Imperial College
London

PROJECT INTERIM REPORT

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

ElasticFusion on the Google Tango Tablet

Author:
Jiahao Lin (CID: 00837321)

Date: January 25, 2017

1 Abstract

In recent years, Google has released an augmented reality Computing platform, Tango[1]. Supporting by with this platform, Google's Tango tablet is an Android device that equip with sensors that enable computer vision functionalities, Including NVIDIA Tegra K1 processor, 4MP color camera, fisheye-lens (motion-tracking) camera, Inertial Measurement Unit (IMU), and Infra-Red projector with RGB-IR camera for integrated depth sensing. Given this cheap commodity mobile hardware and software, a fully self-contained 3D dense surface reconstruction solution can be built by applying state-of-the-art RGB-D SLAM (Simultaneous Localization And Mapping) algorithms on the hardware. This means that high quality 3D room-scale scanning using portable device in real time would become possible.
In the recently published dense RGB-D SLAM algorithms, ElasticFusion[1] published by Dyson Robotics Lab[2], is one that achieve state-of-the-art performance by applying local and global loop closures using a deformation graph, therefore drifts raised by accumulated errors can be recovered and global consistency of the map is maintained. This is particularly useful in 3D surface reconstruction. Also, the current code exist for ElasticFusion made heavily use of GPU programming, in which CUDA is mainly used in the tracking process, which is supported by the NVIDIA Tegra K1 processor on Google tango tablet to enable fast calculation. The fact Elastic Fusion scored high marks on benchmarks for trajectory estimation and surface estimation makes it perfect to be applied on Google Tango Tablet.

2 Introduction (1-3 pages)

2.1 Project Objectives

The objectives of this project is to develop a fully self-contained mobile dense 3D surface reconstruction solution running on Google Tango Tablet. The 3D colored depth map should be rendered on screen in real time from the tablet's viewpoint, and after the 3D scanning stopped the 3D map created should be able saved on device to allow for later use or shown on other devices. The main point of this solution is to produce high quality reconstructed surface map that avoids misalignment, which is often caused by loopy motion of the scanning device during the process. When doing 3D reconstruction, simple fusion of 3D point clouds would produce drifts caused by accumulating errors during the process, therefore by applying ElasticFusion algorithm to the solution, even extremely loopy trajectories of the tablet and long duration of scanning process will not decrease accuracy of the map. On the contrary, revisit areas that has previously been mapped will only help maintain the global consistency of the map by ElasticFusion Algorithm. The part of algorithm that made this possible is the local and global loop closure checking after camera tracking process, which will be discussed in detail later. In our solution, we intend to use Android Native

[1] `https://get.google.com/tango/`
[2] `http://www.imperial.ac.uk/dyson-robotics-lab/`

Development Kit (NDK) to allow the use of Tango C API, which makes it possible to reuse some of the existing code for ElasticFusion. If the development of the solution is successful, an augmented reality application can be created as demo to illustrate the interaction of virtual object with scanned 3D surface map.

3 Background (10- 20 pages)

3.1 Related Work (3 pages)

3.1.1 SLAM basic

The paper: past present and future of SLAM [2] gave a survey about the current development state of modern SLAM, including standard formulation and models, different methods used in SLAM. Below we give a brief summary of basic knowledge about SLAM from this paper.
Simultaneous Localization and Mapping (SLAM) means constructing the model of environment (map) that the robot is in, and estimating the robot state within the environment at the same time. Usually the robot is equipped with some kind of sensor: RGB camera, depth sensor, IMU, or GPS, So that the robot is able to perceive the environment in some way from these sensor. In standard models, the robot state includes its position and orientation, velocity, sensor biases and calibration parameter. Using the robot status and data read from sensors, the environment(map) constructed could be representation in different forms. With the existence of the map, errors raised in estimating robot state could be eliminated and also doing ”loop closure” when robot revisits a place in the map allows drift to be corrected and hence improve accuracy of the estimated status of robot.

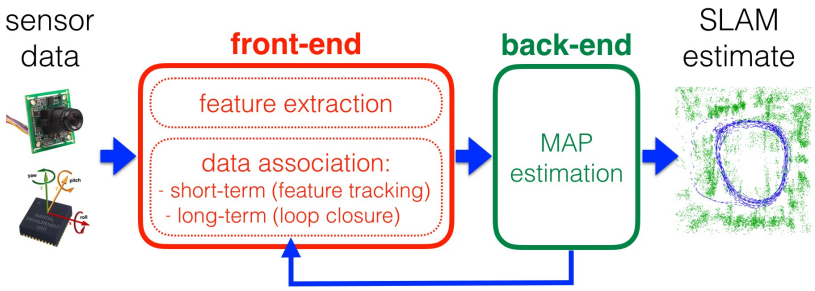

Figure 1: Front-end and back-end in a typical SLAM system[2]

The structure of SLAM system is mainly divided into two parts: front-end and back-end The front-end extract features from sensor data and associate them to the model used to make predictions, then the back-end does estimation of robot state and the map using this model produced by front-end. As you can see in the figure 1, this process forms a continuous process of tracking and update.
Classical formulations of SlAM is to take probabilistic approaches such as models based on Maximum Likelihood Estimation, Maximum-a-posteriori (MAP) estimation,

EKF(Extended Kalman Filters), and particle filters.
Many of the popular SLAM algorithm models it as a maximum-a-posteriori estimation problem by taking probability approach. And usually using a factor graph to show relationships and constraints between variables. In these models, often an observation or measurement model is used to represent the probability of observe measurements given the state of robot, given some random measurement noise that usually model to be under a zero-mean Gaussian distribution. By using Bayes theorem, the MAP estimate, in which the posterior is the probability of robot state given a measurement, can be calculated given some prior belief of the robot state. In the case that there is no prior information known about robot state, the MAP estimate would simply become a maximum-likelihood-estimate. When we are given a set of independent robot states and measurements, these variable would become factors(nodes) in the factor graph, and their probabilistic relationships would become the constraints(edges) in the factor graph.
Factor graph allows us to visualize our problem in a simple way and provides an insight into how the variables are related together by their constraints. 2 from the paper have shown an example of it for a simple SLAM problem.

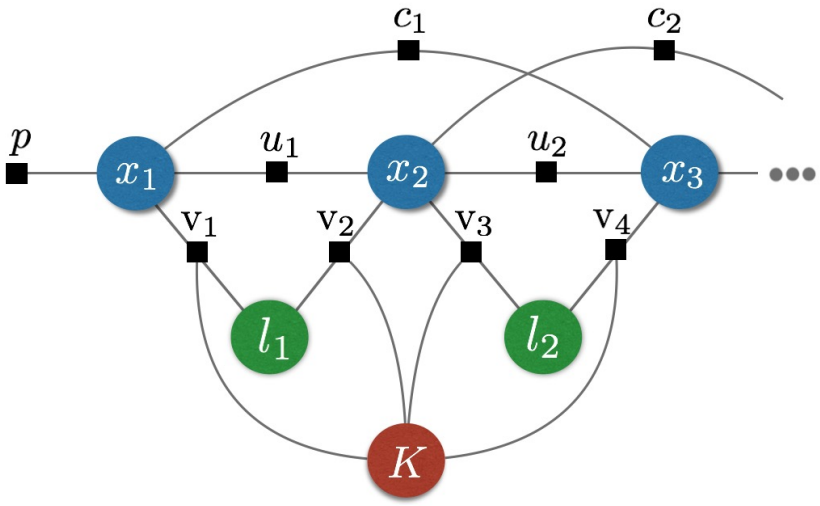

Figure 2: SLAM as a factor graph[2]

As explained in [2]: "Blue circles denote robot poses at consecutive time steps ($x1$, $x2$ ...), green circles denote landmark positions ($l1$, $l2$ ...), red circle denotes the variable associated with the intrinsic calibration parameters (K). Factors are shows are black dots: the label "u"marks factors corresponding to odometry constraints, "v "marks factors corresponding to camera observations, "c"denotes loop closures, and "p"denotes prior factors."

When calculating the MAP estimate of robot state, we can transform the problem into minimizing the negative log of posterior, assuming the noise is zero-mean Gaussian distributed, this becomes a non-linear least squares problem, since we will be minimising the measurement error's $l2$ norm. This method is similar to the BA (Bun-

dle Adjustment) method in computer vision area, which minimise the reprojection error. However factor graph can not only contain visual and geometry factor, but also a variety of data from different sensors. Also the MAP estimate is calculated every time a measurement is arrived, instead of performing calculation after all the data is given.
Different from factor graph optimization, a similar approach is called pose graph optimization, which estimate trajectory of robot using measurements between relative poses as constraints, and optimized these poses to achieve better accuracy trajectory.
Other than optimization-based estimation, some filtering approach using EKF model have also achieve great results, such as MSCKF(Multi-State Constraint Kalman Filter of Mourikis and Roumeliotis)[3], EKF is used for performing filtering on the linearised point of estimate, and have shown to be accurate in visual-inertial navigation.
In terms of the representation of the map, one way is to have a group of features or landmarks in 3D, these features are usually extracted from sensor images to be allow easy distinguishable from environment, such as edges and corners. The method used to extracting features from images are well-established within computer vision area, using descriptors like ROB, SIFT, and SURF. The discriminative and repetitive properties of landmarks allow the correspondence of each landmark and sensor measurement(image), and then via triangulating, the robot is able to compute the relative pose between measurements and geometric information about landmarks, therefore achieve localization and mapping.

Other than feature based approach, sometimes raw dense representation of the environment is used. different than extracting landmarks, this approach stores a dense 3D map using high resolution points, the 3D geometry of environment is described by these vast amount of points, so called point clouds. With stereo or depth cameras, lase r depth sensors, 3D information of the points can be easily obtained, therefore this method is popular in RGB-D SLAM. Sometimes, these points doesn't only stores simple 3D information, in Elasticfusion[1], each 3D point is represented by a surfel, which is a disk with certain radius centred by the 3D point and stored information such as normal vector of that point on the surface and color. These suefel disk combine together can encode the geometry of environment better than simple points.
At more a higher level than raw, unstructured points, dense related element that partitioned by space is also used to represent surface and boundaries, such as surface mesh models(connected polygons formed by points) and implicit surface representations like octree and occupancy grid. KinectFusion[4] storing surfaces using volumetric representation by uniformly subdivide a 3D physical space into a 3D grid of fixed size voxels, voxels defining surface has a zeros value of the truncated signed-distance function (TSDF) function that stores distance information relative to the surface. We can see that represent the environment in these types of dense data requires huge amount of storage, however they give very low level geometry information which is suitable for 3D reconstruction and rendering.

Comparing feature-based and dense SLAM methods, for the purpose of this project,

we will use dense SLAM algorithm such as Elasticfusion[1] for 3D reconstruction, since it maintains a dense 3D surfel map that make use of all the information about the environment from sensor thus more robust for localization and mapping.

3.1.2 Kinectfusion

KinectFusion[4] is a dense RGB-D SLAM method that enables high quality 3D reconstruction and interaction in real time. This method make use of mobile low cost RGB-D camera, scanning the environment in real time and acquire 3D point information about the space. During the reconstructing process, holes caused by absence of depth measurement in original 3D point cloud is filled, and model is more refined by repetitive scanning over time.
Firstly the 3D points reading are then stored in a 3D vertex map and their normal vectors are calculated by reprojection of neighbouring vertexes. After that the camera pose is tracked by using Iterative Closest Point (ICP) algorithm to align current 3D points with the previous frame. In ICP algorithm, first the corresponding point of each current 3D point is found by doing projective data association, which reprojects each previous point into image coordinates and use this to look up current vertexes that may be correspondence points. In the second part these correspondences are tested by calculated distance and angle between them and reject any outliers that are beyond certain threshold. Please be noted that the running of ICP in real time is because the use of GPU to parallel processing a current vertex per thread, according to the original paper.

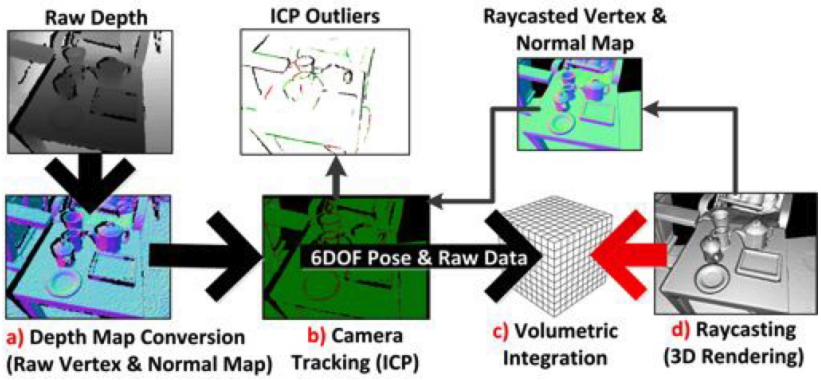

Figure 3: Overview of tracking and reconstruction pipeline from raw depth map to rendered view of 3D scenem[4]

The structure of KinecFusion is given in Fig 3 from original paper. The result of ICP is a transformation matrix from previous camera pose to the current pose, which is later used for converting the coordinates of current vertexes into the global coordinate frame. These coordinate data is then fused into the current model that uses a volumetric representation, as mentioned before. The 3D space is uniform divided into fixed size voxel grid. The geometry of surface is implicitly stores in these voxels by having zero values of Truncated Signed Distance Functions (TSDFs) at the

surface, positive value in the front of surface, and negative at the back of it. Also, to improve efficiency and minimise storage usage only a truncated area around the surface is actually stored, thus the name ”truncated”.
When integrating the current vertexes into voxel grid, each voxel is perspective projected back onto the image to calculate the difference between it and the measurement depth of this coordinate. The distance difference is then normalized to maximum TSDF distance and updated using weighted average with previous value. Similar to the implementation of tracking stage, the integration and update process also make use of GPU parallel processing, each thread is assigned a slice of of grid along the $Z-axis$, and sweep through each voxel by its (x,y) position for processing, as shown in Fig 4.

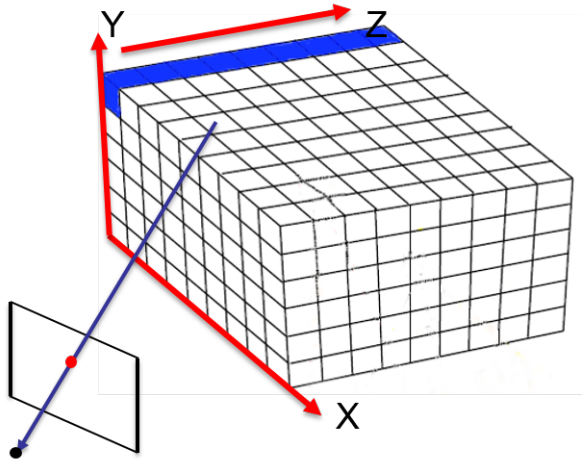

Figure 4: Volumetric Integration using GPU thread [5]

Finally, in order to render the 3D reconstructed surface raycasting is performed. This is done by having each GPU thread to follow a ray from the camera center along one pixel in the image into the voxel grid in camera coordinate space, the ray is extended until a zero-crossing voxel is met, this voxel represents the surface point observed by this pixel and the position is extracted by doing a trilinear interpolation on this voxel point. And the normal at this position is calculated by find the surface gradent of TSDF values around the position. With these information for each pixel, a live 3D reconstruction view from camera view point can be rendered using camera's pose and intrinsic information.

3.1.3 Keyframe-Based Visual-Inertial SLAM Using Nonlinear Optimization

Other than using depth sensor with RGB camera, another kind of sensor equipped on the Google Tango table is Inertial Measurement Unit (IMU). Okvis[6] is a sparse feature-based SLAM approach that tightly coupled IMU measurement with stereo visual measurement using nonlinear optimization.
Usually in optimization-based visual SLAM, the structure of geometry is to relate camera poses using landmarks, and optimization is performed to minimize the error

of landmark reprojection in different observing frames. However, in visual inertial SLAM, the IMU measurement is used to add additional constraint between camera poses and speed and estimation biases from gyroscopes and accelerometers. The structure graph from original paper is shown in Fig 3 with detail labelling of different variables and constraints.

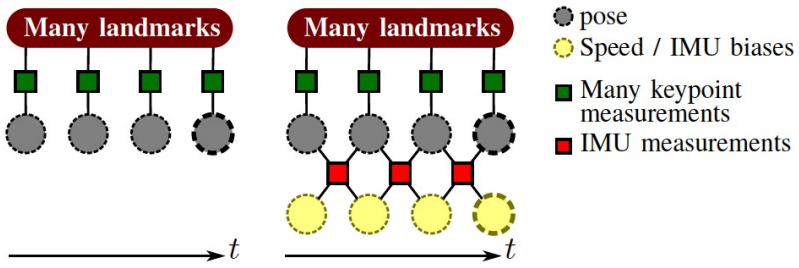

Figure 5: graph of variables and their relationships in visual SLAM on the left and visual intertial SLAM on the right [6]

In order to tightly couple the visual and inertial measurements, a joint nonlinear cost function that contains both visual reprojection error term and IMU error term is created. These both errors are eliminated by optimizing this one error function.
To reduce complexity and ensuring real time operation, a fixed size temporal window of robots poses and relative states is maintained, and old robot poses will be marginalized out as new frames are inserted.
In the visual frontend, a sparse map of frames and landmarks is stored. For each new frame, keypoints are extracted using SSE(Streaming SIMD Extensions)-optimized Harris corner detector and BRISK(Binary Robust Invariant Scalable Keypoints) descriptor, their 3D positions are then triangulated by stereo and put into the sparse map. During the process landmarks and keypoints are brute-forcely matched and outliers are rejected by using predicted pose from IMU data. In the optimization window, two kind of frames is maintained, first is a temporal window of the S most recent frames, second is N keyframes in the past. A frame is determined as keyframe if the number of matched points with previous frame is less than 50 to 60% of the number of detected keypoints.

In the marginalization process, initially the first $N + 1$ frames forms the marginalization window, as shown in Fig 6. When a new frame is inserted into the marginalization window, there are two ways to marginalize old state. If the oldest frame in temporal window is not a keyframe, then the state of that frame with its measurements, speed and biases will be dropped, as shown in Fig 7. On the other hand if it is a keyframe, landmarks that are visible in first temporal frame but in most recent keyframe is also dropped, as shown in Fig 8.

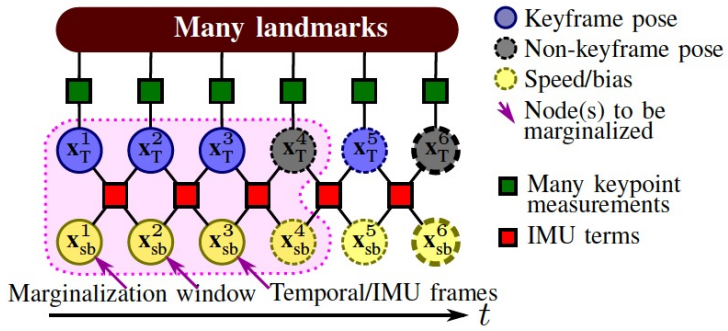

Figure 6: Graph of initial marginalization window [6]

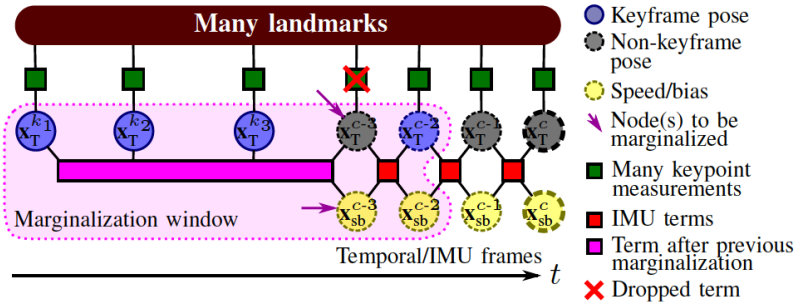

Figure 7: Graph for marginalization in the first case [6]

3.1.4 Elasticfusion

Elasticfusion[1] is a RGB-D dense SLAM method for 3D reconstruction in room scale without using pose graph , it also use surfels to represent the map instead of occupancy grid. In the scanning process, it iteratively check for local and global loop closures and refine the map by using a deformation graph to apply non rigid deformation to the surface.

When tracking camera poses, it combine calculating poses using ICP algorithm on point cloud with matching intensities of pixels from RGB image. When manipulating point cloud, OpenGL shading language is used to update, fusion and render the view. However, while this approach has achieved great results in room scale, it may be difficult to use it on bigger or outdoor scenes.First, when tracking camera poses, using dense frame-to model camera tracking,photometric and geometric frame-to-model predictive tracking, geometric and photometric pose estimations are combined into a joint cost function to be minimised. geometric error is calculated using frame to model projective data association and ICP algorithm as in kinecFusion[4], and photometric error is gain by the intensity difference between current RGB image and backprojected image. segment older parts of the map which have not been observed in a period of time t into the inactive area of the model. The surfel map used is similar to what is used in [7], each surfel stores information about local surface area around the center to reduce holes on the surface. The information includes position,

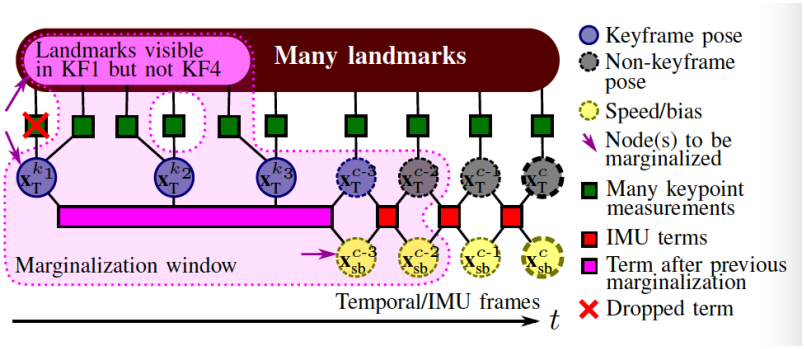

Figure 8: Graph for marginalization in the second case [6]

normal, color and radius. The radius is larger as distance between image center and surfel is longer.

Deformation graph is constructed for each frame for efficiency, nodes are sampled uniformly from the surfels and each node have a rotation matrix and position vector that represented optimization parameter. The neighbourhood of each node forms the edge of them.The optimization is done by have a deformation graph at each frame that is the same as in DynamicFusion[8], the graph is used to perform non-rigid surface deformation to the surfel map which will optimize the model. the set of nodes which influence each surfel Ms must be determined.

Also local and global loop detection is used to recover from drifts and realign the camera pose with the map. If the estimated camera post error is above a threshold it means drift is too far, then a global loop closure is trigged to try to realign camera pose with past frames. The past frame information is stored in a randomized fern encoding database [9], and if a high quality matched is found, points are sampled from the image to build constraints that are used to optimize the deformation graph. Which in turn is used to perfrom non-rigid deformation on the surrface.

If no global loop closure is found then local loop closure if detected,local model-to-model surface loop closure optimisations, by try to register the part of inactive model with active model within the same frame, register the portion of the active model within the current estimated camera frame with the portion of the inactive model underlaid within the same frame., when hight quality alignment is found similar method is used to optimize the deformation graph, The inactive portion of the map which caused this loop closure is then reactivated.

The surfel map for current fram after deformation is then fused into global map using OpenGL.

4 Project Plan (1-2 pages)

The plan of this project is to start with simple sample project on Google Tango Tablet, using the Tango C API, based on exist C++ code for Okvis[6], to iteratively build out the solution step by step, so that milestones are available if a certain stage is unable to complete before the project deadline.
In this project the goal is run ElasticFusion[1] on Google Tango Tablet for real time 3D reconstruction, but the algorithm will be adapted to fit into the Tango platform, also to make use of the Tango motion tracking and depth perception API. Given that Tango's camera pose estimation already make use of RGB-D and IMU sensor data, the pose tracking could possibly be replaced with 3D pose gain directly from Tango. Also, camera frame and 3D point cloud data for a time point can be acquired with the API. However, although the Tango 3D Reconstruction Library C provides functions and structs reconstruct 3D suface using mesh or voxel grid, it is different than the surfel map that we will be using for ElasticFusion[1]. However, it will be worth to study the sample project of building mesh and rendering on screen.
First, we need to set up a simple demo Google Tango Android application for motion tacking that output real time pose data into the logs.
Then camera frames needed to be rendered on screen in real time with buttons to resume and pause the rendering.
Next step is to make use of the mesh builder in Tango 3D Reconstruction Library for fusing point clouds and use OpenGL to render it on screen from the camera view point.
After that we can build out the structs and functions for surfel map and global map representation used in ElasticFusion[1], in which the surfels are divided into active and inactive types according the time it is last observed.
With the surfel map classes, integration with Tango could be done by try to extract surfels from point cloud data and then render the surfels into the screen.
Then the structs and functions for deformation graph, its nodes and surface constraints will be built to allow for non rigid deformation of the surface.
In order to perform the optimization using surface constraints, global and local loop closure needs to be done. For global loop closure, ElasticFusion[1] uses randomised fern encoding database to store frame information and check for match frames. Therefore fern and frame structs and this store, lookup mechanism needs to be implemented.
With global loop closure in place, this can be tested out by manually triggering the loop closing and see whether deformation graph is applied, which will cause the surfaces to be realigned, and camera pose be updated.
If this is successful, local loop closure can also be done using model-to-model tracking by trying to register the active and inactive point clouds together under the current frame view point. This is done by using the same tracking and optimization process in global loop sure.
It will be the ideal case if the above is all achieved before the project deadline, since this is a very challenging project. And if at a certain stage it is realised that it could not be done before the deadline, the progress can be fall back to the previous milestone and start clean up and report write up.

However, extension and optimization could be done if more time is given and basic solution is completed. Until here the 3D reconstruction process will be running on Tango tablet's NVIDIA Tegra K1 processor without utilizing the CUDA feature on it to improve performance by parallel processing. A optimization could be to use CUDA in the camera tracking process. Also on the application side, extra features can be added such as saving the reconstructed surfel map on the Tango tablet, add an interactive view of reconstructed 3D surface map on the screen to allow for drag and zoom operation, showing the camera's current pose on the map. Further more, demo application and be built to utilising the augmented reality ability on Tango platform, such as placing a virtual static object on the 3D reconstructing surface.
The project timeline is shown in Fig 9 with each task corresponding to a stage above.

5 Evaluation plan (1-2 pages)

–

Project evaluation is very important, so it's important to think now about how you plan to measure success. For example, what functionality do you need to demonstrate? What experiments to you need to undertake and what outcome(s) would constitute success? What benchmarks should you use? How has your project extended the state of the art? How do you measure qualitative aspects, such as ease of use? These are the sort of questions that your project evaluation should address; this section should outline your plan.

References

[1] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. Elasticfusion: Dense slam without a pose graph. *Proc. Robotics: Science and Systems, Rome, Italy*, 2015. pages 2, 5, 6, 9, 11

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016. pages 3, 4

[3] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Robotics and automation, 2007 IEEE international conference on*, pages 3565–3572. IEEE, 2007. pages 5

[4] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011. pages 5, 6, 9

[5] Fuxing Yin. *Volumetric Representation on KinecFusion*. 2016. pages 7

[6] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015. pages 7, 8, 9, 10, 11

[7] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *3DTV-Conference, 2013 International Conference on*, pages 1–8. IEEE, 2013. pages 9

[8] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of*

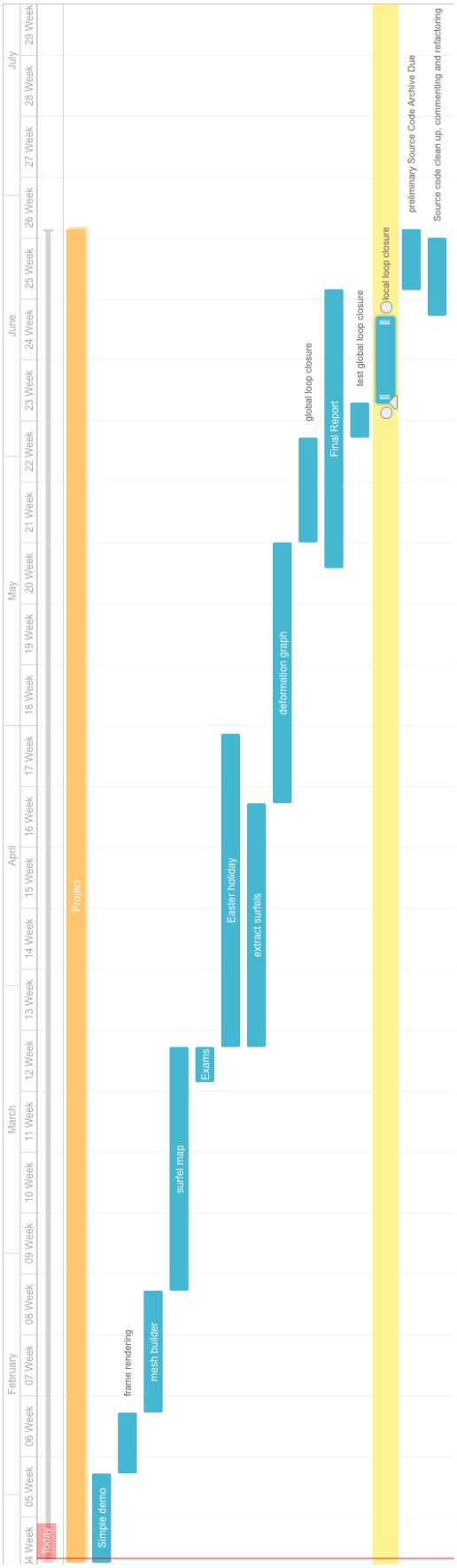

Figure 9: Graph for project timeline

the IEEE conference on computer vision and pattern recognition, pages 343–352, 2015. pages 10

[9] Ben Glocker, Jamie Shotton, Antonio Criminisi, and Shahram Izadi. Real-time rgb-d camera relocalization via randomized ferns for keyframe encoding. *IEEE transactions on visualization and computer graphics*, 21(5):571–583, 2015. pages 10