KO-Fusion: Dense Visual SLAM with Tightly-Coupled Kinematic and Odometric Tracking

Charlie Houseago, Michael Bloesch, Stefan Leutenegger
Dyson Robotics Lab at Imperial College, Department of Computing, Imperial College London
email: c.houseago16@imperial.ac.uk

***Abstract*— Dense visual SLAM methods are able to estimate the 3D structure of an environment and locate the observer within them. They estimate the motion of a camera by matching visual information between consecutive frames, and are thus prone to failure under extreme motion conditions or when observing texture-poor regions. The integration of additional sensor modalities has shown great promise in improving the robustness and accuracy of such SLAM systems. In contrast to the popular use of inertial measurements, which may cause strong global drift due to noise and biases, we propose to tightly-couple a dense RGB-D SLAM system with kinematic and odometry measurements from a wheeled robot equipped with a manipulator. The system has real-time capability while running on GPU. It optimizes the camera pose by considering the geometric alignment of the map as well as kinematic and odometric data from the robot. Through experimentation in the real-world, we show that the system is more robust to challenging trajectories featuring fast and loopy motion than the equivalent system without the additional kinematic and odometric knowledge, whilst retaining comparable performance to the equivalent RGB-D only system on easy trajectories.**

I. INTRODUCTION

The integration of Simultaneous Localization and Mapping (SLAM) algorithms into real-world agents is a significant challenge in the future of mobile robotics. Candidate algorithms can be broadly categorized into sparse feature-based systems and dense or semi-dense systems. Sparse systems have numerous computational advantages, and display state-of-the-art performance in pose estimation and robustness [1] [2]. Success has been found in further improving the performance of sparse SLAM methods by tightly coupling the visual data to an inertial measurement unit [3], [4], [5].

In contrast, Dense SLAM methods provide a more effective means to understand the 3D structure of an environment. Mapping is achieved by merging incoming measurements into a representation of its constituent surfaces. Dense maps offer much greater potential for problem solving and safe navigation for robotics, and a robust Dense SLAM system running aboard a mobile robot is an essential precursor to the idea of a general domestic robotic platform. In Dense SLAM, the environment is mapped using a detailed geometric model, against which the position of an observing camera or other sensor is tracked. Whilst early methods used 2D occupancy maps [6] due to computational constraints, the widespread presence of GPU computing and depth sensing technology has led to algorithms which maintain full 3D maps with high-dimensional state spaces. One of the first of these was DTAM: Dense Tracking and Mapping [7], which demonstrated the additional usefulness of a dense map for real-time scene interaction. DTAM is however limited to small scenes and quickly loses tracking under fast motion. One of the most influential works using GPU computing and a depth sensing camera is Kinect Fusion [8], which uses a Truncated Signed Distance Field (TSDF) [9] to construct a map, and the Iterative Closest Point algorithm for alignment [10]. With real-time tracking Kinect Fusion is capable of producing photorealistic reconstructions, but has large memory requirements for mapping. Kinect Fusion forms the basis for the bulk of modern SLAM systems, with some approaches attempting to increase the efficiency of the map storage [11], [12], or to correct for misalignment and drift by deforming the underlying structure of the map representation as in ElasticFusion [13]. To enable this deformation, ElasticFusion makes use of an unordered list of surfels [14] as its map representation, constructing visually consistent maps which do not comprise of continuously connected surfaces.

Fig. 1: Tightly integrating pose predictions from the on-board kinematics of a robot arm into RGB-D dense SLAM improves tracking and robustness on challenging trajectories. The above surface reconstruction was captured in challenging conditions with a fast moving robot under conditions where state-of-the-art dense SLAM methods lose tracking.

There are several examples of systems which aim to improve tracking robustness in Dense SLAM by including acceleration and rotation rate measurements from an Inertial Measurement Unit (IMU). These systems can be categorised by the methods which they use to integrate the IMU data, with tightly-coupled systems which incorporate data directly into the tracking and mapping optimisation and loosely-coupled approaches which determine an inertial pose which

is then used to refine the predicted pose from camera tracking. The work in [15] is loosely-coupled and uses an IMU-driven Extended Kalman Filter to aid estimating the transformation between image pairs. In [16] a pose-graph like approach is used, where poses predicted both from the inertial measurements and stereo camera measurements are fused together.

A number of systems tightly-couple inertial data into a semi-dense SLAM system. For example the work in [17] presents a monocular visual-inertial system with semi-dense tracking and uses a planar prior to build a dense map below frame rate. The tightly-coupled system in [5] is based on LSD-SLAM [18] and outperforms vision only systems and loosely-coupled approaches.

One of the first tightly coupled fully Dense Visual-Inertial SLAM system is the work of Laidlow *et al.* [19] which performed tight-coupling of inertial data into ElasticFusion [13]. They extend the frame-to-frame tracking by constructing a combined geometric, photometric and inertial alignment error, which is then minimised. Approaches such as this based on IMU data suffer from increased drift and divergence which can be due to the dependence on accelerometer measurements and the need for IMU bias estimation. A particularly interesting approach to reducing the drift in Inertial Methods is found in GravityFusion [20]. They extend ElasticFusion [13] by attaching a gravity measurement to each surfel, and then use mesh deformation to enforce this gravity direction across all the surfels in the map, effectively removing two degrees of freedom from the drift in the map.

Whilst a vision-only or visual-inertial system is applicable across more morphologies, a mobile robotics platform almost always has odometry available, and commonly manipulator kinematics for systems designed for environmental interaction. It is therefore convenient to exploit these in a tightly-coupled manner to improve navigation. Therefore we advocate the addition of a direct kinematic error dependent on the forward kinematics of its manipulator pose and a relative odometric pose error. Much like a visual-inertial system, a system which combines predictions from a robot manipulator and odometric base similarly affords a robustness improvement on challenging trajectories.

The inclusion of an error term from absolute pose actuators on a robotic agent in our work is closely related to that of Klingensmith *et al.* [21]. Like ARM-SLAM, we treat the forward kinematics as a strong pose measurement between base and camera while optimising a cost function at each frame to localize the camera. However, their system features a volumetric map and sparse, feature-based tracking, as opposed to the dense map and full geometric tracking employed in the work at hand. Additionally, our inclusion of an odometric error allows us to move the mobile base.

Another closely related work is [22] which extends ElasticFusion [13] with kinematic and inertial data from a humanoid robot. They demonstrate that the addition of this information allows their system to be robust to a variety of challenging situations such as low visual information, fast motion and continuous scene dynamics. Their system operates in a semi-dense manner on stereo camera images at 12Hz and sends pose corrections to the kinematic-inertial tracking. Our system operates at a higher framerate and with more rapid and complex rotations, due to the mounting of the camera at the end of a manipulator rather than on the head of a humanoid robot.

Our use of vision to improve the localisation of an omnidirectional vehicle is related to a number of previous works [23] [24] [17], but to our knowledge, ours is the only system which implements a full Dense SLAM system on a mobile agent with omnidirectional motion capability where the transform between the camera frame and the agent body frame is allowed to change.

In summary, we extend the RGB-D SLAM system ElasticFusion [13] by incorporating tightly coupled kinematic information from the robot actuators and odometric information from the moving base. This system is capable of real-time tracking and dense mapping running on a GPU at 30Hz. It can run on any platform which is capable of providing incremental 6-DOF pose measurements of its base as well as relative pose measurements between base and camera. An example map output is shown in figure 1. We demonstrate greater robustness in the face of challenging loopy trajectories, and handling of large surfaces of similar texture by maintaining a consistent trajectory and globally consistent map in cases where visual tracking alone cannot be relied upon.

II. Preliminaries

A. Notation

The following notation conventions will be used throughout this report:

- A reference frame A will be denoted as $\boldsymbol{\mathcal{F}}_A$
- The translation vector from the origin of $\boldsymbol{\mathcal{F}}_B$ to the origin of $\boldsymbol{\mathcal{F}}_C$ as expressed in $\boldsymbol{\mathcal{F}}_A$ is written ${}_A\boldsymbol{r}_{BC}$.
- The transformation matrix $\boldsymbol{T}_{AB} \in \mathbb{R}^{4\times4}$ is used to represent the transformation of homogeneous coordinates of a point in $\boldsymbol{\mathcal{F}}_B$ to $\boldsymbol{\mathcal{F}}_A$. The $SO(3)$ rotation corresponding with the transformation $\boldsymbol{T}_{AB}$ can be represented as a rotation matrix, $\boldsymbol{C}_{AB}$. Therefore, the transformation can be composed as:

$$\boldsymbol{T}_{AB} = \begin{bmatrix} \boldsymbol{C}_{AB} & {}_A\boldsymbol{r}_{AB} \\ \boldsymbol{0}^{\boldsymbol{T}} & 1 \end{bmatrix} \in SE(3).$$

- The rotation of $\boldsymbol{\mathcal{F}}_B$ with respect to $\boldsymbol{\mathcal{F}}_A$ can also be expressed as a quaternion, $\boldsymbol{q}_{AB}$, with a left-hand side operator $[]^+$, and a right-hand side operator, $[]^\oplus$.
- A tilde above a variable, such as $\tilde{u}$, indicates that the value was obtained from one or more sensor measurements.

B. Hardware and Calibration

Our system is deployable on any platform which can provide a 6-DOF forward kinematics estimate and 6-DOF incremental odometry updates. The real-world research platform used in this work is a KUKA Robotics YouBot [25]. It consists of a 5-DOF robot arm which we have fitted with

an ASUS Xtion Pro camera mounted on the final link close to the end effector.

The arm is mounted on an omnidirectional base, allowing the entirety of the robot to move in an arbitrary direction on a surface of fairly constant elevation. A full description of computing the predicted odometric pose is omitted here as it is fairly involved. A good analysis of the motions of the Mecanum wheels used is given in [26].

To calibrate the camera extrinsic transform with respect to the robot's end effector $\boldsymbol{T}_{CE}$, we developed a custom calibration package based on detecting AprilTags [27]. A standard bundle adjustment algorithm implemented in Google Ceres was used to minimise the re-projection error of detected corner points by jointly optimising the camera extrinsics and the relative poses of the tags in the world frame.

III. Approach Overview

The core architecture of our system is based on that described by Whelan *et al.* [13]. Their approach alternates between tracking the pose of an RGB-D camera against a dense 3D map of an environment, and refining this map with new observations. In this work, we refine the tracking module with the addition of error terms based on measurements from the actuators of a real-world robot with a moving base and a manipulator with 5 degrees of freedom.

Specifically, in the approach described in [13] the tracking step consists of a combined photometric and geometric cost which is minimized at each frame to obtain an estimation of the camera pose. Our system replaces this with a joint optimisation over the pose of the robot base and the held camera. We remove the photometric cost and impose additional constraints based on the kinematic and odometric information obtained from the robot. The odometric constraint relates to both the current and previous base pose, so we build a sliding-window optimisation problem and solve at each iteration for the current and previous state. Figure 2 shows the state graph which is solved at each tracking state. The linear prior is obtained by marginalisation of previous states and related error terms.

We detect loop closures in the same manner as [13] and transform the linear prior and previous state to maintain a consistent world frame between the state graph and the map, whilst smoothly deforming the map to maintain geometric consistency.

IV. Tracking

At the arrival of each new camera frame, we estimate the current state $\boldsymbol{x}_i$ by refining the prediction obtained from odometry and forward kinematics. The system state is comprised of the camera pose in the world frame with translation ${}_W\boldsymbol{r}_{WC}$ and orientation $\boldsymbol{q}_{WC}$, as well as the base pose in the world frame comprised of translation ${}_W\boldsymbol{r}_{WB}$ and orientation $\boldsymbol{q}_{WB}$:

$$\boldsymbol{x}_i = ({}_W\boldsymbol{r}_{WB_i},\ \boldsymbol{q}_{WB_i},\ {}_W\boldsymbol{r}_{WC_i},\ \boldsymbol{q}_{WC_i}) \in \mathbb{R}^6 \times SO(3)^2, \tag{1}$$

where index $i \in \{0, 1\}$ indicates previous or current state.

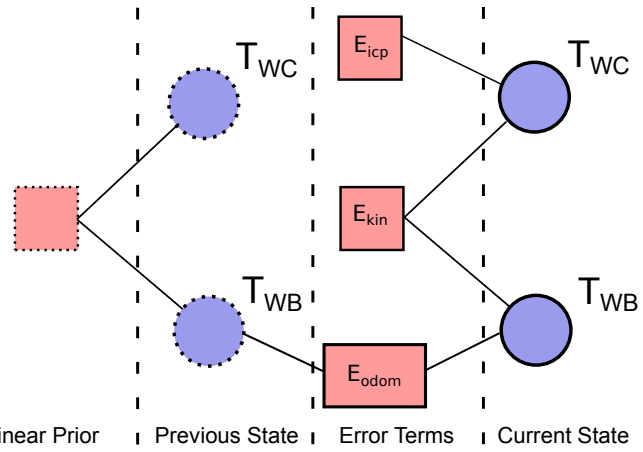

Fig. 2: Our system jointly optimises the current and previous robot state using error terms determined from the current frame and a linear prior obtained from marginalisation of states and errors from previous time iterations.

System states exist on a 12 dimensional manifold, and state refinement is performed by a local perturbation $\delta\boldsymbol{x}$ in the tangent space using the $\boxplus$ operator, such that $\boldsymbol{x} = \overline{\boldsymbol{x}} \boxplus \delta\boldsymbol{x}$. This operation is equivalent to standard vector operation for the translations. For orientation a combination of quaternion multiplication $\otimes$ and the exponential map is used: $\boldsymbol{q} \boxplus \delta\boldsymbol{\alpha} = \exp(\delta\boldsymbol{\alpha}) \otimes \boldsymbol{q}$. This results in the following minimal local coordinate representation:

$$\delta\boldsymbol{x}_i = [\delta\boldsymbol{r}_{WB_i}^T\ \delta\boldsymbol{\alpha}_{WB_i}^T\ \delta\boldsymbol{r}_{WC_i}^T\ \delta\boldsymbol{\alpha}_{WC_i}^T]^T \in \mathbb{R}^{12}. \tag{2}$$

A $\boxminus$ operator can also be defined to compute the difference between two system states. Similar to the update operator, this corresponds to standard subtraction for regular vector quantities. For the orientation, the form of this operator can be constructed by finding the inverse of the $\boxplus$ operator: $\boldsymbol{p} \boxminus \boldsymbol{q} = \exp^{-1}(\boldsymbol{p} \otimes \boldsymbol{q}^{-1})$.

Please refer to [28] or [29] for further details.

At each iteration, we compute a minimal state update to the combined current and previous camera pose by minimising a combined cost c_{track} computed from a linear prior, geometric tracking, and the robot's kinematics and odometry. This cost has the following form:

$$\begin{aligned} c_{\text{track}} = &\frac{1}{2}\boldsymbol{e}_{\text{prior}}^T\boldsymbol{e}_{\text{prior}} + \frac{1}{2}\sum_k e_{\text{icp},k}^T W_{\text{icp},k} e_{\text{icp},k} + \\ &\frac{1}{2}\boldsymbol{e}_{\text{kin}}^T\boldsymbol{W}_{\text{kin}}\boldsymbol{e}_{\text{kin}} + \frac{1}{2}\boldsymbol{e}_{\text{odom}}^T\boldsymbol{W}_{\text{odom}}\boldsymbol{e}_{\text{odom}}. \end{aligned} \tag{3}$$

A. State Propagation

To promote convergence of the tracking optimisation, rather than following the usual dense tracking convention of initializing with the previous camera state, we initialise the current state using the robot kinematics and odometry. We use the relative transform, measured by odometry $\tilde{\boldsymbol{T}}_{B_1B_0}$ between the robots previous and current base frame, to initialise the base pose as $\bar{\boldsymbol{T}}_{WB_1} = \bar{\boldsymbol{T}}_{WB_0}\tilde{\boldsymbol{T}}_{B_1B_0}^{-1}$. And similarly, initialise the arm pose as $\bar{\boldsymbol{T}}_{WC_1} = \bar{\boldsymbol{T}}_{WB_1}\tilde{\boldsymbol{T}}_{B_1C_1}$ with the measured forward kinematics $\tilde{\boldsymbol{T}}_{B_1C_1}$.

B. Geometric Alignment

The geometric alignment error is based on the point-to-plane iterative closest point (ICP) algorithm. We aim to find the state update that minimises the point-to-plane error between 3D back-projected points as viewed from the current camera pose, and corresponding points in the global model. The cost function for a single depth measurement k is formulated as the difference between the back-projected point, ${}_C\boldsymbol{v}_k$, as viewed from the current camera pose, and a corresponding point in the global model ${}_W\boldsymbol{v}_k$:

$$e_{\text{icp},k} = {}_W\boldsymbol{n}_k \cdot ({}_W\boldsymbol{v}_k - \boldsymbol{T}_{WCC}\boldsymbol{v}_k). \tag{4}$$

This geometric alignment error is projected along the surface normal, ${}_W\boldsymbol{n}_k$, and weighted according to the inverse covariance W_{icp} associated with the measurement uncertainty.

C. Kinematic Error Term

The kinematic constraint acts on both $\boldsymbol{T}_{WC}$ and $\boldsymbol{T}_{WB}$. It constrains the relative transformation between them $\boldsymbol{T}_{BC} = \boldsymbol{T}_{WB}^{-1}\boldsymbol{T}_{WC}$ to be close to $\tilde{\boldsymbol{T}}_{BC}$ measured from the robot kinematics. It has the following form:

$$\boldsymbol{e}_{\text{kin}} = \begin{bmatrix} {}_B\tilde{\boldsymbol{r}}_{BC} - {}_B\boldsymbol{r}_{BC} \\ 2\left[\tilde{\boldsymbol{q}}_{BC} \otimes \boldsymbol{q}_{BC}^{-1}\right]_{1:3} \end{bmatrix}. \tag{5}$$

The Information Matrix $\boldsymbol{W}_{\text{kin}}$ we used for the robot manipulator measurements was:

$$\boldsymbol{W}_{\text{kin}} = \begin{bmatrix} \sigma_{\text{kin},r}^{-2}\boldsymbol{I}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \sigma_{\text{kin},\alpha}^{-2}\boldsymbol{I}_3 \end{bmatrix}, \tag{6}$$

where we used $\sigma_{\text{kin},r} = 0.001$ m and $\sigma_{\text{kin},\alpha} = 0.003$ rad.

The manipulator pose measurement is not platform specific, allowing our system to run on any platform capable of providing a 6-DOF forward kinematics estimate. The information matrix should be updated according to the platform.

D. Odometric Error Term

The odometric error acts on both $\boldsymbol{T}_{WB_0}$ and $\boldsymbol{T}_{WB_1}$. It constrains the relative transformation between them $\boldsymbol{T}_{B_0B_1} = \boldsymbol{T}_{WB_0}^{-1}\boldsymbol{T}_{WB_1}$ to be close to $\tilde{\boldsymbol{T}}_{B_0B_1}$ measured from the robot odometry. It has the following form:

$$\boldsymbol{e}_{\text{odom}} = \begin{bmatrix} {}_{B_0}\tilde{\boldsymbol{r}}_{B_0B_1} - {}_{B_0}\boldsymbol{r}_{B_0B_1} \\ 2\left[\tilde{\boldsymbol{q}}_{B_0B_1} \otimes \boldsymbol{q}_{B_0B_1}^{-1}\right]_{1:3} \end{bmatrix}. \tag{7}$$

The Information Matrix $\boldsymbol{W}_{\text{odom}}$ we used for the robot odometry measurements was:

$$\boldsymbol{W}_{\text{odom}} = \begin{bmatrix} \sigma_{\text{odom},r}^{-2}\boldsymbol{I}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \sigma_{\text{odom},\alpha}^{-2}\boldsymbol{I}_3 \end{bmatrix}, \tag{8}$$

where we used $\sigma_{\text{odom},r} = 0.005$ m and $\sigma_{\text{odom},\alpha} = 0.003$ m.

The transformation measurement is not platform specific, allowing our system to run on any platform capable of providing a 6-DOF incremental pose measurement. The information matrix should be updated according to the platform.

E. Optimisation

To solve for the minimal cost c_{track} at each iteration we calculate the least-square solution

$$\underset{\delta\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{J}\delta\boldsymbol{x} + \boldsymbol{b}\|_2^2. \tag{9}$$

Blocks of the Jacobian $\boldsymbol{J}$ and residual $\boldsymbol{b}$ for the geometric alignment are populated and solved with a highly parallel tree reduction in CUDA; the Jacobians and residuals for the kinematic and odometric constraints are computed analytically, the Jacobian and residual for the linear prior are obtained either from marginalisation of previous states (in normal operation), or by initialisation at the first frame or following a loop closure.

The 24 x 24 system of normal equations is then solved on CPU using a Gauss-Newton Iterative method with a three level coarse-to-fine pyramid to compute the minimal state update $[\delta\boldsymbol{x}_0^T \quad \delta\boldsymbol{x}_1^T]^T$ which is then applied using the $\boxplus$ operator.

F. Partial Marginalisation and Linearisation

Rather than solving the full historical state graph at each time iteration, after solving for the current state and previous state, we marginalise out the errors and previous state into the linear prior for the next iteration. The equations for the Gauss-Newton system are constructed from the combined Jacobians, error terms and the previous and current states. The system of equations takes the following form:

$$\begin{bmatrix} \boldsymbol{J}_0^T\boldsymbol{J}_0 & \boldsymbol{J}_0^T\boldsymbol{J}_1 \\ \boldsymbol{J}_1^T\boldsymbol{J}_0 & \boldsymbol{J}_1^T\boldsymbol{J}_1 \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x}_0 \\ \delta\boldsymbol{x}_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_0^T\boldsymbol{b}_0 \\ \boldsymbol{J}_1^T\boldsymbol{b}_1 \end{bmatrix}. \tag{10}$$

After updating the current and previous states, we construct the linear prior for the next iteration $\boldsymbol{H}_{11}^*$ by marginalising out the previous state using the Schur-Complement:

$$\boldsymbol{H}_{11}^* = \boldsymbol{J}_1^T\boldsymbol{J}_1 - \boldsymbol{J}_1^T\boldsymbol{J}_0(\boldsymbol{J}_0^T\boldsymbol{J}_0)^{-1}\boldsymbol{J}_0^T\boldsymbol{J}_1 \tag{11}$$

$$\boldsymbol{b}_1^* = \boldsymbol{J}_1^T\boldsymbol{b}_1 - \boldsymbol{J}_1^T\boldsymbol{J}_0(\boldsymbol{J}_0^T\boldsymbol{J}_0)^{-1}\boldsymbol{J}_0^T\boldsymbol{b}_0. \tag{12}$$

The computed $\boldsymbol{H}_{11}^*$ and $\boldsymbol{b}_1^*$ are then used as the linear prior for the next instance of the optimisation problem upon arrival of a new frame. This partial marginalisation fixes the linearisation point, but each subsequent iteration will cause this point to change. We thus need to apply a first order correction based on the difference $\Delta\boldsymbol{x}$ between the new and old linearisation points [3],[5]:

$$\begin{aligned} \boldsymbol{H}_{11}^{*'} &= \boldsymbol{H}_{11}^* \\ \boldsymbol{b}_1^{*'} &= \boldsymbol{b}_1^* + \boldsymbol{H}_{11}^*\Delta\boldsymbol{x}. \end{aligned} \tag{13}$$

G. Initialisation

The state graph solved at each iteration requires a linear prior and a previous state. For the first frame we initialise the system with an arbitrary body pose and a camera pose predicted directly from the forward kinematics. The linear prior is then constructed as a deviation from the desired initial state:

$$\boldsymbol{e}_{prior} = \begin{bmatrix} {}_W\tilde{\boldsymbol{r}}_{WB} - {}_W\boldsymbol{r}_{WB} \\ 2\left[\tilde{\boldsymbol{q}}_{WB} \otimes \boldsymbol{q}_{WB}^{-1}\right]_{1:3} \\ {}_W\tilde{\boldsymbol{r}}_{WC} - {}_W\boldsymbol{r}_{WC} \\ 2\left[\tilde{\boldsymbol{q}}_{WC} \otimes \boldsymbol{q}_{WC}^{-1}\right]_{1:3} \end{bmatrix}. \quad (14)$$

H. Loop Closure

When the mapping module detects a loop closure, it computes a new camera pose $\boldsymbol{T}_{WC'}$ to bring the camera back into alignment with the map. Since only the base suffers from drift, we first make the assumption $\boldsymbol{T}_{B'C'} = \boldsymbol{T}_{BC}$, and then compute $T_{WB'} = \boldsymbol{T}_{WC'}\boldsymbol{T}_{B'C'}^{-1}$. For the next iteration after loop closure, we set the previous state to $\boldsymbol{x}_0 = ({}_W\boldsymbol{r}_{WB'},\ \boldsymbol{q}_{WB'},\ {}_W\boldsymbol{r}_{WC'},\ \boldsymbol{q}_{WC'})$ and use a linear prior of the same form as in the initialisation (equation 14).

V. Mapping

Like the original ElasticFusion, our map is divided into *active* and *inactive* regions. Tracking and fusing of new points is done against the active region of the map which is more recently observed. Patches of the active map which have not been observed for a set period of time become inactive. We maintain global consistency by matching currently observed portions of the active map with the inactive map and then applying non-rigid spatial deformations with a sparse deformation graph. We make no changes to the mapping module in ElasticFusion so the full details are omitted here. Please see [13] for further details.

VI. Results

We evaluate our system on real-world data across two criteria: quantitative tracking accuracy against ground truth and qualitative reconstruction consistency. Our ground truth trajectory data was collected using Vicon motion capture system with 10 Bonita cameras. This ground truth pose data is not suitable to construct a reliable ground truth scene reconstruction [30], as very small pose errors can lead to very large errors in reconstruction after depth projection. We therefore restrict our reconstruction analysis to qualitative statements. The computational hardware used consists of an Nvidia GeForce 1080 CPU with an Intel Core i7 CPU running at 3.60GHz. The system ran at 30FPS with an Asus Xtion Pro at a resolution of 640x480 pixels.

Since we require robot kinematic data in addition to camera frames, it is not possible to evaluate our system on existing synthetic datasets. We evaluate on ten real-world trajectories representing different types of motion. The first four trajectories feature a cross section of the types of behaviour we are interested in, featuring three scenarios that commonly cause Dense SLAM systems to lose tracking, as well as one easy trajectory. Sequence 1 features slow and smooth motion with plenty of geometry to aid tracking. This is typical of the type of sequence that dense visual SLAM systems perform well on. Sequence 2 features fast motion, including rapid rotation, but does not loop back on itself and retains plenty of interesting geometry. Sequence 3 features long stretches of low texture visual information with the robot looking either at carpeted floor or a tiled ceiling. Sequence 4 is very challenging, featuring loopy, erratic and fast motion. The remaining six trajectories cover a cross section of different types of behaviour, featuring some standard trajectories and some additional peculiar behaviour commonly encountered during robotic motion. Sequences 1, 4, 6, 9 and 10 feature at least one Loop Closure.

Type	ID	ElasticFusion	KO-Fusion	Length
Slow	1	**0.063293**	0.085115	4.7621m
Fast Rotation	2	0.835225	**0.142547**	10.9872m
Low Texture	3	0.958348	**0.350196**	6.7732m
Loopy	4	0.956216	**0.235642**	20.7392m
Slow	5	0.062735	**0.048303**	3.9627m
Low Texture, Fast Rotation	6	0.113347	**0.055187**	5.1444m
Slow	7	0.151327	**0.128612**	7.8781m
Slow, High Geometry	8	**0.009310**	0.015374	2.6699m
Continuous Rotation	9	1.167166	**0.244728**	7.1256m
Disconnected Regions	10	0.746062	**0.071342**	10.8354m

TABLE I: Comparison of ElasticFusion and KO-Fusion on 10 real-world trajectories. The table shows the absolute trajectory error in metres of the predicted trajectory as compared to Ground Truth. The Length is computed along the Ground Truth.

Through our experiments, we show that our dense RGB-D-KO system outperforms the equivalent RGB-D system on challenging trajectories typical of robot motion and is more robust in scenarios with fast motions or with little geometric variation. Elastic Fusion does show marginally better performance on two of the trajectories which could be due to a number of effects such as calibration or slippage in the odometry.

A. Absolute Trajectory Error

We use the absolute trajectory error (ATE) described in [30] to evaluate the predicted camera trajectories. Table I shows the root mean-squared error (RMSE) in metres for ElasticFusion and KO-Fusion against ground truth. Our system displays comparable performance on the slow trajectories, whilst proving much more capable on the more challenging trajectories where the RGB-D only system loses tracking.

Figure 3 shows the trajectory estimations for sequence 2 which features sections of fast rotation across previously unseen parts of the world. ElasticFusion struggles to maintain an accurate trajectory across the fast rotation where the visual information does not provide a good constraint on the system. In constrast, KO-Fusion is able to fall back on the robot's onboard measurements to help it navigate in these conditions.

B. Qualitative Map Comparison

Figure 4 shows a qualitative mapping comparison between ElasticFusion and KO-Fusion on sequence 2 - the fast

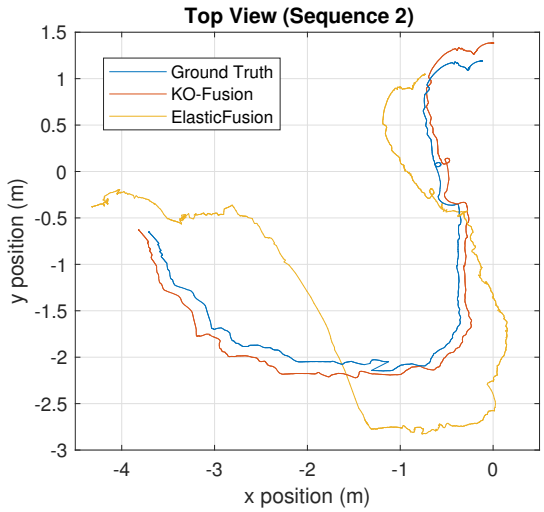

Fig. 3: Top view of the estimated poses from ElasticFusion and KO-Fusion on sequence 2. ElasticFusion loses tracking on the sections of fast rotation, whereas KO-Fusion is able to retain an accurate pose estimation.

trajectory with quick rotations. We achieve a higher degree of map consistency on this challenging trajectory. The top row displays the greater global consistency of our system. The robot first explores one section of the room and then does a fast camera sweep on its way around the corner to another section, during this motion visual tracking is lost, and ElasticFusion completely loses global alignment, such that the section of the room round the corner is mapped almost perpendicular to the ground plane of the first section. KO-Fusion maintains ground plane consistency through the turn, and the second section of the room is aligned correctly in the map.

The lower row shows local consistency in an object observed during the turn. The chair by the doorway is observed multiple times during the rotation, and our system is able to more accurately position subsequent observations of the object, causing better local consistency in the map.

C. Kinematic Drift

To demonstrate that our system is robust to drift in the odometric data, we show in figure 5 a visualisation of the accumulated error. The two wireframes show the predicted position of the system according to the raw kinematic and odometric data and according to our system. At the first frame of operation, the two wireframes are in alignment, but the visual tracking in the system allows it to account for the drift by maintaining alignment with the map.

VII. CONCLUSIONS

We have demonstrated what we believe to be the first Dense SLAM system to feature tight coupling of kinematic and odometric data from a mobile robot. Our system runs in real-time on any platform which provides incremental 6-DOF pose measurements of the robot base as well as pose measurements between this base and the camera. Our system jointly optimises a cost function incorporating the scene

Fig. 4: Qualitative map comparison between KO-Fusion (left) and ElasticFusion (right). The top two images highlight the improvement in global coordinate consistency, whereas the bottom row shows consistency in observing a single object (highlighted in red) before and after a fast rotation.

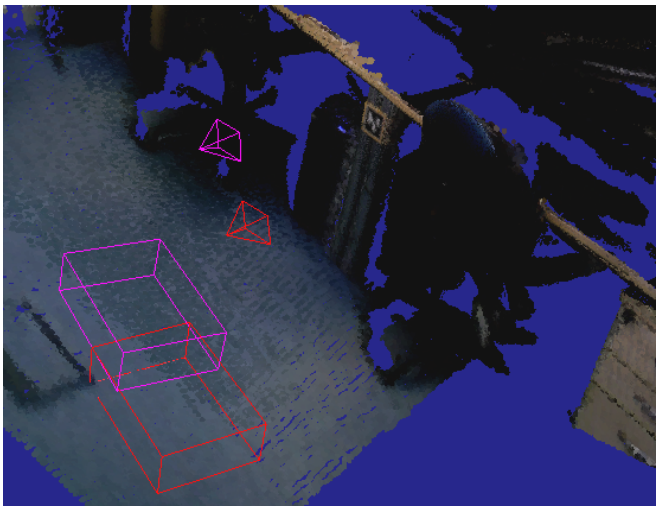

Fig. 5: Visualisation of the difference between predicted pose (pink) and raw kinematic and odometric data (red) about 30s into the simple trajectory.

geometry and robot actuator data to estimate the pose of both the robot base and a camera mounted on the end effector. It supports a real-time fully dense 3D reconstruction supporting loop closure.

We show through experiments on real-world data that our system performs comparably to the RGB-D only version on simple trajectories whilst being substantially more robust to challenging motion and visual data including fast and erratic motion with rapid rotation, and a lack of significant photometric and geometric variation.

VIII. ACKNOWLEDGEMENTS

Research presented in this paper has been supported by Dyson Technology Ltd. and the Engineering and Physical Science Research Council [EPSRC] as part of the High Performance Embedded and Distributed Systems Centre for Doctoral Training [HIPEDS].

REFERENCES

[1] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics (T-RO)*, vol. 33, no. 5, pp. 1255–1262, 2017.

[2] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-Direct Monocular Visual Odometry," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[3] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2014.

[4] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proceedings of Robotics: Science and Systems (RSS)*, 2015.

[5] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[6] A. I. Eliazar and R. Parr, "Dp-slam 2.0," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[7] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "DTAM: Dense Tracking and Mapping in Real-Time," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[8] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. A. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. J. Davison, and A. Fitzgibbon, "KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera," in *Proceedings of ACM Symposium on User Interface Software and Technolog (UIST)*, 2011.

[9] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of SIGGRAPH*, 1996.

[10] P. Besl and N. McKay, "A method for Registration of 3D Shapes." *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 14, no. 2, pp. 239–256, 1992.

[11] T. Whelan, J. B. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard, "Kintinuous: Spatially Extended KinectFusion," in *Workshop on RGB-D: Advanced Reasoning with Depth Cameras, in conjunction with Robotics: Science and Systems*, 2012.

[12] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao, "Chisel: Real time large scale 3d reconstruction onboard a mobile device," Pittsburgh, PA, July 2015.

[13] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," *International Journal of Robotics Research (IJRR)*, vol. 35, no. 14, pp. 1697–1716, 2016.

[14] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion," in *Proc. of Joint 3DIM/3DPVT Conference (3DV)*, 2013.

[15] S. Omari, M. Bloesch, P. Gohl, and R. Siegwart, "Dense visual-inertial navigation system for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[16] L. Ma, J. M. Falquez, S. McGuire, and G. Sibley, "Large scale dense visual inertial SLAM," in *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 2015.

[17] A. Concha, G. Loianna, V. Kumar, and J. Civera, "Visual-inertial direct SLAM," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[18] J. Engel, T. Schoeps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[19] T. Laidlow, M. Bloesch, W. Li, and S. Leutenegger, "Dense RGB-D-Inertial SLAM with map deformations," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2017.

[20] P. Puri, D. Jia, and M. Kaess, "Gravityfusion: Real-time dense mapping without pose graph using deformation and orientation," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, September 2017.

[21] M. Klingensmith, S. Srinivasa, and M. Kaess, "Articulated robot motion for simultaneous localization and mapping (arm-slam)," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, January 2016.

[22] R. Scona, S. Nobili, Y. R. Petillot, and M. Fallon, "Direct visual SLAM fusing proprioception for a humanoid robot," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-Septe, no. Figure 1, pp. 1419–1426, 2017.

[23] K. Nagatani, S. Tachibana, M. Sofue, and Y. Tanaka, "Improvement of Odometry for Omnidirectional Vehicle using Optical Flow Information," Tech. Rep., 2000.

[24] J. Inthiam and C. Deelertpaiboon, "Self-localization and navigation of holonomic mobile robot using omni-directional wheel odometry," in *TENCON 2014 - 2014 IEEE Region 10 Conference*. IEEE, oct 2014, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/7022281/

[25] "Kuka YouBot." [Online]. Available: http://www.youbot-store.com/

[26] A. Gfrerrer, "Geometry and kinematics of the Mecanum wheel," *Computer Aided Geometric Design*, vol. 25, no. 9, pp. 784–791, dec 2008.

[27] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[28] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253511000571

[29] M. Bloesch, H. Sommer, T. Laidlow, M. Burri, G. Nützi, P. Fankhauser, D. Bellicoso, C. Gehring, S. Leutenegger, M. Hutter, and R. Siegwart, "A Primer on the Differential Calculus of 3D Orientations," *CoRR*, vol. abs/1606.0, 2016. [Online]. Available: http://arxiv.org/abs/1606.05285

[30] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012.