturer: *Dr Michael Bronstein*

Prepared by: *Aaron Wetzler*

Description: *Porting PTAM to the iPhone and enabling it to work with the live camera feed at frame rate.*

Introduction:
Extracting a user's 3D pose relative to some fixed world coordinate frame is an important component of any augmented reality application involving mobile phones or cameras of any kind. The simplest and most popular approach is to use a set of fiducial markers to extract ones 3D position.There are various SLAM (Simultaneous Localization and Mapping) algorithms , most notably PTAM (Parallel Tracking and Mapping), that exist that work in environments where there are no known markers and instead uses chosen interest points from the camera feed to rely on their ability to provide pose estimates. PTAM was presented in 2007 and in 2009 its authors successfully redesigned various elements of the algorithm and ran it on an iPhone 3G.

Project objective:
Apart from the PTAM author's own port (known as MiniPTAM) to an iPhone an known an extensive online search at the time of starting this project (July 2010) yielded the result that no other known academic institution, individual or company had ported PTAM to a mobile device. The operative word here is ported because reimplementing the entire algorithm would be a very difficult task despite the two papers on the subject. The most feasible and widespread use of PTAM is generally as a black box that produces a pose and its inner workings can be more or less ignored. Within the last month (October 2010) videos of a semi functional version of PTAM running on a Samsung Android phone appeared on YouTube [1]. They were created by a research group called AVatAR from the Graphics lab at Kookmin University in Korea.

The primary objective of this report is to ducment the implementation of a port of PTAM to the iPhone 4 and make use of this phones much stronger graphics hardware and CPU to improve on the results of MiniPTAM shown in [2].

The original PTAM:
George Klein's PTAM [3][4] for the desktop has to be compiled with libCVD [5], TooN[6] and gvars[7].

LibCVD: This is a powerful C++ vision library from The University of Cambridge vision group. It has a large amount of functionality that is not used by PTAM and therefore includes unnecessary external dependencies. It can be linked as a library on OSX but in order to use it on the iPhone (especially if one intends to eventually make it to the AppStore) the code needs to be statically compiled for the phone which is ARM architecture.

TooN: A C++ OOP lightweight vector and matrix library. This is just set of header files that have no external dependencies.

Gvars: A C++ configuration library that uses LibCVD and helps PTAM store constants and parameters and create a menu system.

Project plan:
The initial plan was to port PTAM to the iPhone and then to optimize it for speed and robustness once the code was compiling. Thereafter the approach and optmizations described in [8] would be applied to whatever extent that would enable useful application of the algorithm.

Porting:
This initial stage was expected to be trivial but in practice ended up being the major bottleneck for the project. PTAM is a powerful and complex piece of research software. The source code includes many comments and the overall action of the algorithm is described in [3]. With that said it is quite difficult to delve into the code itself and know what can and cannot be removed without breaking it. In order to explain the porting process and understand the difficulties it is necessary to briefly explain how PTAM operates.

PTAM runs in two threads: one for tracking and one for map building and optimization. The system initializes the map with a set of tracked image interest points and then in the Tracker class tracks points using various matching measures. Using point matches and map matches the tracker creates a motion model and a pose correction from refined matched points. The MapMaker runs in a low priority thread and is meant to be active whenever the Tracker is waiting for a new frame. The logic behind this is that the MapMaker does not have to be online all the time because it operates and optimizes on input that is not constant such as adding new keyframes and tracked points and global consistency.

Built directly alongside the core algorithm code (i.e. within the same functions) is the graphical display code including OpenGL calls and gvars menu commands. Neither the menu nor the graphics are necessary for PTAM to operate so the first priority in the porting process was to remove all of these features until PTAM was stripped down to its bare minimum which is essentially the ability to tell the user what her 6DOF position is in the world relative to the initial capturing position and surface.

The very first problem encountered before even starting the project was simply running PTAM on the desktop Mac Mini that was used for all coding. The main issue was that the webcam I had was not compatible with LibCVD's available video sources and so I eventually downloaded and compiled OpenCV[9] which is a widely used computer vision library with support for multiple devices and linked it into the project so that I could access the webcam through an OpenCV VideoSource that I found on the online PTAM blog [10].

The next stage was to calibrate the webcam using the

PTAM calibrator and then run PTAM and observe its behavior using a camera that has a very similar FOV to the iPhone 4 (42 degrees). PTAM operated satisfactorily, correctly tracking points and creating maps provided the camera motion was not too fast and the room was sufficiently "interesting" and not devoid of texture points.

Following this success it became immediately apparent that to do any testing of the code in a reasonable sense I would need to simulate the video input so videos of a simple chessboard scene were used from that point on for the stripping down process to verify that PTAM was still operating despite the ongoing surgery.

The process of stripping down was extremely time consuming. The first stage was to remove all the gvars code and simply hardcode the relevant variables. Once this was done there was no reliance on gvars whatsoever and the dependency was removed from the project.

The next step was to remove all unnecessary functionality from LibCVD so that it would be easier to compile it directly onto the iPhone. In order to do so I went through every file and wrote down the LibCVD functions in use and determined their inner dependencies on other LibCVD functions and external ones on other libraries. There was no simple way to do this and there were numerous occasions where I deleted PTAM functionality that I then couldn't find the correct place to recover it so I had to constantly rollback the code to previous versions and then redo what I already knew worked. In the end I removed all external dependencies for LibCVD and whittled the code base down to the bare minimum necessary files and functions.

For all the reduced iPhone compatible sources I changed their names to iLibCVD, iTooN and iPTAM and created a new XCode project. I used the recently opened Apple video API's to get video frames from the live camera. Then I connected everything together and recoded the basic grid drawing features to work with OpenGL ES for the iPhone.

To perform the calibration of iPTAM's camera parameters I recorded a video on the iPhone and used the calibration project provided with regular PTAM on the MacMini to extract the values and hardcoded them into the project. Once this was done the project was ready for compiling and testing.

Results:
A number of issues had to be overcome before the stereo initialization stage worked but it eventually correctly initialized a set of points from a planar surface and started the Tracker and MapMaker threads.

The results can be seen in the links [12][13][14].

The main issue as the port currently stands is that after initialization of the Map iPTAM does not detect any matched points with its tracked points and immediately goes into recovery mode as a result. I have no doubt that somewhere one of the multiple hardcoded constants in globals.h is incorrect or the corner matches are not good enough and so the matching is failing. However after extensive searching I was unable to find the culprit.

To rectify this and enable at least some form of tracking I disabled the recovery mode of the tracker and forced PTAM to continue tracking which essentially results in all the updates being done by the motion model in relation to detected point motion. In essence only the model predictive portion of PTAM is being used and no map building or true closed-loop tracking is taking place.

The current version tracks fairly large rotations, small translations and absolutely no scale changes. Moving the phone about quickly illustrates the problem of drift and scale issues.
With that said one can actually achieve a level of augmented reality and some additional interactive effects like being able to alter the trackers pose estimate by waving ones hand in front of the camera.

Conclusions:
In order to proceed with a full implementation of PTAM on the iPhone it is necessary to find the errors that are causing the keypoints not to be correctly matched. Furthermore the next logical step after that is to start implementing the optimizations described in [8].

Comments:
I have created an open github repository of the code [11] and have linked to it in various places. Ideally someone will choose to help in furthering work on this project.
I have also had a number of correspondences with George Klein to ask for his advice on implementation and have requested access to the MiniPTAM code under an academic license which I am waiting for a response to.

References:
[1] Android PTAM, YouTube: http://www.youtube.com/user/TheBoxil#p/a/u/0/drSEK8k3QJ0
[2] MiniPTAM, YouTube: http://www.youtube.com/watch?v=pBI5HwitBX4&feature=related
[3] Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proc Intl. Symposium on Mixed and Augmented Reality (ISMAR 2007), Nara (November 2007)
[4] PTAM code download: http://www.robots.ox.ac.uk/~gk/PTAM/download.html
[5] LibCVD: http://mi.eng.cam.ac.uk/~er258/cvd/
[6] TooN: http://mi.eng.cam.ac.uk/~er258/cvd/toon.html
[7] Gvars3: http://mi.eng.cam.ac.uk/~er258/cvd/gvars3.html
[8] Klein, G., Murray, D.: Parallel Tracking and Mapping on a Camera Phone In: Proc. Eigth {IEEE} and {ACM} International Symposium on Mixed and Augmented Reality (ISMAR'09), Orlando (October 2009)
[9] PTAM Blog: http://ewokrampage.wordpress.com/
[10] OpenCV: http://opencv.willowgarage.com/
[11] iPTAM github: https://github.com/twerdster/iPTAM

[12]http://www.youtube.com/watch?v=WgCsnYiseGY
[13]http://www.youtube.com/watch?v=_IOO6LN2nng
[14]http://www.youtube.com/watch?v=o-JrVIPRcjA