A survey on VSLAM and Visual odometry for Robotics

1st Senthil Palanisamy

Student

Northwestern

senthilpalanisamy2020@u.northwestern.edu

***Abstract*—The goal of this survey is to gain a broad knowledge on the techniques of SLAM and to gain an insight into the evolving role of vision in present day SLAM. The primary objective is not to be coarse and learn these systems at a block diagram level but to dig deeper and understand them at the mathematical formulation level. Visual SLAM is a huge community. With the introduction of new sensors, the role of vision in SLAM is actively getting redefined. The paper selection methodology is such that papers are sampled across all sensing modalities that can be used in combination with a vision system. Attention was also paid to distribute papers across different SLAM frameworks and visual SLAM techniques. Thus at the end of this paper, it is my primary objective to be well versed in all of the vocabulary used across all of visual SLAM system and to summarize my understanding and inference in a clean way for future reference.**

I. Introduction

SLAM stands for simultaneous localization and mapping. It is the act of inferring motion as well reconstruction of the whole map which was navigated. In terms of rigid body like a camera, the localization is about estimating a trajectory of rigid body transform (Rotation and Translation) as well as trying to reconstruct the 3D map structure through the images. There are three types of approaches in doing visual slam. A concise block diagram of the whole system is shown below

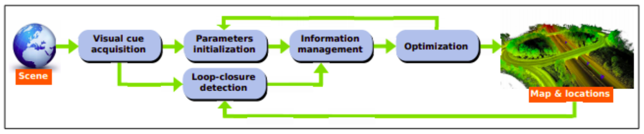


Fig. 1: A block diagram view of a SLAM system

We will discuss a few broad classifications of SLAM system in this section before stepping further, based on classification presented in [21]

Bases on the kind of features used in the visual pipeline of the SLAM system, a SLAM system can be categorised as follows

- Feature based - Features are detected in every frame and then these features are matched to estimate the camera transformation
- Direct image - Image is directly used without any feature extraction step
- Semi direct - A hybrid between both. Direct methods are used for estimating the local point cloud. while feature based methods are used for aligning the point cloud and estimating the camera pose.

Based on type of sensor used, a Visual slam used can be categorised as follows

- Monocular - A single camera is used
- Stereo/ MultiView - Two cameras or more than two cameras are used
- RGBD - Depth cameras are used.
- visual-inertia - Inertial sensors are used along vision sensors

Based on the technique for information fusion between different sensors with the system, a SLAM system can be categorized as

- tightly coupled - All Probabilistic filters fall into this category. The information esimate from each sensor is fused at each possible time step.
- loosely coupled - A trajectory estimate from two different sensing modalities are obtained in isolation and integrated finally. Tightly coupled systems are more accurate than loosely coupled systems but are more expensive.

Based on the density of features mapped by the SLAM system, it can categorised as follows

- sparse- Only a few features are mapped by the SLAM system. All feature based approaches fall into this category
- Dense- A dense reconstruction of the scene obtained. It must be noted that not all direct method are dense. In fact, they are quite a lot spare direct methods as well

Based on core technique used for trajectory and map estimation, SLAM systems can be categorized as

- Filter based - The core of state and map estimation is recursive bayessian estimation
- Optimization - States and map points are obtained by optimizing the bundle adjustment cost function.

II. Concept Glossary

In this section, the basic concepts and notation used in structure from motion problems or visual SLAM are briefly described. This section is an summary of all keys ideas I came across while doing the survey.

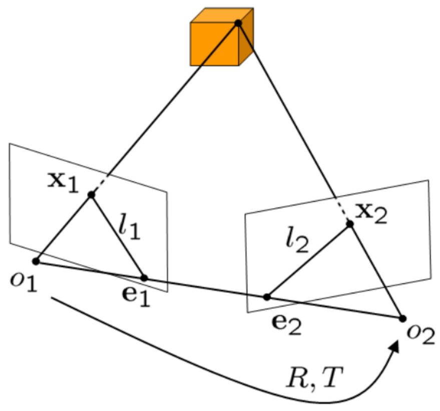


Fig. 2: An illustrative image for a two view setting

- **Epipolar constraint:** The camera centers and the 3D point being lie on the same plane. The epipolar constraint is compactly written as

$$x_2^T \hat{T} R x_1 = 0 \tag{1}$$

where x_2 and x_1 are normalised pixel co-ordinates (pixel co-ordinates which have been converted to to 3D location on the virtual image plane, where the z component of the homogenous transformation is one. It must be noted that the process of normalisation invloves the knowledge of Intrinsic parameters of the camera). The above constraint can be more intuitively understood by analysing the same idea from the view of a vector triple product. The fact that three vectors a,b and c lie on a plane is the same as stating that the volume of the paralleopiped spanned by these vectors is zero, which can be mathematically stated as

$$a^T(b \times c) = 0$$

In our case, the fact that the vectors $\overrightarrow{\mathrm{o1X}}, \overrightarrow{\mathrm{o2o1}}$ and $\overrightarrow{\mathrm{o2X}}$ all lie on the same plane can be represented as

$$\text{volume} = x_2^T(T \times Rx_1) = 0$$

$$\implies x_2^T \hat{T} R x_1 = 0$$

The matrix $E = \hat{T}R$ is called the essential matrix

The points e1 and e2 where the line joining the camera centers intersect with the image planes are called the epipoles.

- **Fundamental matrix** If the intrinsic calibration of the camera is unknown, the epipolar constraint can be extended to include the calibration equations,

  $$\acute{x_2}^T K^{-T} \hat{T} R K^{-1} \acute{x_1} = 0$$

  $F = K^{-T}\hat{T}RK^{-1}$ is called the fundamental matrix. This is useful in cases where reconstruction is desired with unknown calibration. There is a more nasty case of this equation where images are taken from two different cameras with unknown camera calibrations

  $$\acute{x_2}^T K_1^{-T} \hat{T} R K_2^{-1} \acute{x_1} = 0$$

  Typical calibration with unknown camera parameters is not studied extensively since its often deemed an unnecessary problem to be solved.

- **8 point algorithm:** [32] is one of the most basic and first proposed algorithms for reconstructing a 3D scene. This algorithm proves that the 3D structure can be recovered from 8 point correspondences between the two views. The various steps in the algorithm are quickly summarized below

  – **Approximate Essential Matrix computation:** The epipolar constraint in stated earlier can be stated in a slightly different way as

  $$x_2^T E x_1 = a^T E^S = 0$$

  where

  $$E^S = (e_{11}, e_{21}, e_{31}, e_{21}, e_{22}, e_{22}, e_{31}, e_{32}, e_{33}) \in \mathbb{R}^9$$

  is a vector containing all entries of the essential matrix

  $$a = (x_1x_2, x_1y_2, x_1z_2, y_1x_2, y_1y_2, y_1z_2, z_1x_2, z_1y_2, z_1z_2)^T$$

  $a \in \mathbb{R}^9$ is the kronecker product of $x_1 \text{and} x_2$ For n points, which gives n linear equations the above equations can be further extended as

  $$\xi E^S = 0$$

  where $\xi = (a^1, a^2, ..., a^n)^T \in \mathbb{R}^{9\times n}$ The null space of the matrix xi gives the solution for the essential matrix

  – **Projection into essential space:** The essential matrix computed above does not fulfill all the necessary constraints for a matrix to be essential. It turns out that the necessary and sufficient condition for a matrix to be essential is that the first two singular values are equal while the third is zero. In order to project this matrix into the essential space, its factored according to its SVD values $E = U\text{diag}(\sigma_1, \sigma_2, \sigma_3)V^T$ The first two singular values $(sigma_1, sigma_2)$ are replaced by $\sigma = \frac{\sigma_1+\sigma_2}{2}$ while the third singular value is replaced by zero. Thus the projection of the matrix computed in step1 onto the essential space then becomes $E = U\text{diag}(\sigma, \sigma, \sigma_3)V^T$. This is the solution to minimizing the Frobenius norm between the computed matrix and its projection in the essential space. In general, since we fix an arbitrary scale for the baseline, the first two singular values are replaced by ones, resulting in projection into the normalized essential space. $E = U\text{diag}(1, 1, 0)V^T$.

  - **Recovering displacement:** Since we know that essential matrix $E = \hat{T}R$, the problem of recovering camera displacement becomes a matrix decomposition problem of the essential matrix. The solution for this decomposition turns out to

$$R = R_Z^T(\pm\frac{\pi}{2})V^T$$

$$\hat{T} = UR_Z(\frac{\pi}{2}\Sigma U^T)$$

where R_Z^T stands for rotation about Z axis

One key point about this algorithm is that it decouples the structure and motion estimation into two separate problems, estimating the structure and the motion separately.

- Bundle Adjustment: The 8 point algorithm is not the best method to be used especially when there is noisy data. The algorithm is not robust meaning that there is no guarantee that for small changes in the observed pixels due to noise, there will only be a small change in the estimated 3D points or camera motion. For a two view case with N points, the cost function for bundle adjustment can be stated as follows

$$E(R, T, X_1, ..., X_N) = \Sigma_{j=1}^{j=N}|\tilde{x_1}^j - \pi(X_j)|^2 + |\tilde{x_2}^j - \pi(R, T, X_j)|^2$$

This cost function aims at minimizing the reprojection error between the observed 2D co-ordinates and the 2D coordinates obtained by projecting the 3D point onto the image planes. It is quite straight forward to extend the same cost function to M views as follows

$$E(R, T_{i=1..m}, X_j)_{j=1..N} = \Sigma_{i=1}^{m}\Sigma_{j=1}^{N}\theta_{ij}|\tilde{x_1}^j - \pi(R_i, T_i, X_j)|^2$$

where the reprojection error of all N points is minimized across all m views. It must be noted that a given point may not be visible in all of the m views, the θ_{ij} accounts for this by making the whole cost function zero if a given point j is not visible in the view i and one otherwise i.e., θ_{ij} = 1 if the point j is visible in the view i and θ_{ij} =0 otherwise. It must be not that the above function is non-convex and hence, there aren't any closed form solutions. Therefore, we must resort to iterative methods to find the minima of the bundle adjustment cost functions.
It can also be seen that the cost function is unconstrained but in reality the projection of points between two views is bound by epipolar constraints. Therefore, the same optimization can also be performed in a constrained setting by incorporating the epipolar constraints.

$$E(x_1^j, \lambda_1^j, R, T) = \Sigma_{j=1}^{j=N}|\tilde{x_1}^j - \tilde{x_1^j}|^2 + |\tilde{x_2}^j - \pi(R\lambda_1^j x_1^j + T)|^2$$

An alternate formulation of bundle adjustment cost function is to have simple least square cost function that minimizes the error between the true 2D co-ordinates (x_i^j) and the observed 2D co-ordinates $\tilde{x_i}^j$ but the values that x_i^j can take on are constrained by the epipolar geometry i.e.,

$$E(x_{i\,j=1..N}^j, R, T) = \Sigma_{j=1}^{N}\Sigma_{i=1}^{2}|x_i^j - \tilde{x_i^j}|^2$$

such that $x_1^{jT}\hat{T}Rx_1^j = 0, x_1^{jT}e_3 = 1, x_2^{jT}e_3 = 1$ j = 1,...N

- **Types of Bundle Adjustment:** There are small variants in Bundle Adjustment, that are often discussed without reference. Here is short summary of the concept taxonomy

– **Motion only BA / Pose Graph Optimisation:** The optimisable parameters are only the camera movement and hence, the map parameters are fixed. This is known as motion only BA and seeks to optimise the same reprojection error but with known and fixed map.
– **Structure only BA:** The optimisable parameteres are only the map points or the structure parameters with a fixed camera locations. The same reprojection error is used in this case.
– **local BA** This is the whole Bundle adjustment where both camera parameters and map parameters are optimisable in the reprojection error but it is done only in the vicinity of the current frame being observed. What constitutes small vicinity is down to the hueristic of the author in the paper.
– **Global BA / BA:** When BA is simply specifcied as ”BA”, it simply refers to the Global bundle adjustment where the all points in the map are projected into all key frames and the reprojection error across all frames and points are optimised. It must be noted that Global BA is extremely costly and only done at the end of the optimisation pipeline where a good estimate of parameters have already been obtained and its only done to refine the accuracies further.
– **Iterative Point Cloud:** This is closely related to the idea of Motion only BA but the key difference here is that the depth of 3D point correponding pixel is known and is a fixed quantity. Thus in this technique, we seek to find the motion of the camera that algin the point cloud that we observe in the current frame with the existing point cloud we have observed so far.
– It must be noted that there can be combinatorics between the sets {Motion only, Structure only and plain} with {Global, Local}, which means that there can be global motion only BA or local Structure only BA etc.. But when something is plainly specified as BA, it generally refers to Global BA with both structure and motion optimisation unless otherwise specified within the context

- **Camera Pose Parameterisation:** The camera Pose is a 6 DoF system and it sounds natural to represent this by a six parameter explicit representation. This kind of representation though is the most straightforward parameter representation to optimise over, doesn’t turn out to be ideal particulary due to the singularities associaties with Euler Angle representation. Therefore, typically, an over parameterised system is employed so that it is singularity free but this now means that constraints have to be imposed on the extra degrees of freedom during optimisation. The standard representation choice is the SE(3) matrix. A standard SE(3) matrix can be defined as

$$SE(3) \equiv g = \left\{ \left[\begin{array}{cc} R & T \\ 0 & 1 \end{array} \right] \middle| R \in SO(3), T \in R^3 \right\} \subset \mathbb{R}^{4x4}$$

It can be clearly seen that this is an over parameterised system, with 16 parameters and there has to be some constraints. The constratints arise from the fact that $R \in SO(3)$, which means that R has to satisfy the following constraints $R^T R = I$ and $det(R) = 1$, which is a very concise and elegant way of specifying 6 constraints of roatatin matrices. The bottom most row of the SE(3) matrix are always $[0, 0, 0, 1]$, which altogether specifies 6+4=10 constraints. Therefore, camera poses are represented by an overparameterised 16 parameter system, with 10 constraints and this is nasty problem for optimisation. The easier way to solve this problem is not to solve this optimsation problem in the space of SE(3) but in its lie Algebra se(3). A compact way of specifying the relationship between SE(3) and its lie algebra se(3) is as follows

exp: se(3) $\rightarrow$ SE(3); $\zeta \rightarrow e^{\zeta}$

log: SE(3) $\rightarrow$ se(3); $e^{\zeta} \rightarrow \zeta$

The lie algebra of SE(3) ζ is called the exponential coordinates of SE(3) or sometimes called the twist coordinates and is an easier space to optimise over since $\zeta \in \mathbb{R}^6$ and therefore is a more explicit parameter space to optimise over. It has to be noted that there is a bijective mapping between SE(3) and se(3) meaning for every element in SE(3), there exists a unique element in se(3) and vice-versa. It has to be noted that taking large steps in se(3) space is not a good idea since the se(3) is a tanget space to SE(3) at the identity and can be considered to be a linear approximation to the actual SE(3) manifold but for our purposes where the pose change between any two camera frames is not so significant, we can safely optimise in the linearly approximated lie algebra space se(3).

- **Similarity Transform:** With a single monocular camera, structure from motion estimation always comes with an inherent scale ambiguity. Imagine moving camera at a far away distance from a huge building and moving the same camera at a very close distance to a toy building which have the same visual appreance. It can be shown through such examples that we can get two completely different solution for images which look the same but generated under two very different settings. This is root cause of inherent scale ambuiguity in Monocular systems and hence the estimates are only upto a scale. Mathematically, a similarity transform can be written as

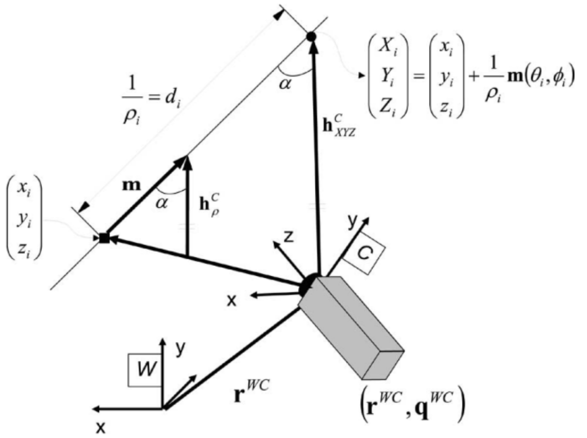
$$S = \begin{bmatrix} sR & T \\ 0 & 1 \end{bmatrix}$$

where $s \in \mathbb{R}^+$ is positive real number. These similarity matrices also form a group SIM(3) and we can construct an analogous lie algebra sim(3) for the group so that we can a more explicit parameterisation to optmise over. The elements of this lie algebra $\zeta \in \mathbb{R}^7$, which is logical since we have 7 degrees of freedom in a similarity transform. We can jump between the lie group and lie algebra through logarithms and exponentiantions similar to SE(3) group.

- **Why inverse depth is the right paramterisation choice rather than the actual depth:** [3] proposed that using inverse depths instead of depths improves the performance of the system. The limitations of direct depth parameterisation are
  - **delayed Point initialisation:** When points are first initialised they never contribute to the whole location or mapping since their uncertainities are high and they are only added after a few passing frames reduce the uncertainity in depth. The key insight to see how points with uncertain depth can contribute to the system is that points with uncertain depth contribute very little in the camera translation they serve as very useful bearing references for rotation calculation. Probabilisitcs framworks before this took a very conservative approach towards this by not including points in SLAM processing pipeline until the depth of points have been estimated with very low uncertainity.
  - Points at Infinity: Points at infinity produce very low

parallax or disparity and hence, they usually don't make it into the SLAM pipeline. An example of a paricle at inifnity could be a star in the sky which is obeserved when viewing an image of the road. While these infinity points don't contribute to camera translation, they are very good landmarks for estimating camera rotations.

Inverse depth parameterisation can be understood by looking at the picture given below Instead of using special treament for uninitialised points or points at infinity, [3] proposed a unified framework under which this can be handled elegantly.



$\frac{1}{\rho_i} = d_i$
α
$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i}\mathbf{m}(\theta_i, \phi_i)$
$\mathbf{h}^C_{XYZ}$
$\mathbf{m}$
$\mathbf{h}^C_\rho$
$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$
z
y
C
x
W
y
x
$\mathbf{r}^{WC}$
$(\mathbf{r}^{WC}, \mathbf{q}^{WC})$

Fig. 3: Inverse depth parameterisation

While a Eucledian 3D point is represented by (X_i, Y_i, Z_i), a point in inverse depth is represented by $y_i = (x_i, y_i, z_i, \theta_i, \phi_i, \rho_i)^T$ where x_i, y_i, z_i refer to optical center position of the camera in world co-ordinate frame from which the point was first observed. θ_i, ϕ_p refer to the azimuthal angle and elevation angle and ρ_i is the inverse of depth of the point (1/d). In particular, a point in 3D can be represented as

$$x_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \frac{1}{\rho_i} m(\theta_i, \phi_i)$$

where $m = (cos\phi_i sin\theta_i, -sin\phi_i, cos\phi_i cos\theta_i)^T$ is just a unit vector in the direction joining the camera optical center with the point expressed in cylindrical co-ordinates. In simple terms the parameteristaion can explained as a ray originating from the optical center of the camera from which the point was first observed and the length of ray which measures the depth of a point is paramterised by the inverse depth. In the context of proabablisic filters this mode of parameterisation turns out to be more useful. The paper goes on to show that the linearity indexed of the inverse depth parameterisation is more desirable than the depth parameterisation. In particular, the linearity metric is defined to be

$$L = \left| \frac{\frac{\partial^2 f}{\partial x^2}|_{\mu_x} 2\sigma_x}{\frac{\partial f}{\partial x}|_{\mu_x}} \right|$$

From the equation, we can clearly see that the Linearity term is the ratio of second derivative of a function at the point μ_x to its first derivative at the point mu_x multiplied by the covariance associated with x. Thus, if the function were close to linear, the second order term will be zero or much smaller in magnitude when compared to the first order derivative. The paper goes on to show that bayessian filter measurement update when applied to inverse depth parameter has much smaller linearlity index when to compared to the one with direct depth parameterisation and thus more desirable in the context of bayessian filters. Furthermore, point with infinte depth can be represented safely and thus contributes to system information. Point can be initialised instantly and directly put into the pipeline without having to worry about their uncertainity in depth.

The depth estimation in SLAM system particular in outdoor can sometimes be infinity. Imagine looking at the sky where the objects are faraway. We would observe zero disparity or little disparity for the same feature point observed from two views, the distance estimate shoots upto infinity. To avoid this, an inverse depth parameterisation is usually a safer choice since we are unlikely to see points with zero or close to zero depth (unless there is some sticker attached to the lens of the camera) in our observation scene thereby eliminating the infinity problem.

- **Surface Representation:** Some applications of slam particularly those from the graphics and AR community try to model point cloud generated as a surface. [8] proposed a way to model the surface by using voxels. A voxel is the 3D analogue of a pixel and is a 3 dimensional cube. The idea behind Truncated Singed Distance Function is that the voxel carries a value of zero, for points on the surface, positive values for points infront of the surface and negative values for points at the back of it. An illustraive example of stroing a face as a TDSF surface is shown below

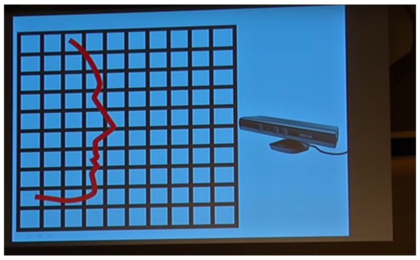


Fig. 4: An example of a surface to be mapped

It must be noted that this kind of representation allows to spot the surface easily since we just have to spot the zero

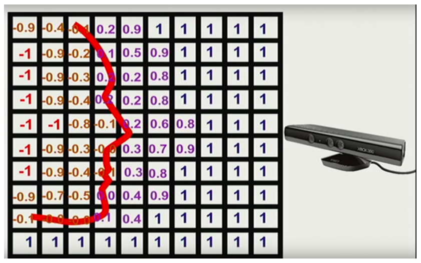


Fig. 5: A TDSF representation of the surface

crossing. It also makes it very easy to render a ray from the camera as per the model since we have to just follow decreasing values along a ray until it detects a positive value. The min and max values are tied between +1 and -1, beyond which the values default to the respective extreme values.

- **Accuracy Metrics:** There are two common error metrics for comparing the tracjectory of Grounth Truth with the an estimated trajcetory
- RMSE error: The ground turth trajectory and the predicted trajectory are aligned at the start point and the error is the pairwise between between every corresponding point along both trajectories. An example of such an error measure is shown below

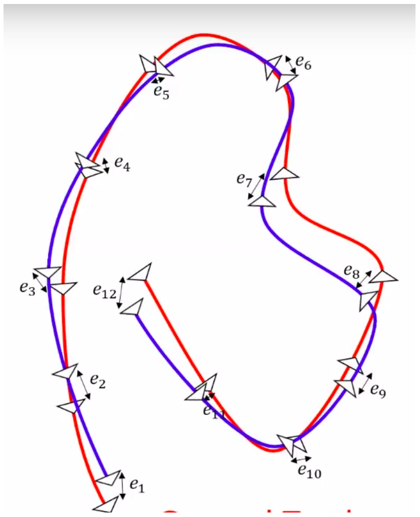


Fig. 6: RMSE error

- Odometric error: The trajectories are aligned considering every point in the predicted trajectory as the starting point and it is aligned against the corresponding point in the ground truth trajectory and the error is the distance between the end point on the ground truth trajectory and the end point on the predicted trajectory after a few operation sequences (typically 4 or 5). A figure illustrating the same is shown below. The figure shows only one term in such calculation- Only the first point is aligned and an error is estimated after four operations. The two trajectories will be aligned at every point on the predicted trajectory and such a measure will be found for each of those points and summed up

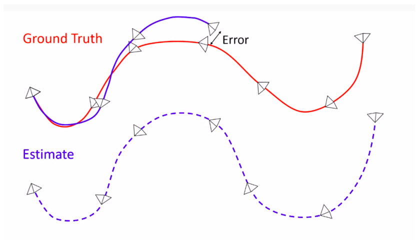


Fig. 7: odometric error

- **Optimisation methods:** There are various methods to optmise over the bundle Adjustment cost function given above. In general, this class of problems fall into a wider category of problems known as Non-linear least squares. A simple least squares formulation can be written as

$$min_x \sum_i (a_i - f_i(x))^2$$

where a_i is the ground truth output value and $f_i(x)$ is the output estimate for the data point x. When this function $f_i(x)$ is linear, this is called linear least squares and its called non-linear when $f_i(x)$ is non-linear. In our Bundle adjustment case, the function $f_i(x)$ involves a non-linear projection function and hence, this falls under non-linear least squares. Some of the optimisation techniques that have been developed for non-linear least squates are as follows

  - **Gradient Descent:** The gradient descent just follows the direction of steepest decent until it hits a local minima. If E(x) is the cost function to be optimised, then one step of gradient descent can be written as

$$x_{k+1} = x_k - \epsilon \frac{dE}{dx}(x_k)$$

  - **Newton's method:** A problem with the gradient descent above is the step size to choose and newton's method solves this probem by constructing a taylor's series approximation of the function and moves towards the point where the approximated function hits zero and then jumps to the point corresponding in the actual function. In general, the function approximation is either linear or quadratic. Quadratic methods are typically employed and an update step using a second order newton method is shown below

$$x_{t+1} = x_t - H^{-1}g$$

    where H is the hessian and g is the gradient. This method typically converges in few iterations than gradient descent but it must be noted that hessain computation is very costly and hence, the time per iteration shoots up as compared to gradient descent

  - **Guass-Newton Method:** A problem with Newton's method is it assumes that the hessian is a positive definite (so that we are marching towards a minima point). In cases where its not a positive definite matrix, the hessian has to approximated by a positive definite matrix. Gauss-Newton method is one such method. The hessain can be written down as

$$H_{jk} = 2\sum_i \left(\frac{\partial r_i}{\partial x_j}\frac{\partial r_i}{\partial x_k} + r_i \frac{\partial^2 r_i}{\partial x_j \partial x_k}\right) \qquad (2)$$

Dropping the second term ensures that the hessian is always positive definite. Hence, Hessian can be approximated as

$$H_{jk} \approx 2\sum_i J_{ij}J_{ik}, \text{with} J_{ij} = \frac{\partial r_i}{\partial x_j}$$

More compactly, the Hessain can be written a matrix way as

$$H \approx 2J^T J, \text{with Jacobian } J = \frac{dr}{dx}$$

Substituting this approximated Hessian and $g = 2J^T r$ into the update for Newton's method leads to the newton-gauss method

$$x_{t+1} = x_t - (J^T J)^{-1} J^T r$$

And it can be seen that this approximation is good only if the magnitude of the second term is much lesser than the magnitude of the first term in hessian equality equation

– **Levenberg-Marquardt:** Newton's method is good in the initial steps but as we get closer to the solution, gradient descent is more efficient. This is the key idea behind Levenberg-Marquardt method, where a single iteration update is given by

$$x_{t+1} = x_t - (H + \lambda I_n)^{-1} g$$

Where H is hessain, g is gradient, and I is the identity matrix. As we can see, when lambda is zero, the update becomes equal to newton's method itself and as $\lambda \to \infty$, the method almost becomes the gradient update and hence by varying λ, we get a hybrid update that is somewhere inbetween gradient descent and newton's method.

III. Front End- Direct/ Feature based

In this section we will look deeply into a few example systems for each of feature based, direct image and semi-direct methods.

A. Feature based SLAM system- ORB/ORB2 SLAM:

[38] integrated a lot of key concepts and developed a powerful realtime slam. One of the foremost premise of the work was to use a single feature representation for both mapping and tracking, where prior works used to use different representation. This is a very useful idea in cutting down the runtime of the algorithm. It builds upon the parallel threads idea of PTAM but uses three threads instead of two: one each for tacking, local mapping and loop closing. Each of these are modules incorporating a lot of key ideas. A detailed description of each of these modules is given below.

– **Tracking:**ORB features are extracted for the current frame being viewed. A constant velocity motion model is used in predicting the expected pose of the camera if the previous tracking was successful. If this constant velocity motion is full-filled, a narrow search of map points is enough where as a wider baseline search is needed if the assumption is violated. The camera pose is then optimized with the known correspondence. It can be noted here that the optimization performed here is a motion only BA, meaning that the map points are fixed and the pose is optimized. If tracking is lost at any poit, the pose of the camera is initialized using a global query search using bag of words. This means a Bag of Visual words is extracted for each frame and each words is stored in the database along with a list of all the key frames in which they occur. One a key frame with enough correspondence is found, the camera pose is reinitialized using PnP solver. Once the pose is estimated, all stored map points are projected into the new frame and new points are added to the map if point pairs with strong correspondence is found. A very generous strategy is used in deciding if the new frame under observation can be given the status of a key frame. The selection rules used mostly revolve around uniqueness and trackability of frames (In simple words, the rules can be thought of answers to questions like is this frame really new or redundant and does this frame have a rich features that can be tracked). It must be noted here the frames are not inserted in the tracking thread. The decision about if a frame can be given the key frame status is taken om the tracking thread. Insertion and Culling of key frames and map points are done by the Mapping thread.

- **Mapping:** All the camera poses is stored as a graph data structure. The nodes in the graph are the camera poses and the edge between the nodes indicates how many common points exits between the two camera poses. This graph is called co-visibility graph and can be very dense and hence, a global bundle adjustment over this entire graph becomes an intractable problem very soon. Though this co-visibility graph was already in existence, a novel idea from the paper was use an essential graph instead of a co-visibility graph. The essential graph is a spanning tree of the co-visibility graph (A spanning tree is a connected graph that contains all nodes from a graph and has only the least number of edges such that the resulting graph is still connected. The details of the essential graph construction is not explained in detail (A graph could have a lot of such spanning trees) but it is safe to assume that the spanning tree whose sum of weight of edges is the maximum is the one to be preferred. Map points and key frames once inserted are checked if they are redundant. A stricter condition is used in key frame culling and map point culling so that the robustness of the tracking is improved. A key frame insertion means adding a node to the co-visibility graph and adding the same node to the essential graph as well. When adding the node to the essential graph, the node is connected to that node which share the most point with the node. A key frame culling means that the node has to be removed from both graphs. A node removed from an essential graph should mean that the graph becomes disconnected. So a proper spanning tree is constructed with some heuristics after culling a key frame.. A local bundle adjustment is performed for new key frames. This local optimization is just done over the key frames, its neighbors and the neighbors of those neighbors.
- **Loop closing:** Bag of visual words is extracted from the latest key frame inserted and a similariy measure is computed for this key frames with all its neighbors. Let s_{min} be the minimum of all such similarity measures. Then the database is queried to see if any other key frame has similar BoW. All frames whose similarity measure is lower than s_{min} are disregarded. If frame which has a very high similarity score is detected, a loop closure is

detected and a similarity transform is computed as an estimation of the drift in camera pose. To close the loop, new edges are added in the co-visibility map so the graph is fused. Finally a pose graph optimization is done meaning the estimated error in drift is distributed across the essential graph so that a smooth trajectory is achieved.

Initialization is one of the prime problems with SLAM systems and this paper incorporated two alternate mechanisms for estimating it. One is homography computation using DLT if the scene is planar and the other is fundamental matrix estimation using 8 point algorithm is the scene is non-planar. In reality, both these models are put into action and a robust heuristic has been proposed in the paper that automatically chooses the right one, thus reducing the constraints on pose initialization, which was problem with previous works.

[37] extended the ORB SLAM idea to stereo and RGBD images. The key idea in this is the chose of parameterisation for representing the point. A stereo keypoint is represented by $x_s = (u_L, v_L, u_R)$ where u_L and v_L are the image co-ordinates of the feature point in the left image and u_R is the horizontal co-ordinate of the matched image point in the right image (rectified image so that the image correspondences lie of a point on the left image always lie on a image point on the corresponding horizontal row). A similar formulation can be calculated for RGB-D images, where u_L, v_L can be calculated with the image of the camera and we can calculate a virtual right horizontal co-ordinate using the formula given below

$$u_R = u_L - \frac{f_x b}{d}$$

where f_x is the focal length of the lens and b is the baseline between the light projector and infrared camera and d is the depth of the point calculated by the sensor. so we, now, get the same triplet representation for both stereo and RGB-D images and the rest of the pipeline can stay the same and build on top of this abstracted feature. The rest of the pipeline is similar to ORB SLAM which involves a motion-only BA estimation with a new frame and then local BA is calculated with a small neighborhood and then a full BA is calculated once a loop closure is detected. It has be noted that in this choice of parameterization, the uncertainty in depth is actually reflected by the uncertainty in the estimate of u_R which indirectly depends on depth. Levenberg-Marquardt method is the optimization method of choice here. The key frame insertion strategy is similar to the previous work except that we have new information to utilize due to the introduction of stereo / RGB-D. We can directly calculate the depth of points from a single view without any scale ambiguity here and this becomes very useful. It turns out that close points are very useful for estimating scale and translation information whereas faraway points are useful for tracking rotation. For example: When majority of the key points are faraway as in the case of a car driving on a road, a good heuristic for key frame insertion is to check if the number of close points in the reference key frame has fallen below a critical threshold and if the current frame can provide a satisfactory number of close enough points. If the two conditions are satisfied, the current frame is then inserted as a new key frame.

B. Direct SLAM system:

[15] was the first system to propose the concept of direct image alignment instead of using feature based approaches. The key idea was to model the camera transformations not as Rigid body transforms but by a similarity transform to account for the scale drift in the formulation explicitly. Uncertainty in depth is modeled by a probabilistic formulation on the inverse depth map. The key steps in the total algorithm is detailed below

- **Tracking** This continuously tracks new camera frames and estimates the se(3) pose of the camera with repect to the previous key frame.
- **depth map estimation** This refines or replaces key frames. The depth is refined by filtering over many per-pixel, small base line stereo.
- **map optimization** Once new key frame is added to the system, the previous key frame is added as a tracking frame. and it is incorporated into the global frame. To detect loop closure and scale drift, a similarity transform to close by key frames is estimated.

It has a very specific bootstrapping for intializing the first key frames. Each key frame consists of three things- A camera image, an inverse depth image and variance of the depth map.

[14] proposed a direct method of optimization in pixels rather than computing intermediate feature which SLAM systems till then were using. To understand how this is achieved, we must first understand the concept of photometric calibration. We all know geometric calibration which relates a pixel to a 3D position and geometry based on the camera pose. A photometric calibration relates pixel intensities to irradiance. A photometric calibration model is written as

$$I_i(x) = G(t_i V(x) B_i(x))$$

where $I_i(x)$ is the irradiance B_i is the observed pixel intensities, t_i is the exposure time V(x) is the lens attenuation (also known as vigetting), which is a function $V : \Omega \to [0,1]$ and G(x) is the reponse function, which maps the light recevied at the sensor to pixel intensities $G : \mathbb{R} \to [0,255]$ The vignetting effect can be best described by the figure 11.

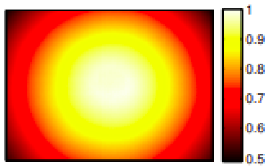


Fig. 8: Vignetting or lens attenuation effect: Light received reduces radially from the lens center

Once we have the photometric model correctly calibrated, we can make a photo metric correction for all pixels in a given frame of the image using the below expression

$$I_i(x) := t_i(x)B_i(x) = \frac{G^{-1}(I_i(x))}{V(x)}$$

Once we have made the photo metric correction, we can optimize over photo consistency to estimate our unknown parameters. The cost function can be written as

$$E_{pj} := \sum_{p \in N_p} w_p \left\| (I_j[p^{'}] - b_j - \frac{t_j e^{a_j}}{t_i e^{a}_{i}} (I_i[p] - b_i)) \right\|_\gamma$$

Here $\|.\|_\gamma$ stands for huber norm, which is quadratic in error initially and shifts to being linear. $p^{'}$ stands for a point whose depth is parameterized by its inverse projection d_p.

This cost function is simply a SSD (sum of square differences) over a small neighborhood of pixels for a point p which has already been observed in one frame i and which is now being observed in another frame j. But this error is only for one point in one frame. Hence to account for all points in all frames, the full photometric error can be written as

$$E_{photo} := \sum_{i \in F} \sum_{p \in P} \sum_{j \in obs(p)} E_{pj}$$

This cost function is simply the SSD photometric cost computed for over every point that is available in all key frames with the current frame. The whole data is stored as a factor graph, making it efficient for doing inference and updation.

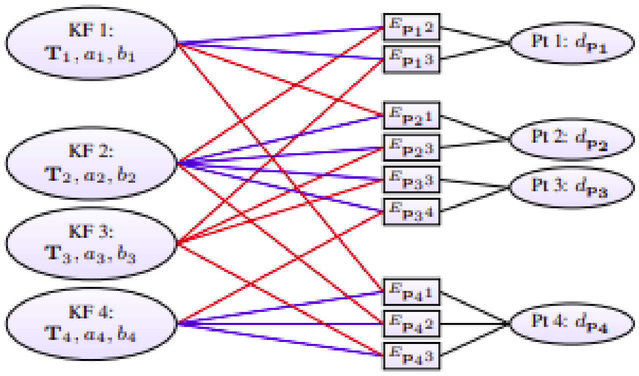


Fig. 9: An example for illustrating the factor graph used

A factor graph is a bipartite graph, meaning the graph can be divided into two groups isolated set of vertices where in there is no connecting edges between nodes of the same group. The above figure can be understood by seeing that each key frame is represented by a unique node and the inverse depth parameter associated with each point also forms a node. Each photometric error term for each point also forms a node. Edges between nodes just establish the dependency of the cost function on the parameters it needs. For example, the cost function E_{p12} depends on the p1 in Key frame1 (the source point:The source point dependency is indicated by blue line) and its matched against p2 in Keyframe2 (This is the matching point: The matching point dependency in the figure is represented by blue lines) and the inverse depth parameterization of point1 (which is indicated by black lines). Once we understand this, we can immediately appreciate the efficiency of factor graphs. A cost function between points in two frames once calculated can be used for calculations there after, there by avoiding repeated computations. The factor graph nodes are updated once there is a change in any of their connected dependencies. It also has to be noted that this method performs windowed optimization, which means that the history maintained is not infinite and hence the keyframe and points sizes have to maintained at a specified threshold. Once the keypoints or keyframe count have reached their maximum limit, the less relevant points and keyframes are marginalized out. The paper establishes heuristics as to which frames need to be marginalized. Marginalization is a better idea than just cutting out history since we preserve a part of the history. When a node in the factor graph has to be marginalized, all its dependencies are integrated across the whole probability distribution of the given node. Thus this paper incorporates a way to maintain a compact history that doesn't run out of memory.

[23] was extension to DSO, where loop closure feature was added to the DSO pipeline and a global optimization was added using a co-visibility graph. Adding a loop closure feature to a direct SLAM method requires some revisions in the actual approach itself. Usually, loop closure are handled by a maintaining a database of BoW and matching the BoW in the current frame to all the key frames observed so far. This poses an implicit challange to a DSO system since direct method typically do not

focus on selecting point feature which are good for tracking since they rely on pixel intensities. Therefore, the criterion for selecting points in DSO pipeline is changed to favor corner like feature points. This also means that not all points selected are used both for tracking and estimation. The corner-like points are used for both build BoW (for loop closure matching) and tracking while other points are used only for tracking. This is still in stark contrast to the indirect methods where corner features are explicitly selected. Here, the number of features is maintained the same while favoring selection more towards corner points if they are available. The BoW used in this work is the Dynamic Bag of Words (DBoW3). For each loop closure candidate, features in the frame are matched against reference key frame and SE(3) transformation is estimated using PnP solver if a matching frame is found. Once an initial estimate is found out, the estimate is refined further using Gauss-Newton method. Not all points that are matched have an estimated depth associated with them. Therefore, this is accounted for in the cost function shown below

$$E_{loop} = \sum_{q_i \in Q_1} w_1 \left\| S_{cr}\pi^{-1}(p_i, d_{p_i}) - \pi^{-1}(q_i, d_{q_i}) \right\|_2 + \sum_{q_j \in Q_2} w_2 \left\| \pi(S_{cr}\pi^{-1}(p_j, d_{pj}) - q_j) \right\|_2 \tag{3}$$

where π stands for the projection matrix and p_k stands for points in the candidate frame and its associate depth is d_{p_k}, q_k and d_{q_k} stands for points in the matched keyframes and the inverse depth associated with those key points respectively.Q_1 stands for the set of points whose depth has been estimated and Q_2 stands for the set of points whose depth has not been estimated. Putting all this together, all this cost functions tries to accomplish is to minimize the disparity in the 2D pixel location after projecting the matched 3D points into the current frame for those points whose depth is known while also fully minimisng the difference in 3D point location between matched points in the two frames for those points whose depths have been estimated. Its just reprojection error expressed in different forms depending on what is known: for feeatures points with unknown depth, the matched points are projected into the current frame and their difference is minimised. For those feature points whose depths have been already estimated, the difference in the 3D point location corresponding to feature points are minimised. Global bundle adjustment is costly and hence only a global pose optimsation is done. These global optimsations are done in such a way that they don't affect the poses in the windowed optimisation. The pose constraints are represented by a sim(3) constraints and the windowed optmisation estimates the aboslute SE(3) transformation between the frames. Once a frame is to be taken out of the window and marginalised, the SE(3) constraints are converted to SIM(3) constraints and inserted into the pose graph.

C. semi-direct methods:

In between the two extremes of direct and feature based methods are semi-direct method. These methods use direct pixel based photometric error for aligning two images and generating the point cloud but the actual integration of the point cloud into the existing map and optimizing the camera pose for such an alignment is done by feature based methods.

[18] proposes a semi-dense direct method. It is semi-dense because not all the points are used for map con-

structions and only a sparse set of points are used and it is direct in the sense that its cost function directly seeks to reduce the photometric error between pixel patches corresponding to a selected point. Like PTAM, this method also uses parallel threads where the camera pose estimation and the mapping are decoupled and run in separate threads. The camera pose estimation and pixel patch location estimation proceeds in three separate steps

– **Sparse Model based Image Alignment:** The relative transformation between two frames is found by minimizing the negative log-likelihood of intensity residuals. An intensity residual is defined to be photometric difference between two pixel patches that is viewable in the two views. Here a pixel patch is a 4 * 4 patch around an image pixel corresponding to a 3D point which is viewable in the two frames.

$$T_{k,k-1} = argmin_{Tk,k-1} 1/2 \sum_{i \in R} |I(T_{k,k-1}, u_i)|^2)$$

$$I_{T,u} = I_k(T^-1(u, d_u) - I_{k-1}(u)) \forall u \in R$$

where $T_k, k-1$ is the frame to frame transformation describing the camera movement and u_i is the pixel patch location This cost function simply means that we would like to choose that transformation that minimizes the photometric error of the image patches observed in two frames. Gauss-Newton method is the optimization method of choice in this work.

– **Relaxation through Feature Alignment:** This step refines the position of image patches based on a fixed camera transformation and 3D point locations.

$$u_i = argmin_{u_i^{'}} 1/2 \left\| I_k(u_i^{'}) - A_i I_r(u_i) \right\|_2 , \forall i$$

We seek to make a adjustment to the pixel patch location here such that the photometric error corresponding to reprojection is minimized. A affine transformation A_i is applied on the reference patch so both patches are aligned for calculating the photometric cost. It is contrary to the previous step where no affine correction was used. This is purely because in the previous step the two frames very close (in fact, they are consecutive) and hence, there wasn't a need where as in this stage, two pixel patches which are matched may be several frames away and hence, an affine correction becomes inevitable. A large patch of 8 *8 pixels is used here.

– **Pose and Structure refinement:** This is the final step where the camera pose and map points are optimised together. Before doing this, all camera poses are optimized. i.e.,

$$T_{k,w} = argmin_{Tk,k-1} 1/2 \sum_{i \in R} \|I(T_{k,w}, u_i)\|^2)$$

The above step is typically called motion only BA. The difference between the above step and the first Image alignment step is that the image alignment step is only for calculating the relative pose between latest frame and the previous frame while motion only BA optimizes over all camera poses. Finally, a local BA adjustment is applied across a local neighborhood of the latest frame where the pose and the structure are jointly optimized.

The mapping thread adds new points and initializes their depth. Recursive bayesian filter

$$p(\bar(d_i^k|d_i), i) = iN(\bar(d_i^k|d_i, \Sigma)) + (1-i)U(\bar(d_i^k|d_i^{min}), d_i^{min}, d_i^{max})$$

where it can be seen that a normal distribution is picked with a probability i and a uniform distribution is picked with probability 1-i. The uniform distribution just account for outliers and for a proper inlier point, the depth is assumed to be normally distributed. Once a point is added, it has very high uncertainty since all points are intialized with the mean depth of their respective frames and very high variance. Once points have been added, they go through multiple measurements before their uncertainty shrinks and they are added to the map for motion estimation. The whole image is divided into 30 * 30 pixel grids and a FAST corner with highest shi-Tomasi score is taken to be the selected point. This is done across different scales of images and thus we have feature points distributed through the image at all scales.

IV. Stero SLAM sytems:

Subsequently, the concept of direct SLAM was extended to stereo systems through [48]. This helps in recovering the depth of the points accurately thereby eliminating the scale dirft completely completely. The basic formulation of the DSO is kept the same except that now we have both temporal and static steteo contraints. The energy function that is being minimised can be stated as below

$$E = \sum_{i \in F} \sum_{p \in P_i} \sum_{j \in obs^t(p)} E_{ij}^p + \lambda E_{is}^p$$

where E_{is}^p is the energy arising out static stereo residuals and E_{ij} stands for the temporal stereo residuals. The parameters that are being optimised in above two energies differ. The parameters that are being optimized in a temporal stereo are $\eta_{geo} = (T_i, T_j, d, c)$ where d stands for the depth of the points and c is the weighting constant in the Huber norm and $T_i T_j$ stand for the transformation between the two keyframes. With the static stereo, the geometric parameters being optimized drops down to $\eta_{geo} = (d, c)$ since the transformation between the two cameras is fixed. In all these DSO methods discussed so far, the optimization method of choice is the Gauss Newton method and a single update of Gauss newton method is given by

$$\delta\eta = -(J^T W J)^- 1 J^T W r$$

where η stands for the parameters being optimized and J stands for Jacobian and W is diagonal matrix containing weights and r is the residuals coming from the least squares cost function. An update step in Gauss Newton's method is given by

$$\eta_{new} = \delta\eta \boxplus \eta$$

It has to be noted that the function η defined above is just an addition element in case of adding two se(3) lie algebra elements but takes an alternate definition when having one se(3) and SE(3) element. In particular,

$$\boxplus : se(3) \times SE(3) \rightarrow SE(3), x \boxplus T := exp(\hat{x})T$$

In partcular, the use of a stereo system in a DSO setting gives us the following advantages

- Absolute scale information can be obtained. We get $SE(3)$ transform between frames as opposed to SIM(3) transform between frames.
- Static stereo provides the initial depth estimate, which avoids a very specific bootstrapping sequence that is typically used in a Monocular stereo
- Both of them complement each other in a few scenario. Static stereo can accurately locate closely by points while the depth of far away points is resolved by temporal stereo. They even complement each other in degenerate cases when epipolar lines are parallel.

A factor graph is used for tracking information in this system.

V. Complementary sensing modalities

In this section, we describe the use of other sensors that can complement vision sensors to make the SLAM system more reliable. We restrict our discussion to two sensing modalities - a RGBD system and Visual Inertial system.

A. *RGBD slam systems:*

AR community has a different view to the SLAM problem. [39] was one of the papers from AR using a depth sensor camera Kinect. The availability of reliable depth information from one of the sensing modalities opens up a new way of thinking about the problem. The basic block diagram of kinect fusion is given below

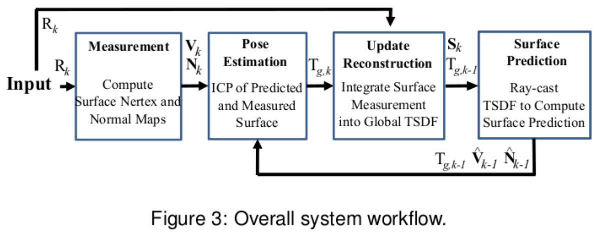

Input
R_k
R_k
Measurement
Compute Surface Nertex and Normal Maps
V_k N_k
Pose Estimation
ICP of Predicted and Measured Surface
T_g,k
Update Reconstruction
Integrate Surface Measurement into Global TSDF
S_k
T_g,k-1
Surface Prediction
Ray-cast TSDF to Compute Surface Prediction
T_g,k-1 V̂_k-1 N̂_k-1
Figure 3: Overall system workflow.

Fig. 10: overall system workflow

Since the kinect gives depth information directly, this information can be transformed to a point cloud directly. A vertex map point in sensor frame can be represented as

$$V_k(u) = D_k(u)K^{-1}u$$

where u is the homogeneous representation of a pixel

K is the intrinsic camera matrix

$D_k(u)$ is the depth associated with pixel and

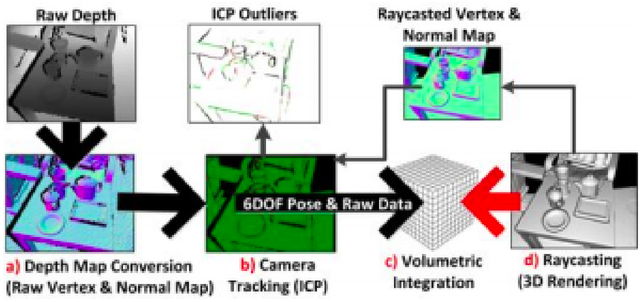
$V_k(u)$ is vertex map point in the 3D point cloud. A point in a map is both represented by a vertex point and a surface normal. The surface normal can be just computed by taking the cross products of two vectors connecting the vertex point with neighboring points. Mathematically, this can be written as

$$\begin{aligned} N_k(u) = v[(V_k(u+1,v) - V_k(u,v))\times \\ (V_k(u,v+1) - V_k(u,v))] \end{aligned} \tag{4}$$

where $v(x) = \frac{x}{\|x\|_2}$ Thus this normal is a unit vector perpendicular to the plan containing the vector joining the current vertex to the vertex immediately above and the current vertex to the vector right of it, which is very good approximation for the surface normal at the point. Once vertex and surface normal are computed, ICP (Iterative closest Point) algorithm is applied to find the camera transformation so that the surface seen in the current frame aligns with the surface existing surface observed so far. The surfaces are typically represented by Truncated Signed Distance Function (TSDF) described previously. The idea of ICP and TDSF have been already described in the key concepts section. Once a final camera pose is obtained, the new points seen by current frame are updated to the global surface maintained in a TSD. Once both Camera pose and Surface prediction is done, the surface is rendered from the TSDF data structure so that the surface mapping can be seen online.

An extension to the original work for kinect fusion was proposed through [26]. An overall block diagram for the whole system is shown below

Here we can see that the module added to the whole system is the generation of synthetic depth map. Once a reliable or decent reconstruction of the surface geometry is constructed, a depth map can be obtained internally



Raw Depth
ICP Outliers
Raycasted Vertex & Normal Map
6DOF Pose & Raw Data
a) Depth Map Conversion (Raw Vertex & Normal Map)
b) Camera Tracking (ICP)
c) Volumetric Integration
d) Raycasting (3D Rendering)

Fig. 11: Block diagram of revised kinect fusion

from the system when doing raycasting to project a view of the surface to the user at the current camera pose. This synthetic depth map can be used for aligning the ICP point cloud as opposed to the original 3D point cloud data from kinect since that can be noisy. This effectively increases the robustness of tracking. The authors also propose an extension to the system to allow slam in a dynamic environment. Since the system uses a time average of depth maps for storing the surface depth, the algorithm is intrinsically not affected by small movements. But the whole pipeline is messed up when a large object like a human hand moves across a screen slowly. In order to counter this, the authors use a very simple strategy. The basic assumption is that the surface has already been mapped to a reliable measure already. When a moving object enters the camera's view, it causes a huge disparity between the model's belief about the surface depth present in a given view point. If such a big disparity is observed, a 3D connected component analysis (including depth dimension) is done and the foreground (where the dynamic interacting object) exists is cleanly segmented out while the background (already present surface) is left undisturbed. Using this simple strategy, the author show great results where users can interact with the AR environment through hand touches.

B. *Visual Inertial systems:*

Another line of sensor suites that typically used in a SLAM system are inertial sensors. [40] was one of the first succesful visual inertial Monocular SLAM system. There are two kinds of visual inertial fusion namely

- **loosely coupled fusion:** Pose estimates are obtained from the camera measurement and IMU separately and then the pose parameters are adjusted to minimize the error between the two
- **tightly coupled fusion:** IMU pose estimates are integrated with the camera measurement estimate at every measurement step. Its pretty obvious that a tightly coupled fusion will give better results but at increased computation cost. [36] was one of the first works to talk on the IMU calibration, a multi-state constraint setup for visual inertial fusion and a measurement model for IMU. An IMU state is denoted by

$$X_{IMU} = [I_{G^i}^{-T} \quad b_g^T \quad G_{VI}^T \quad b_{a^T} \quad GPI^T]$$

Here I is the IMU-affixed frame and G is the global frame of reference. I_{Gq} is the rotation in quaternion the specifies the rotation from frame G to frame frame I, $G_{VI}G_{PI}$ are the IMU's estimate of position and velocity expressed in G frame and b_g, b_a are the bias term in the gyro and accelerometer measurement. Unlike, other sensor, the noise characteristics of the IMU sensor change slightly with time and hence, they need to re-estimated on the fly and therefore, they are not known parameters but parameters to be estimated in our pipeline. It must also be noted that we have a very good initial guess for our these noise parameters to start with. The methodology used in the MCKF system is the idea of a multi-constraint filter. Rather than using the key points observed to impose pairwise constraints between two key frames, the paper imposes multi-frame constraint based on all the key frames in which a particular key frame is visible. The gyroscope and accelerometer of the IMU can be written as

$$\hat{a_T} = a_t + b_{at} + R^t_w g^w + n_a$$

$$\hat{\omega_t} = \omega_t + b_{\omega_t} + n_w$$

where $\hat{a}$ stands for acceleration measurements and its affect by the bias noise b_{at}. The accelerometer should also offset for the gravity since it measures gravity and it is also corrupted by white noise. With a gyrometer, the angular velocity $\hat{\omega}$ is affected by both bias noise and white noise. Though there are few other kinds of noise, there are the most important ones to model for the purpose of our experiment. The position, velocity and orientation estimates with a gyro can be written down as follows

$$\begin{aligned} p^w_{b_{k+1}} = p^w_k + v^w_{b_k}\Delta t_k + \\ \int\int_{t\in[t_k,t_{k+1}]} (R^w_t(\hat{a}_t - b_{a_t} - n_a) - g^w)dt^2 \end{aligned}$$

The position estimate of the $k+1^{th}$ frame with respect to the world frame w is obtained by adding the previous position and doing a single integral over its present velocity and a double integral over its acceleration. The velocity estimate is given by

$$v^w_{b_{k+1}} = v^w_{b_k} + \int_{t\in[t_k,t_{k+1}]} R^w_t(\hat{(a_t)} - b_{a_t} - n_a) - g^w)dt$$

The velocity is obtained by adding the previous velcity with the integral of the acceleration at the frame k.. Finally the orientation of the robot is estimated by

$$q^w_{b_{k+1}} = q^w_{b_k} \bigotimes \int_{t\in[t_k,t_{k+1}]} \frac{1}{2}\Omega(\hat{w}_t - b_{w_t} - n_w)q^{b_k}_t dt$$

where $\bigotimes$ indicates the multiplication operation between two quaternions. While the integral term computes the rotation between k and k+1 frames, the multiplication with the previous rotation q^w_k ensures that the global orientation is obtained. There is big problem with the formulation above and this was discussed in [20] and [19]. The primary problem crops up from the fact that IMU runs at a very high speeds (60 Hz) as compared to normal camera. Since its huge burden on the computation resources to do this integration at every frame, the IMU values are pre-integrated between two key frames i.e., all IMU readings between two key frames are used without any measurement correction terms and a final pose estimate at the new key frame is used for filtering or optimization. However, doing pre-integration and estimating the robotic movement in the world frame poses problems since if find adjustment to the robot pose in the trajectory through optimization, the whole of calculations have to be carried out again but it should be noted that the robot movement in the body frame remains the same between the two key frames irrespective of its starting pose. Thus the whole of pre-integration is carried out in the body frame axis. Thus,

$$R^{b_k}_w p^w_{b_{k+1}} = R^{b_k}_w(p^w_{b_k} + v^w_{bk}\Delta t_k - \frac{1}{2}{g^w_K}^2) + \alpha^{b_k}_{b_{k+1}}$$

$$R_w^{b_k} v_{b_{k+1}}^w = R_w^{b_k}(v_{b_k}^w - g^w \Delta t_k) + \beta_{b_{k+1}}^{b_k}$$

$$q_w^{b_k} \bigotimes q_{b_{k+1}}^w = \gamma_{b_{k+1}^{b_k}}$$

These equations are just a direct consequence of changing the frame of reference from the world frame to the body frame. Here α, β and $gamma$ are the terms which depends on integration and hence, these are the values estimated in the pre-integration and the rest are all constant.

$$\alpha_{b_{k+1}}^{b_k} = \int\int_{t \in [t_k, t_{k+1}]} R_t^{b_k}((\hat{a_t}) - b_{at} - n_a)dt^2$$

$$\beta_{b_{k+1}}^{b_k} = \int_{t \in t_k, t_{k+1}} R_t^{b_k}(\hat{a_t} - b_{at} - n_a)dt$$

$$\gamma_{b_{k+1}}^{b_k} = \int_{t \in [t_k, t_{k+1}]} \frac{1}{2}\Omega((\hat{\omega_t}, b_{wt}, -n_w))\gamma_{b_k} dt$$

The IMU sensor parameters, unlike other sensors, has to estimated on the fly and visual-inertial alignment is a mechanism for initialising the system. The initialising sequence can be carried out as follows

- **gyroscope bias calibration:** To begin with a loosely coupled visual inertial fusion is to used to estimate the IMU bias. The IMU readings are pre-integrated and an initial SLAM involving both camera trajectory and map points is obtained by initializing the vision system with standard bootstrapping algorithm such as 8 Point or 5 point algorithm Therefore, the following cost function is

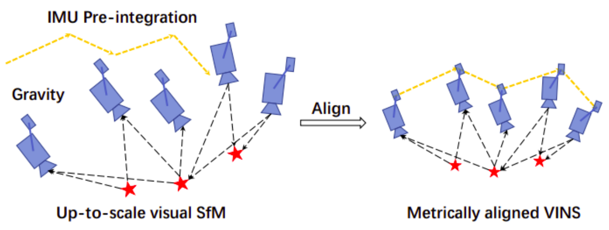


Fig. 12: Pre alignment for estimating IMU bias

minimized to estimate the bias

$$\min_{\delta b_w} \sum_{k \in B} \| q_{b_{k+1}}^{c0}{}^{-1} \bigotimes q_{b_k}^{c0} \bigotimes \gamma_{b_{k+1}^{b_k}} \|^2$$

$$\gamma_{b_{k+1}}^{b_k} \approx \gamma_{b_{k+1}}^{\hat{b_k}} \bigotimes [\frac{1}{2} J_{b_w}^{\gamma} \delta b_w]$$

Here $q_{b_{k+1}}^{c0}{}^{-1} \bigotimes q_{b_k}^{c0}$ gives the orientation in quaternion of the $k+1^{th}$ with respect to the k^{th} frame as per vision measurements and $\gamma_{b_{k+1}}^{b_k}$ is the reverse estimate, the orientation of $k+1^{th}$ frame with respect to the k^{th} frame. In an ideal world, the quaternion product of these two will map to the qauternion with zero rotation for all three axis but due to inaccuracies and the noise in the IMU, these measurements will be off and hence, minimizing this cost function with respect to IMU bias gives a initial estimate of bias.

- **velocity, Gravity, Vector and Metric Scale** In this stage the other IMU parameters and the scale parameter are initialized. The scale parameter arises since we use a mono ocular system and hence, we should set the scale to the right value initially as is done with most slam systems. This is done by optimizing

$$\underset{min}{\chi_I} \sum_{k \in B} \| z_{b_{k+1}}^{b_k} - \hat{H}_{b_{k+1}^{b_k}} \chi_I$$

Here χ_I is the vector consisting of all inter frame velocities, gravity and scaling constant

$$\chi_I - = [v_{b0}^{b0}, v_{b1^{b1}}, ..., v_{b_n}^{b_n}, g^{c0}, s]$$

where v_{bn}^{bn} is the body velocity at the frame n expressed in the body frame co-ordinates. $H_{b_{k+1}^{b_k}}$ is the IMU measurement model, which converts IMU measurements to camera poses and $\hat{z_{b_{k+1}}}^{b_k}$

is the measurement of the same obtained using camera.

* **Garvity Refinement:** Though we have estimated gravity in the previous, a more precise estimate of the gravity vector can be obtained by incorporating another information we know: The value of gravity. Thus by constraining the magnitude of gravity vector.
* **Completing initialization:** After these, the global frame is setup by aligning it with the gavity z axis and all pre-integrated IMU points are transformed to world co-ordinates.

Finally, for normal operation, the visual inertial measurements are tightly coupled. A figure illustrating the same is given below

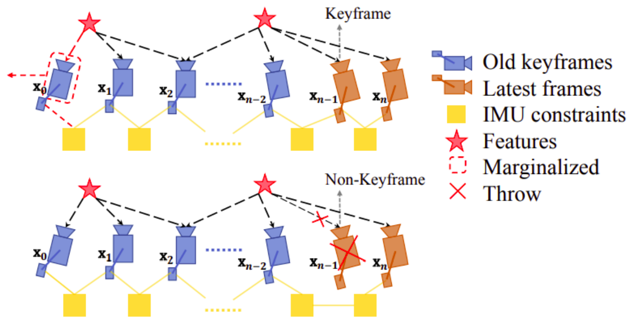


Fig. 13: An illustration of a tightly coupled VIO system

Thus, the IMU provides transformation constraints based on its reading and the camera also provides a set of constraints based on its observation of landmarks. The error between these two readings are minimized using least squares error and this gives the camera trajectory and the landmark points. An overall block diagram of the whole system is shown in the figure

First, the measurements are pre-processed: A KLT tracker is initialized for tracking features in a camera

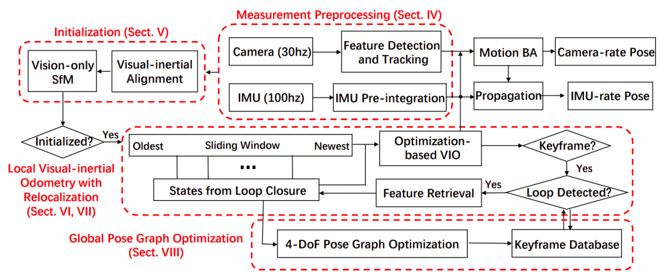


Fig. 14: Visual inertial system VINS block diagram

and the IMU readings are pre-integrated to find an initial trajectory. A motion only BA finds the camera trajectory as per camera measurement and then the estimates from the IMU and camera are loosely coupled to estimate the IMU parameters. Once initialized, the system does VIO bundle adjustments, which acts in a tightly coupled fashion to integrate IMU and camera measurements. The algorithm also actively searches for loop closures using DBoW and corrects the trajectory using a 4 DoF pose optimization graph if a loop was detected. Here, 4 DoF system is used instead of 6, since the IMU gives the absolute values for roll and pitch angles. It can also be noted that the system can disable map refinement and run only Motion only BA to estimate camera trajectory alone for systems with low compute power. There are implementation an of Stereo visual inertial system like [45]. in which the scale ambiguity can be avoided in the beginning due to the use of stereo system and the mapping points are constrained by two objectives one due to the static stereo and the other due to temporal stereo, thereby increasing accuracy. If one reads through these systems, they will be quick to realize that most of the formulation is the except the same at the objective function, which includes cost for both temporal and static stereo. It must be noted that stereo visual inertial systems are expensive in terms of computation costs as compared to a mono ocular system. Thus most of the works focus on coming up with effective strategies that can cut down the computation time for the system. The authors of the VINS systems also extend their system further and propose a general framework for SLAM in [41], where the framework has the flexibility to support multiple sensors. In general, local sensors estimate quite precise but drift prone measurements for SLAM while global sensors supply coarse but drift free estimation for the SLAM system. These global sensors are integrated into pose graph optimization to correct the drift in the system.

VI. Information Management- Graphs and Factor graphs

SLAM systems are often constrained by memory and computation power. Therefore, an efficient representation of data is necessary and inevitable. In this section we discuss the graph and factor graph way of data representation.

A. Graph based representation

The poses and the map point observed from each pose can be represented elegantly as a graph based framework. The initial framework for such a representation was first proposed by Sebastian Thrun in [46] and later, Cyrill Stachniss wrote a very good tutorial on its usage in [25]. The basic concepts in these papers can be described as follows

In the pose graph representation, every pose of the robot is represented by a node and the edges between the nodes represent the constraints the nodes have to obey based on the landmarks they observe from those

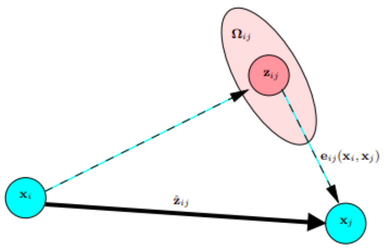


Fig. 15: A simple Pose graph

poses. Hence, the objective is to find the values of these nodes that minimize the error in the absolute constraint imposed by the landmark observation. This is done by negative log likelihood minimization and the solution from this gives the best set of poses that satisfy the imposes constraints. This graph optimisation can be posed as non-linear least squares minimization problem and can be optimized using a Gauss Newton minimization.

$$l_{ij} \propto [z_{ij} - z_{ij}(\hat{x_i}, x_j)]^T \Omega [z_{ij} - z_{ij}(\hat{x_i}, x_j)]$$

which states that the likelihood is proportional to the error between the observed measurement and the anticipated measurement. z_{ij} is the observed measurement and $z_{ij}(\hat{x_i}, x_i)$ is the anticipated measurement between the points x_i and x_j and Ω is the information matrix (inverse of the covariance matrix). The Mahalanobis distance is used for measuring the residual between the observed measurement and the predicted measurement. For notational convenience, lets define $e_{ij}(x_i, x_j) = z_{ij} - z_{ij}(\hat{x_i}, x_j)$, which just computes the difference co-ordinate wise difference between observed measurement and the predicted measurement. Our objective then is to minimise the liklihood function

$$F(x) = \Sigma_{(i,j)\in C} e_{ij}^T \Omega_{ij} e_{ij}$$

Let $F_{ij} = e_{ij}^T \Omega_{ij} e_{ij}$ denote a single term in the likelihood function, which just adds us all the residuals obtained from all such constraints. Our objective now is to negative log likelihood function

$$x^* = x^{argmin} F(x)$$

A solution can be found using Gauss Newton method, when we have a good initial guess of the starting position.

$$\begin{aligned} F_{ij}(x \ \ + \Delta x) &= e_{ij}(x \ \ + \Delta x)^T \Omega_{ij} e_{ij}(x \ \ + \Delta z) \\ &\approx (e_[ij] + J_{ij}\Delta x)^T \Omega_{ij}(e_{ij} + J_{ij}\Delta x) \\ &= e_{ij}^T \Omega_{ij} e_{ij} + 2 e_{ij}^T \Omega_{ij} J_{ij} \Delta x + \Delta x^T J_{ij} \Omega_{ij} J_{ij} \Delta x \\ &= c_{ij} + 2 b_{ij} \Delta x + \Delta x^T H_{ij} \Delta x \end{aligned}$$

where $c_{ij} = e_{ij}^T \Omega_{ij} e_{ij}, b_{ij} = e_{ij}^T \Omega_{ij} J_{ij}, H_{ij} = J_{ij} \Omega_{ij} J_{ij}$. With this local approximation, we can go ahead and simplify the likelihood function

$$\begin{aligned} F(x \ \ + \Delta x) &= \Sigma_{(i,j)\in C} F_{ij}(x \ \ + \Delta x) \\ &\approx \sigma_{(i,j)\in C} c_{ij} + 2 b_{ij} \Delta x + \Delta x^T H_{ij} \Delta x \\ &= x + 2 b^T \Delta x + \Delta x^T H \Delta x \end{aligned}$$

where $c = c_{ij}, b = \Sigma b_{ij}$ and $H = \Sigma_{ij}$. Taking derivative with respect to Δx leads use to a solution of minima, which can be written down as

$$H \Delta x^* = -b$$

$$x^* = x \ \ + \Delta x^*$$

It must be noted here that x should be in minimum parameter representation or else this iterative style of optimisation will break down. For example, a common choice of parameterization for rotation is to use $SO(3)$ matrices but a rotation has 3 degrees of freedom but a rotation matrix has 9 values and hence, has 6 implicit constraints (This has been clearly described in the earlier sections). Therefore, optimizing directly in the SO(3) space will lead to matrices that violate the constraints, unless their constraints have been correctly specified as a constrained optimization problem. But such a constrained optimization problem is very diifcult to solve and therefore, an optimization is always carried out in the implicit representation and is transformed to the preferred choice of parameterization. This has also been clearly discussed in the previous sections. Graph based optimization becomes prevalent in SLAM system and it becomes inevitable that someone creates a graph framework for the larger community to work with. [30] proposed a general framework for graph based optimization problem. This framework was setup with an intention to exploit the spare connectivity of graphs, special structures of graph that come in SLAM specific problems, use of advanced spare linear solvers and SIMD instructions for exploiting parallelism. Non-linear least squares optimization for BA can be easily setup in the framework and can be optimized using Gauss-Newton method or Levenberg-Marquardt method. This method was tested in popular SLAM datasets of 2011 and results show that the framework has comparable run time performance with state-of-the-art methods or they were able to beat state-of-the-art methods with no drop in accuracy.

B. factor graphs

In [13], Michael Kaess proposed a method of smoothing where having this whole information actually helps unlike in other cases where it hurts. A solution to this was proposed based on a matrix decomposition based on Cholesky decomposition and later on refined to be a QR decomposition in [28]. [28] proposed an organized way of storing all the constraints in the form of a factor graph to come up with fast inference. An incremental method of updating the solution is also shown, thus providing a very powerful framework for inference. The SLAM problem can be stated as

$$\begin{aligned} X^*, L^* &= argmax_{X,L} P(X, L, U, Z) \\ &= argmin_{X,L} - logP(X, L, U, Z) \end{aligned}$$

where $X*$ and $L*$ stand for the maximum likelihood estimate of the robot pose and landmark locations and U denotes the controls applied and z, the measurements made. Inserting measurement model and motion model, the maximum aposterior estimate leads to the following nonlinear least squares problem with gaussian noise assumption

$$\begin{aligned} X^*, L^* = argmin_{X,L} \{ \Sigma_{i=1}^{M} \|f_i(x_{i-1}, u_i) - x_i\|^2 \\ + \Sigma_{K=1}^{K} \|h_k(x_{ik,l_{jk})-z_k} \ \|^2 \} \end{aligned}$$

where f stands for the state transition function and g stands for the measurement model. If we approximate these non-linear functions by a locally linear model, this problem simplifies to a simple least squares minization problem

$$\theta^* = argmin_\theta \|A\theta - b\|^2$$

where θ stands for all our parameters including the pose and the landmarks and A is a large spare Jacobian matrix that is a linear approximation at the measurement function and the motion model. While this a very standard formulation, finding a solution to the above problem using pseudo inverse is very difficult since the pesudo-inverse solution $A^T A^{-1} A^T$ involves inverting such a large matrix and hence, is not a practical solution. But the matrix A is very sparse and hence, this can be exploited to come up with an alternative solution, which is through QR decomposition. The matrix A can be factorized as

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$$

The solution to the least square problem can be rewritten as

$$\|A\theta - b\|^2 = \|Q \begin{bmatrix} R \\ 0 \end{bmatrix} \theta - b\|^2$$

Since the norm is not changed due to multiplication by an orthogonal matrix Q (The Q matrix in QR decomposition is orthogonal), we can write the same formulation as

$$\begin{aligned} \|A\theta - b\|^2 &= \|Q^T Q \begin{bmatrix} R \\ 0 \end{bmatrix} \theta - Q^T b\|^2 \\ &= \| \begin{bmatrix} R \\ 0 \end{bmatrix} - \begin{bmatrix} d \\ e \end{bmatrix} \|^2 \\ &= \|R\theta - d\|^2 + \|e\|^2 \end{aligned}$$

This system is minimized when $R\theta = d$ and e is the residual of the fitting the given constraints. The real power of such a factorization comes in incremental updates. Suppose that we have the QR factorization of the system with n linear constraints. Suppose that after making a measurement, we get one more constraints. Instead of calculating the A matrix from scratch and computing the whole of QR decomposition fully, we can proceed to add the new constraint w^T to the existing QR factorization.

$$\begin{bmatrix} Q^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} A \\ w^T \end{bmatrix} = \begin{bmatrix} R \\ w^T \end{bmatrix}$$

with new rhs= $\begin{bmatrix} d \\ \gamma \end{bmatrix}$. We have to find an orthogonal matrix R, which removes the last roww^T given in the matrix $A^{'}\}$. This is found by Given's rotation and that rotation matrix is applied to both LHS and RHS to get a matrix similar to the sparse representations obtained earlier and thus leads to incremental and fast update solutions. It must be noted that the same procedure when applied in a scenario where a loop closure is detected leads to the destruction of sparsity structure within the matrices. Linear algebra comes to the rescue as variable reordering is applied to maintain the sparsity structure in the matrix

Until now, we have discussed the whole system in the context of the linear approximation model of the measurement model and motion model. In case of non-linear models, a linear approximation is iteravity applied iteratively on the solution until the change in the solution is minimial. This can be refined slightly for our slam problem since, in slam we make incremental solving of contraints where

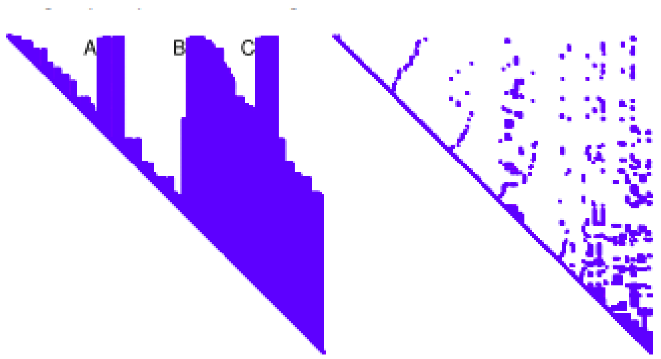


Fig. 16: On the left, The R facor after doing a loop closure and on the right, the same R factor after doing variable reordering.

each new constraint only affects a local part and the initial poses to start with for optimsation are good gueses. Hence, in this context, the relinearisation is only done when variable reordering is done, which is reasonable approximation that saves a lot of time. The interpretation of this QR factorisation as a QR graph is less obvious and this interpretation was clarified through the next work [27].

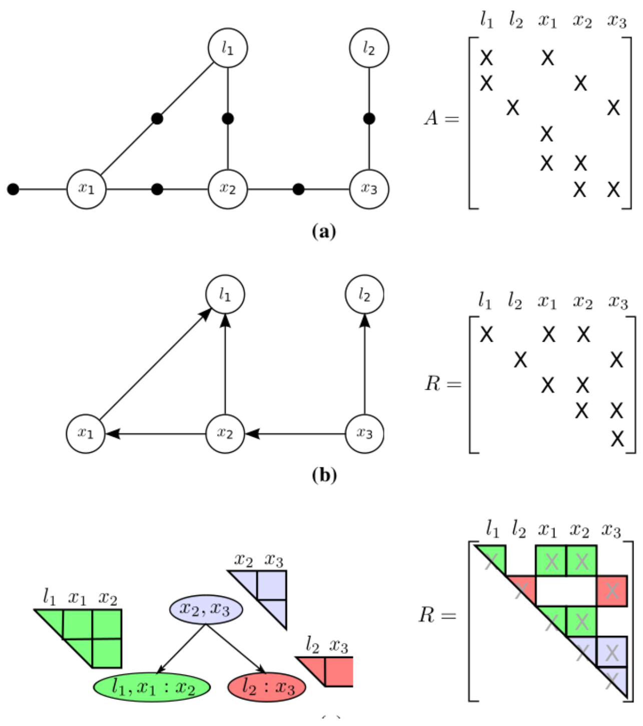


Fig. 17: Interpretation of QR factorisation as a factor graph and transforming a factor graph into a bayessian network

The Jacobian matrix A can be interpreted as a factor graph as shown in the figure. A factor graph is a bipartite graph with two groups of nodes, the variable nodes and the factor nodes. The factor nodes represents the contraints imposed on the system through observations and data association.All parameters to be estimated (pose and landmark) form a variable node in the factor graph. The first row the A jacobian indicate that there is constriaint imposed on the sytem involving l1 and x1. Consequently, this constriant is represented as factor node in the factor graph. Edges are drawn from the factor node to both the parameters involved. The factor nodes are indicated by small black circles in the above figure. Such a factor graph can be drawn by considering all the constraints seen by the system in the A matrix. The same network can be converted to a bayes network for a bayessian inference or it can also be turned into a bayessian tree. A method is proposed, which is analogoues to the QR decomposition in SAM but the same operation defined on the acutal factor graph data structure itself, which is more intuitive. Based on the insights from the new datastrucutre, the auhtors propose a method for handling non-linear equations more efficiently through a process called fluid relineariasation.

VII. Data Association:

There are multiple instances where the need to associate data arises. The use cases are slightly different but the core of the solution used i.e., data association is the same

* **Loop closure:** This problem requirement arises as an idea to correct the drift accumulated in the system. When camera does a loop in a SLAM system, we would expect the camera to return to the starting point at the end of the loop but this is rarely the case in reality due to drift. Thus, if a loop closure were detected we can correct the whole map and camera trajectory to accurately comply with our observations. For successfully detecting loop closure a camera has to constantly keep checking if the current frame being observed has already been observed any where and initiate a loop closure correction if a loop closure was detected.
* **Kidnapped robot:** Kidnapped robot problem is a scenario in which the robot has lost track of its state. This could happen when a vision system is deprived of features when looking at plain wall for example and hence, the robot may moved into an unmapped territory and hence loses track of its state.
* cooperative mapping: Many robots could map an area together and hence the data from all of them need to associated together to create the full map.

Though these problem statements seem different, all of them at their core rely on data association, the ability to a certain image or map being observed has been observed elsewhere before. [49] gave a broad classification of methods in data association and compared their results. The data association can be divided into three broad classes as shown below

* **map to map:** The map matching tries to find common features between two map. A approach like this was first proposed in [4]Though this may

look like the most general approach since it can use both geometry and visual information, its not always the ideal way to go forward.

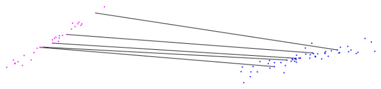


Fig. 18: map to map matching

At a broad level, this method can make use of the geometric similarities between point cloud structure in one map with the other while also utilizing the visual appearance as well. It can seen that Iterative Closest Point Algorithm is a special case of this broad class where the visual information is fully ignored while the geometry information is fully utilized. The effectiveness of this method largely depends on the density of points used in mapping. In a typical landmark based mapping system, only a few mapping points are used and hence, to associate two maps, we should have detected a considerable amount of same points between both frames to detect a loop closure. In sparse mapping setting, this rarely detects a loop closure since there is very low likelihood that the same set of mapping points were generated from the two frames.

∗ image to image: This method relies on computing feature descriptors on each image and estimating the correspondences between the two frames if a large number of key point descriptors are matched. This technique of matching was thoroughly investigated by [6] While this method does work for reasonably well, we should understand that we are throwing away all the geometry information we know about the structure being observed.

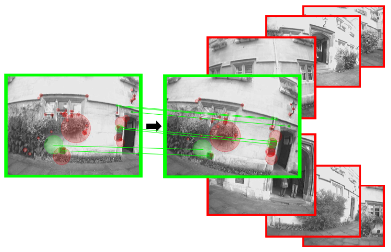


Fig. 19: image to image matching

∗ image to maps This method tries to relocalise the robot's view of the current frame with all the submaps observed in the existing map. [7] proposed one such method for loop closure. To do this, first all the features detected on the image are sent through a classifier, which computes the posterior distribution of the feature for all the other features that have been observed so far. More details on the operation of this classifier are presented later. Thus for each feature in the map, we now have potential map by looking at the class that has the maximum posterior distribution as given out by the classifier. For each of the hypothesis, a camera pose is reinitialized on that part of map. This is done in a RANSAC methodology. First, a three points are picked up at random and a camera pose is estimated based on the points and submap. An error is calculated by measuring the projection error of all the other feature points that is common both the image and sub map being tested.

A critical requirement of data association atleast from a loop closure stand point is the system should have zero false positives. This is a logical requirement since since a wrong loop closure match will

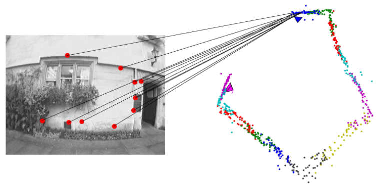


Fig. 20: image to map matching

change the map and camera trajectory in a irrecoverable way. [49] investigated and compared the three algorithms show above and concluded that image to map points are more suited to SLAM application which have spare image points. To quote some numbers from their paper, map to map matching of [4] found 0 percent true positive and zero percent true negatives, which makes it useless. This is primarily due to sparse map points used and that the algorithms under the same category perform really well when dense map are used particularly in case of AR applications. [6] achieved zero percent false negatives with 8 percent true positive while [7] achieved zero percent false negatives with 20 percent true positives indicating its superiority. It also has to be mentioned that some hyper parameters in all of the three works above were purposefully tuned to have zero false negatives in order to comply with requirements of loop closure.

One thing that was left blurry in the previous discussion is how the classifiers are generated in image to map matching described earlier. [31] was the first to propose the idea of building tree for key point recognition. The formulation is such that the recognition problem is posed as a classification problem. A diagram explaining the overall system is shown below

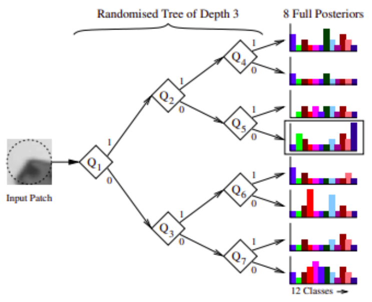


Fig. 21: Random Trees for key point matching

The way the system works is explained as follows. Each node in the tree represents a binary question. In particular, it represents if the pixel intensity at a location a is greater than the pixel intensity at the location b. Thus each image patch which is to be tested goes from the top of the tree to the bottom of the tree going through D binary questions where 2^D is the number of leaves in the tree. Each of 2^D acutally represents a probability distribution across C classes. Thus given a image patch or feature, we will be able to identify distribution of the probability indicating the feature classes it is likely to match. To train such a tree, a feature class to be matched is taken and a lot of augmented data is computed on top of the feature (about 400 patches are generated) and they are then sent down the tree. The probability of the target node in the leaf it reaches is increased so that the distribution at the leaf starts moving towards the target classes it should capture. It can noted that these binary questions are arranged in a tree structure since its assumed that there is some hierarchical relationship between the questions. In scenarios where this is not the case, a sequential line of question is asked as shown in the figure below.

The answer to each question produces a bit (0/1) for the final output string. The answers to each question are concatenated into a D length string (D is the number of questions) and this is used to index a hash table where its target distribution is stored and updated.

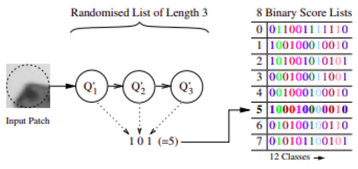


Fig. 22: Random Trees for key point matching

It can seen that the above system is very similar to the system shown above, except that the questions are assumed to be independent in which case all questions are sequentially asked for the image instead of organising the questions hierarchially in a binary tree. A similar classifier system is used for matching a given features against all known features in SLAM system of [7]. for every new feature detected in the new frame, it is passed through the classfier and a posteior distribution indicating the probability distribution of all features it is likely to match is obtained and the top prediction (provided that it is above a threshold) is taken out and it is treated as a likely loop closure candidate for further analysis.

A common choice in many of these works for loop closure identifcation is the DBoW. Inorder to understand DBoW, we should first consider BoW (Bag of Words) [5]. Bag of Words representation were historically inspired by its usuage in document classification, where a document is represented by a histogram of words. To replicate the same process for an image, we must first determine what our dictionary vocabulary is. This is determined through machine learning from a series of training images. First, interest points are detected on an image and desrciptors are built around them. All descriptors from all images are fed into a K-means clustering algorithm where these features are clustered. Each cluster center now represents a vacobulary word for reprsenting the image. Whenever, we have a new image to be reprsented, we detected descriptors throughout the image and associate features to words in our dictionary. We count the histogram of the words (Count the occurence of each words) and this histogram becomes the vector representation for later classification stages. It has to be noted that choice of key point detector and descriptor is important and depends on the problem at hand. The descriptors have to be invariant to all the variations of the problems yet dicriminative enough to classify and should be extractable in a very fast time frame.

A very common methodology used for detecting loop closures in SLAM system is [22]. It builds on the Bag of Words concepts. In order to make computation very fast, binary features are chosen. FAST is the feature detector of choice and BRIEF is the descriptor of choice. A BoW is initially built by training on the targetted application. These words are organised hierarchially into a tree for easy mathching and access. A database is used where in an inverse index is maintained. For a every word that has been encountered in all keyframes, a list of all images in which the word occurs is maintained. A L1 score is used for computing the similariy between two word vectors

$$s(v1, v2) = 1 - \frac{1}{2}\left|\frac{v1}{|v1|} - \frac{v2}{|v2|}\right|$$

Hence, for checking if a given image is a likely loop closing candidate, we build the Bag of words for the image and query the databse to get all the images which have a similar bag of words using the mertric shown above. All s scores are normalised with the best s score we expect to receive.

$$\eta(v_t, v_{t_j}) = \frac{s(v_t, v_{t_j})}{s(v_t, v_t - \Delta t)}$$

Here we approxiate the best score we expect to receive to the score between the word vector for currrent frame and previous frame (which is reasonable). This may fail in rare cases when the robot is turning fast for example. We skip those rare images where such cases occur. A threshold is then used on the η score to determine if the two words vectors match. A another problem to factor is that many sequential frames will match against a given query frame. Thus query frames, which are closly by in time are concatenated together to create an island and an island matching is found. Once island is found, a temporal check is imposed which means that atleast k framees in succession should match the same island to be considered a valid loop closure candidate. As a final check, geometrical consistency is verified. First the image which has best matching score for the current image within the island is selected and correspondences are found between the two images to esimate the fundamental matrix. The estimated fundamental matrix is checked against relative pose between the two frames. In order to facilitate the correspondence estimate problem, a direct index is stored in the database. For each image, for each word at a level l, its ancestors and all the features associated with it are stored so that a given feature can be matched against a very few likely candidates. Since correspondence estimation is a $\theta(n^2)$ problem, this reduces the computation workload into a few constant operations. This a widely used method for detecting loop closure candidates in a SLAM system.

VIII. Matrix Factorization based solutions to SLAM

There are a few works on posing the Visual SLAM problem as a matrix factorization problem. In particular, [47] was the first to point out that such a solution is possible. For this work, it is assumed that we are able to observe the trajectories followed by P points through F frames. Let u_{fp}, v_{fp} be the horizontal and vertical coordinate of each feature point P in the frame F. We can store u and v together in a compact matrix W as follows

$$W = \begin{bmatrix} U \\ V \end{bmatrix}$$

W is a 2F x P matrix called the measurement matrix. The rows of these matrices are zero centered i.e.,

$$\hat{u_{fp}} = u_{fp} - a_f$$

$$\hat{v_{fp}} = v_{fp} - b_f$$

where a_f is the mean of all u_{fp} points and b_f is the mean of all v_{fp} points. Now we can organise these zero-centered points into a measurement matrix

$$\tilde{W} = \begin{bmatrix} \tilde{U} \\ \tilde{V} \end{bmatrix}$$

Now we look at the expression for $\tilde{u_{fp}}$ to simplify this

$$\tilde{u_{fp}} = u_{fp} - a_f$$

$$\tilde{u_{fp}} = i_f^T(s_p - t_f) - \frac{1}{P}\sum_{q=1}^{P} i_f^T(s_q - t_f)$$

The above expression is just the x co-corindate of the point p expreesed in the camera co-ordinate system. This can be simplified as

$$\tilde{u_{fp}} = i_f^T s_p$$

Similarly, for the y co-ordinate the expression can be simplified as

$$\tilde{v_{fp}} = j_f^T s_p$$

Noticing from the above equations, the vector i and j describe the rotation of the camera frame with respect to the world frame while 3D points are described by s_p. The vector form of these equations can be matricized as follows

$$\tilde{W} = RS$$

It must be noticed that R is 2F X 3 matrix and S is 3 x P matrix. Thus, if we were above to find a matrix decomposition to the original W matrix that obeys the constraints imposed by R and S, we would arrive at the solution. The solution to this lies in SVD. We can find the first left and right Eigen vectors along with their corresponding singular values

$$\hat{W} \approx O_1^{'} \Sigma^{'} O_2^{'}$$

This approximation follows from the actual decomposition

$$\hat{W} = O_1^{'} \Sigma^{'} O_2^{'} + O_1^{''} \Sigma^{''} O_2^{''}$$

But we consider all singular values beyond the 3rd singular values to be zeros. Whether this is good assumption depends on the noise level in the system and in particular, a measure of it can be found by the ratio of the 3rd singular value to the 4th singular value which is should be a very high value for this assumption to hold. Furthermore, we can further factorize the above equation to match the R and S dimensions so that we have a factorization solution

$$\tilde{W} = O_1^{'} \Sigma^{1/2} \Sigma^{1/2} O_2^{'}$$

where we can now directly observe that $\tilde{R} = O_1^{'} \Sigma^{1/2}$ and $\tilde{S} = \Sigma^{\frac{1}{2}} O_2^{'}$ But this solution is not unique and is only determined to an affine scale. In general, if Q is any 3 x 3 invertible matrix, a valid decomposition could be

$$\tilde{R} Q Q^{-1} \tilde{S} = \tilde{R}\tilde{S} = \tilde{W}$$

We must impost constraints to make the value of Q unique and the constraints come from the constraints of a rotation matrix. We now that the basis of the rotation matrix have to unit vectors and that there are orthogonal to each other. We can impose these constraints on the system as follows

$$\hat{i_f} Q Q^T \hat{i_f} = 1$$

$$\hat{j_f} Q Q^T \hat{j_f} = 1$$

$$\hat{i_f} Q Q^T \hat{j_f} = 0$$

These are non-linear constraints but the solution to this can be found to find the unique Q matrix which gives the solution. Thus the final rotation and structure matrix can be written as

$$R = \tilde{R}Q$$

$$S = Q^{-1}\tilde{S}$$

IX. Probabilistic filters:

Probabilistic frameworks like bayesian networks have been extensively tested in VSLAM. Most successful of them for a monocular sytem was [9], where an extended Kalman Filter was used to recursively update the map points and camera state together. A follow up paper to the original one was proposed in [12]. The keys ideas in both these papers can be summarised as follows. The state vector for the Kalman filter can be written down as

$$\hat{x} = \begin{bmatrix} \hat{x_v} \\ \hat{y_1} \\ \hat{y_2} \\ . \\ . \end{bmatrix}$$

and the co-variance matrix P is given by

$$P = \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & .. \\ P_{y_1x} & P_{y_1y_2} & P_{y_1y_2} & .. \\ P_{y_2x} & P_{y_2y_1} & P_{y_1y_2} & .. \\ . & . & . & \\ . & . & . & \end{bmatrix}$$

where $\hat{x_v}$ is the state vector of the camera while $\hat{y_i}$ stands for map points. In particular, the state of the camera is represented (using a smooth motion assumption) as follows

$$x_v = \begin{bmatrix} r^W \\ q^{WR} \\ v^W \\ \omega^W \end{bmatrix}$$

where r^w represents the Eucledian position of the camera. q^{WR}, the rotation quaternion and v^W and ω^W represent the linear and angular velocity of the camera. It must be noted here that in the prediction step of the Kalman filter there is no deterministic motion model update since there is no concept of control inputs here. But we are just modeling this as a stochastic process with a smooth motion prior, which states that the camera can transition to the next state with a constant acceleration defined by its prior. It must be noted that a map point is represented here as a semi infinite ray as explained in the key concepts section about inverse depth. But here, the direct depth is modeled instead of the inverse depth but the idea of a representing a point as a semi infinite ray is utilized to facilitate instant integration of the point into the pipeline the moment the points are observed. This is useful because unlike the Cartesian representation of a point as $X, Y, Z)$ where the point cannot be introduced into the map until its depth is determined to a reliable degree. This method of initialization introduces 6 parameters for initializing a point, 5 of which are nearly fixed at the point of detecting the feature. This means that the direction of the semi-infinite, which is characterized by the optical center and the azimuth and elevation angles are fixed while depth is the only uncertain variable that needs to be estimated.The uncertainty for the depth of a point is initialized as uniform distribution with 100 particles but replaced with a gaussian estimate once the depth of the point has been determined to a reliable degree in [9]. In the follow up paper [12] the clarify and clearly explain how the bayessian update for a point with such a parameter is carried. The whole depth range (0.5m to 5m) where a uniform distribution is initialized is split into multiple elliptical regions for searching convenience. These hypotheses are all projected into the new frame when the same feature is observed and the likelihood for each for the elliptical hypothesis is calculated and a bayesian update is made to the probabilities of each of the hypothesis, thereby changing the uniform distribution prior which sequentially converge to a distribution peaked at its true depth location after 2-4 runs where the same feature is observe. Though this works in a confined indoor setting, the method does not extend reliably to outdoor settings. This is very understandable since the core foucs described in [12] is to achieve reliable tracking through very spare features. Hence, map produced are very few points which are very reliable for tracking across view points. This is one of the downsides of using probabilistic framework. The author report that back in 2007, when the paper was published that the highest state vector dimension that the then computational resources could handle was 100. In particular, the authors have a very hard cutoff of 12 points for the number of landmarks stored.The other simplifying assumption made in this work is that the observed surface is locally planar. This is critical due to two reason: The system actually observes a patch of reasonable size (11 x 11) and uses Normalized cross correlation for finding patches. Hence, each patch which is added to 3D map is more than a 3D point. It actually a mini surface. So assuming that the whole patch which was observed is locally planar is indeed bad assumption of the real world but the authors report that the assumption is a good for tracking within probabilistic framework. This is useful in setting when we want to find the relative self position in the given environment but is not very useful when we want to do motion planning since a free space doesn't mean the space is free but it just means that its not being tracked. Since its a monocular system, it relies on a specific bootstrapping so that known objects to the system in the first few frames so the scale ambiguity in the system is resolved. In particular, the system always begins its operation with a rectangular board with four black corner. This known structure at the beginning of the system resolves the scale ambiguity and gives reliable points to track at the beginning of the pipeline. The authors in [12] further go on to show the effectiveness of their system by using it for a variety of applications two systems namely interactive AR and humanoid SLAM.

One aspect that was briefly mentioned in the description of Davison's work on MonoSlam is the construction of surfaces as locally planar surfaces. This assumption greatly aids in matching and mapping. In [33], Davison explains in detail about such a system. Firstly, to clarify how this differs from feature point matching, this is not a single 3D point and hence has a surface associated with its back projection in 3D.

The simplifying assumption placed in his work is that patches are locally planar, which means a surface like a complete sphere would be wrongly captured by the system but for small regions, it is decent assumption. A template view of the planar patch is stored and whenever a matching needs to be found for a patch in another image produced by another camera view, the planar image is warped perspectively according to the estimated transformation between two planes. The result is that we get prediction of how the patch is likely to be imaged in the new view, which greatly helps in finding correspondences. It must be noted that for this simplification to work, the orientation of the plane matter. The orientation of the plane is defined by its surface normal. The initialization of the normal is done such that is parallel to viewing direction (line connecting the optical center with center of the patch). This is definitely not the actual orientation of the plane but the paper proposes a method to sequentially refine the estimate of the surface normal so that the estimate converges to its true normal. This is done by using gradient descent on a cost function which measures the differences between the predicated surface view and the actual surface viewed. In order to

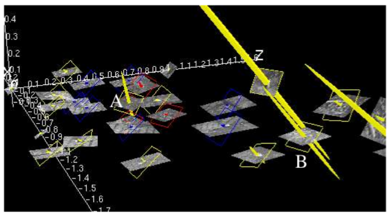


Fig. 23: A map of landmarks generated with the locally planar assumption

accomplish this, two things need to stored: The surface patch (A planar image with its orientation) and the camera state from which that patch is view. Whenever the same patch is expected to viewed from another angle, the planar patch is projected into the initial camera view and homographically transformed to the new view so that a prediction of the warped patch is made. This warped patch will be nearly the same for small movements but not exactly due to incorrect surface normal initialization. Therefore, the surface normal is updated based on this error and this process is continued and the surface patch eventually converges to its true orientation (due to the surface normal update). Once its converged, it becomes easier to find the where a given patch occurs in the image since we can easily calculate a prediction of the warped planar patch, which is likely to be viewed and hence, the final matching of the warped patch with an image can be preformed by simple NCC.

X. Active SLAM:

Active SLAM considers the SLAM from a different angle. It tries to solve an interesting question from an information theoretic standpoint: Which points should be observed next so that the uncertainty in the system is minimized. Davison pursed this line of research for a while and we will discussing some of his works in this section

In [11], Davison explored on a framework for actively determining which measurements of already observed features is near optimal one. In a sense, he devises a strategy, that given N features in map, says which is the best feature to take the next measurement so that the overall uncertainty in map and features is minimized. Davison's approach is hard rooted in sparse features for reliable navigation. He epitomizes this principle by presenting results of navigation with just two landmark features. In order to understand the core of this idea, we must first understand the concept of a predicted measurement. The measurement model for measuring the 3D bearing to a feature point can be stated as

$$h_i = \begin{bmatrix} \alpha_{pi} \\ \alpha_{ei} \\ \alpha_{vi} \end{bmatrix} = \begin{bmatrix} \tan^{-1} \frac{h_{Gix}}{tan^{-1}_{Giz}} \\ \tan^{-1} \frac{h_{Giy}}{tan^{-1}_{Gip}} \\ \tan^{-1} \frac{1}{2\|h_{Gi\|}} \end{bmatrix}$$

where

$$h_{Gi} = \begin{bmatrix} h_{Gix} \\ h_{Giy} \\ h_{Giz} \end{bmatrix} = \begin{bmatrix} os\phi(X_i - x) - sin\phi(Z_i - z) \\ Y_i - H \\ sin\phi(X_i - x) + cos\phi(Z_i - z) \end{bmatrix}$$

The h_{Gi} vector gives the relative position of a landmark in world co-ordinate frame. Here ϕ is the robot heading, x and y the robot's position in world co-ordinate frame. X_i, Y_i, Z_i is the location of a landmark point in world co-ordinate frame. The h_{iG} vector is the point transformed into robot's coordinate frame and h_i is the three bearing angles extracted from the location of landmark in the robot co-ordinate frame. The paper introduces the idea of a innovation matrix to measure the variance of landmark point. The innovation term associated with a landmark is given by

$$S_i = \frac{\partial h_i}{\partial x_v} P_{xx} \frac{\partial h_i}{\partial x_v}^T + \frac{\partial h_i}{\partial x_v} P_{xyi} \frac{\partial h_i}{\partial y_i}^T + \frac{\partial h_i}{\partial y_i} P_{yix} \frac{\partial h_i}{\partial x_v}^T + \frac{\partial h_i}{\partial y_i} P_{yiyi} \frac{\partial h_i}{\partial yi}^T + R \qquad (11)$$

In [10], a thorough analysis from information theoretic standpoint on what is the best feature to observe next is presented. A short description of how this is applied to the work in [11] is presented below. Innovation term in simple terms is measure of the joint variance (or covariance) of the the robot pose and other features with a given feature. It is intuitive from the equation above from the structure of the summation. The joint probability of a pair (or a feature with the robot pose or a feature with itself or pose with itself) is multiplied with the partial derivative of the h_i with the same respective quantities. R is the measurement noise. This idea of a innovation is the key behind choosing which is the near optimal feature to observe next. The volume of the ellipsoid represented by the innovation co-variance is a good measure of uncertainty surrounding a feature point. The volume of the ellipsoid can be written in terms of the eigen values of the innovation matrix as

$$V_s = \frac{4\pi}{3} n_\sigma^3 \sqrt{\lambda_1 \lambda_2 \lambda_3}$$

Thus if we had 5 landmarks in our map, we would measure the V_s values for each of landmarks and observe the landmark with the highest V_s. This is intuitive from a information theoretic standpoint because we gain the most by observing something we are most uncertain about. The most powerful thing about this framework is its update of covariance i.e., observing this highest variance point will automatically decrease the uncertainty associated with other feature points. Let V_{smax} be the maximum of all V_s calculated for all feature points. This V_s tells us something important about the system. If V_{smax} is high, it means that there is a feature point that requires immediate attention, after observing which the overall uncertainty in the system will drastically reduce. If V_{smax} is low, it means that all position and landmark locations are very well determined to a reliable degree. The other point that has to be emphasized is that innovation term is solely calculated based on predicted measurements i.e., based on the robot's belief about its present state and the landmark features, it calculates the information matrix and chooses the highest variance point. This is enough when the robot is stationary but when the robot is moving and the landmarks are constantly shifting. Another cost that also has to be factored in is the time for transition. In setting used in the paper, the robot has a mechanical pan tilt which has to be rotated sometimes to observe a point and this transition time is potentially wasted opportunity time for doing measurements and sharpening our beliefs. Factoring in all this, the paper proposes the following technique for choosing the next point to observe

$*$ Calculate the transition time required for mechanically changing the fixation point for each of the feature point from the present robot state. The paper calls this saccade time. Each measurement is approximated to be of 200 ms. This saccade time is divided by this average measurement time to give an estimate of the measurement that is opportunistically lost in the transition. Lets call this N_i for each feature.

$*$ Calculate N_{max} which is the max of all N_i

$*$ For each feature i, calculate the V_s after making $N_{max} + 1$ steps by performing an immediate saccade. This means that the first N_i steps will be lost in the saccade transition and the remaining $N_{max} - N_i + 1$ steps will be a predicted measurement update re-observing the same point again and again.

$*$ After doing this for every feature point, we choose that saccade which lead to the lowest V_{max}

This procedure is the same as maximizing our information gain by observing the highest variance feature point but this factors in time also. Though this is not the exact optimal solution from a information theoretic standpoint, this is a very near optimal thing to do. The algorithm also maintains map by inserting and deleting landmarks. The heuristic for maintaining points is simple. A landmark point is inserted when there is a sparsity of landmark in the current view of the camera. Based on the observed position of the landmark, a prediction is made about a region of robot movement throughout which the landmark should be visible for it to be considered a good tracking point. Based on their empirical results, this roughly turned out to be 45 degree changes in viewpoint during a robot movement. If a point doesn't meet this requirement it is killed. Thus this algorithm is very liberal in its insertion policy and by having a strong criteria for maintaining a point, a natural selection evolves similar to survival of the fittest strategy.

XI. Use of deep learning in SLAM system

Deep learning based methods have penetrated computer vision and have showed considerate improvements in performance above state of the art methods for several problems. But Visual SLAM is one area where pure deep learning methods fail to match the performance of classical methods. However, some papers have shown that integrating deep learning in a meaningful way with existing classical methods could provide good performance in this filed. [50] showed that a monocular system could beat state of art stereo SLAM system by including deep learning in its work flow. The paper uses stack net, whose architecture is shown below

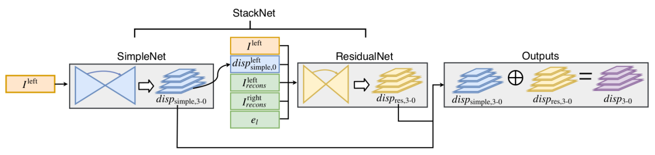


Fig. 24: Over view of stacknet architecture

This network is called stacknet since it has two networks stacked one on top of the other. The first sub-network, which is called the simple net is a encoder-decoder architecture with a ResNet-50 based encoder and skip connections between encoder and decoder. Given an image, the network predicts 4 pairs of disparity images- the left and right disparity images are produced at 3 different scales. The residual network predicts a residual that should be added to the disparity map predicted by simple Net so that higher resolution disparity maps can be obtained. The inputs to a residual network contain the prediction and the errors made the simple net. The 5 inputs fed into the resnet are

- I^{left} which is the (left) image of the stereo pair fed into the network. In inference time, of the algorithm this is just the image from the monocular camera.
- $disp^{left}_{simple,0}$ is the left disparity image predicted at the lowest scale
- I^{right}_{recons} is the reconstructed right image by applying the right disparity map on the left image
- I^{left}_{recons} is the reconstructed left image by back-warping I^{right}_{recons} using left disparity map
- e_1 is the L-1 reconstruction error between I^{left} and I^{left}_{recons}

One key advantage of this system is that it only requires stereo pairs of images for training and avoids the expensive LIDAR or depth camera based data collection process. The loss function used for training the network is shown below

$$\begin{aligned} L_s =&\alpha_U(L_U^{left} + L_U^{right}) + \alpha_S(L_S^{left} + L_S^{right}) + \\ &\alpha_{lr}(Llr^{left} + L_{lr}^{right}) + \alpha_{smooth}(L_{smooth}^{left} + L_{smooth}^{right}) + \\ &\alpha_{occ}(L_{occ}^{left} + L_{occ}^{right}) \end{aligned}$$

The various terms in this loss function are explained as follows. Ł$_U$ is the self-supervised loss the measures the quality of the reconstruction. This loss term checks how consistent is the left and right image under the given stereo image for the predicted depth map.Ł$_S$ is the supervised loss that measures the deviation of the predicted depth from the ground truth depth. Ground is generated for a few points using the static stereo, with which the data is collected and it is for these few spare points that this loss function imposes a constraint. L_{lr}^{left} verifies one image along with its corresponding disparity produces the corresponding stereo pair (ie., left image with disparity produces right and right image with disparity produces left). L_{smooth} constraints the depth predictions to be locally smooth. The effect of L_{smooth} is that the depth maps are smoothened even during transition from foreground to background and vice-versa. This term penalizes the absolute sum of disparities which avoids this problem of over smoothing at object boundaries.

Eventually, once the network is trained, it will product a disparity map for the given image as per the trained stereo baseline setting. One straight forward way to use this information would be to use it in initialized depth when a frame or structure is detected for the first time. This overcomes the well known scaling issue in monocular systems and prevents scale drift in the system. But the paper goes beyond this straight forward idea and models a virtual stereo image. The idea is to use the network to generate the disparity image. Now a right image corresponding to the same view under the stereo setting with which the network was trained was trained in. Now this virtual image is treated as if this were a real stereo image and the whole bundle adjustment optimization is carried very similar to [**?**] but with one subtle change. Since the prediction from the network will turn to noisy, the optimization is not applied on all of pixel but only on a subset of pixels. In particular,

$$e_{lr} = \left| D^{L}(p) - D^{R}(p^{'})a \right|$$

just reprsents the difference in disparity estimates produced by the left and right images. If $e_{lr} > 1$, then pixels are not selected for all the other pixels for which the criterion is satisfied. Finally the cost function to be optimized is very similar to Stereo DSO. Since, we have already discussed in detail about the stereo DSO cost function, we directly jump to the final formulation here

$$E_{photo} := \Sigma_{i \in F} \Sigma_{p \in P_i} \lambda E_i^{+p} + \Sigma_{j \in obsp E_{ij}^{p}}$$

This cost function is nothing but the photometric erros between points observed in frames in the temporal stereo and our virtual static stereo. For each frame i and for every key point in the frame, we calculate the photometric reprojection error between the image and its corresponding stero image generated by the deep network to simulate a virtual stereo. This is indicated by the cost function E_i^{+p}. Similarly, for every frame i and every key point p in the frame, its photometric reprojection error is minized in all the other frames that key point was observed. The term λ represents the weighting to trust between static stereo and temporal stereo. Back at the time of publication, this method achieved the top results in various stereo SLAM benchmarks.

It worth discussing how neural network are trained to predict depth. Prior to the work of [50], several network architectures were proposed to predict the depth from a mono-ocular image. The ground truth for them were collected using various means like LIDAR and depth sensor. [24] proposed an easy mechanism to train such network with stereo pairs. The data for training just consists of a stero pair of image and this constitutes a self-supervised training. The architecture of this network is shown below

This is much simpler version of the depth prediction network used in [50] and simply enforces a left right disparity consistency in addition to two other losses. There are three loss functions that help in training the network to state of the art results. The losses are C_{ap}, which is the appearance cost which penalizes dissimilarity between image reconstructions, C_{ds} which promotes smooths in disparity map prediction

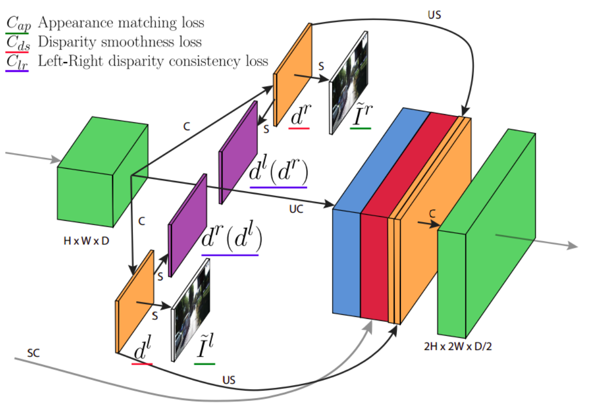

C_{ap} Appearance matching loss
C_{ds} Disparity smoothness loss
C_{lr} Left-Right disparity consistency loss
H x W x D
2H x 2W x D/2
d^r
$\tilde{I}^r$
$d^l(d^r)$
$d^r(d^l)$
d^l
$\tilde{I}^l$
US
UC
C
S
SC

Fig. 25: Over view of network Architecture

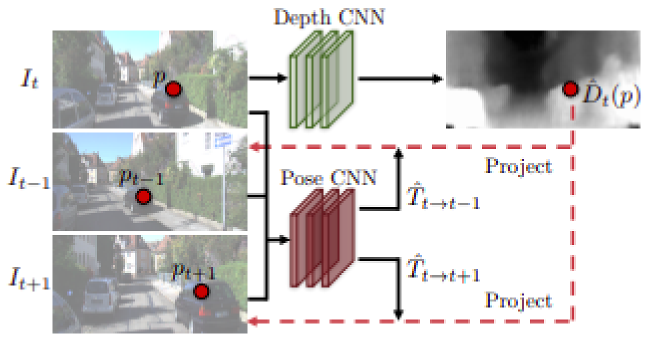


Fig. 26: An end to end deep learning based methods for SLAM

and finally c_{rl}, which is the left right consistency and enforces the disparity maps generated for both images of the stereo pair to be consistent i.e., if the both input images are rectified, the per pixel disparity predicted for the right image should produce the left image and the per-pixel disparity predicted for the left image should give the right image if the disparities are applied. Once we have disparities, we can get the depth trivially from the system as given below $\hat{d} = bf/d$ where $\hat{d}$ is the depth predicted for a pixel, f is the focal length and b is the baseline distance.

End to end deep learning methods are exist for SLAM problems but they perform very poorly. [51] was one of the system to propose to deep learning model for predicting both depth and ego-motion. A simplified block diagram of the system is shown below

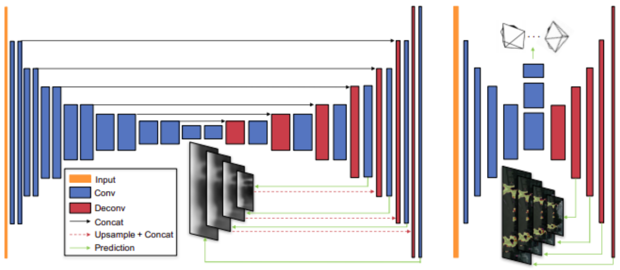
The images are fed through a depth prediction network which predicts the depth for all pixels. The depth is fed to the next image in the sequence and this combination of depth map and next image is fed through a pose prediction network. The key sight in this paper is the formulation of cost function to make training this network feasible. By using the depth image, the previous image is back projected to create the 3D point cloud and this point cloud is then reprojected into the camera frame in the predicted pose. A photometeric loss measuring the difference between the projected image and the actual image is the main mechanism for training this network. This loss is called view synthesis loss and can be mathematically as follows

$$L_{vs} = \Sigma_s \Sigma_p \left| I_t(p) - \hat{I}_s(p) \right|$$

Let ¡$I_1, I_2, ...I_N$ denote each image in image sequence given through the video. Let I_t be the target image to which all the other image will be projected to for checking photo consistency for the depth and the pose estimated. This means that s in I_s can take any value between 0 and N but s!=t. The view synthesis cost function just takes every image in the sequence and uses its depth map to find the projection of the point cloud to the target view to find the photometric error. This view synthesis loss is applied across different scales of the depth map and few other terms are added to the loss function to account for smoothness of depth maps and to overcome modeling limitations. Putting everything together, the final loss function for training the network becomes

$$L_{final} = \Sigma L_{vs}^{l} + \lambda_s L_{smooth}^{l} + \lambda_c \Sigma_s L_{reg}(\hat{E}_s^{l})$$

Here, l loops over different image scales and s loops over different source images. $_s$ and λ_c stands for relative weighting between different terms. As explained earlier, the smoothness terms just imposes a smoothness in depth prediction in the depth map generated by network and regularization term is added to overcome modelling limitation. There are scenarios where the model is not designed to handle like moving objects, non-rigid objects in scence. This is eliminated by using explainability mask where the network can choose to ignore parts of image in computation to avoid these problems. A quick peek into the network is shown below The first few



(a) Single-view depth network (b) Pose/explainability network

convolutional layers are shared by all three branches depth prediction, pose prediction and explainability mask. It must be noted that all of these branches can be independently trained or tested by setting lambda value. Choosing lambda values to be zero cuts off that part of the branch. The paper goes on to show that the model exhibits comparable performance to ORB slam but the system is tested only by running a 5 image sequence. It is understand that the system will start to perform poorly thereafter due to accumulated drift. Since there is no scope for loop closure frameworks like these where end to end deep learning is performed, these methods to this day cannot compete with classical techniques. If deep learning has a scope in SLAM, its in a hybrid approaches that utilize deep learning predictions in SLAM classical optimization pipelines like [50]

XII. FAST SLAM

Among the literature of probabilistic filters, one approach that has garnered widespread attention for is FAST SLAM (FACTORISED SLAM) proposed in [34]. The key insight in building the algorithm is to factorize the joint probability distribution of locations and map points into two separate distributions one for modeling the distribution of robot states and the other for modeling the conditional distribution of map points given the robot state. Conditional independence is further assummed among different landmark points to simply the structure and reduce the computation overload. An efficient tree data structure is further proposed to optimize computations.

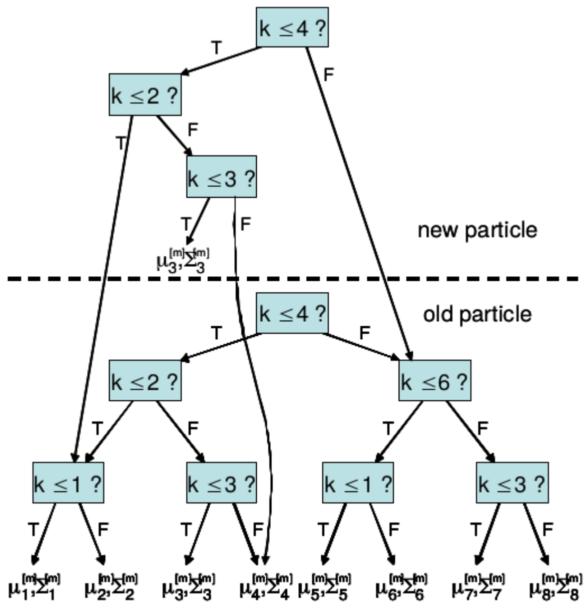
$$p(s^t, \theta \left| z^t, u^t, n^t \right) = p(s^t|z^t, u^t, n^t)\pi_k p(\theta_k \left| s^t, z^t, u^t, n^t \right)$$

Here s denotes the robot states and s^t denotes all sates till t $(s_1, s_2....s_t$, hence the entire robot trajectory till now. z^t indicates all measurements made till time step t, u^t refers to all controls applied till time t. n^t denotes the indices of all landmarks observed till the current time step. This signifies the correspondence problem in matching landmarks. By having an estimate of n_k at time step k, we mean that we have an internal estimate which landmark was observed. Another key assumption that has been very subtly stated in the above formulation is the landmarks are conditionally independent i.e., the landmark observed at time step k θ_k is conditionally independent of the landmark observed at another time step m θ_m. After making such a factorisation, the work is implemented by using a particle filter for building the state estimate $p(s^t|z^t, u^t, n^t)$ and a EKF filter for estimating the conditional distribution of landmarks given present robot state $\theta_k\,|s^t, z^t, u^t, n^t$. These modeling choices make this a very powerful framework to operate since these updates can be very fast. First, the use of particle filters for state estimation doesn't impose too much computation burden since the state is only a 3 DoF system in case of a mobile robot or 6 DoF system in case of flying robots like quadcoptor. In other words, it can be stated as justified and intelligent use of the powerful modeling of particle filters. But landmarks are built as EKF filters conditioned on the state of the robot. This means that if there were K particles in the particle filter estimate of state and M landmarks totally, there would be a total of KM EKF filters for estimating landmarks. The conditional independence of landmark observations means that the covariance of a landmark is only a 2 x 2 matrix, which makes computations faster. In normal probabilistic frameworks, the covariance estimate becomes a bottleneck. If there were K landmarks, we would have K^2 elements to update for every iteration and hence the complexity is O(n^2), meaning normal probabilistic filters are quadratic in terms of the number of landmarks. But due to the 2 * 2 covariance in FAST SLAM, the algorithm is only linear in the number of landmarks. Thus the complexity of running one step should be O(NK) where N is the number of particles and K is the number of landmarks. This the naive implementation and the paper goes on to show how this can be done in O(N log_2K). The key insight to build this solution is that when the robot transitions to a new state S_t from an old state S_{t-1}, and makes a landmark observation θ_t which associated with the index n_k, only the EKF estimation of that landmark should change and all the other landmarks can remain the same. To state this idea concisely, when the robot changes to a new state S_t from S_{t-1}, there are only two things that change

* The new S_t will poses a different value than the previous state estimate s_{t-1}
* The gaussian with index index n_t should be updated since it was the landmark that was observed

All the other gaussians will remain the same. Instead of copying all landmarks from the previous particle to the new particle naively, we could organize all gaussians corresponding to all landmarks in binary tree as shown below

It must be noted that in the data structure above all leaves nodes corresponds to a landmark estimate and that we can access any landmark in O($log_2 K$) where K is the number of landmarks (the number of leaves in the tree). When a new particle is created corresponding to a state transition, an incomplete tree



k ≤4 ?
T
F
k ≤2 ?
k ≤3 ?
new particle
old particle
k ≤6 ?
k ≤1 ?
k ≤3 ?
$\mu_1^{[m]},\Sigma_1^{[m]}$ $\mu_2^{[m]},\Sigma_2^{[m]}$ $\mu_3^{[m]},\Sigma_3^{[m]}$ $\mu_4^{[m]},\Sigma_4^{[m]}$ $\mu_5^{[m]},\Sigma_5^{[m]}$ $\mu_6^{[m]},\Sigma_6^{[m]}$ $\mu_7^{[m]},\Sigma_7^{[m]}$ $\mu_8^{[m]},\Sigma_8^{[m]}$

Fig. 27: A binary search tree data structure for storing the landmark estimates

is first created and only the gaussian index k which was observed is stored in the tree (In the example shown above k=3). But all the other tree nodes are just completed by copying the corresponding node pointers from the old tree. In this way, the tree can be constructed in O($log_2 K$). Since such an update needs to carried for each particle, a single update step can take O($N log_2 K$), where N is the number of particles.Using this approaches, the authors showed that they can track as many as 50,000 particles.

One of the problems with FAST slam is the well known problem with particle filters - particle deprivation and this was handled in the next version of FAST SLAM [35] This is especially true when having a very noisy motion model or prediction step. Since the measurements are primarily accounted through the resampling process, many of the particles might get killed due to the noisy motion model. If the gaussian co-variance of associated position is high but we have a very peaked distribution in measurements, the resampling just kills many particles and this results in particle deprivation, when a single particle or a few particle dominates the whole distribution and hence, destroys the point of having a particle filter, which is to represent multiple hypothesis. This work proposes a framework to compute $P(s_t|s^{t-1,[m]}, u^t, z^t, n^t)$. The key difference in the above equation is the addition of z_t into the state estimation. This therefore leads to better results. Since we like to estimate s_t by incorporating the most recent measurements, it makes sense to factor in our observed landmarks into our calculation.

$$p(s_t|s^{t-1,[m]}, u^t, z^t, n^t)$$
$$= \eta^{[m]} \int p(z_t|\theta_{n_t}, s_t, n_t) p(\theta_{n_t}|s^{t-1,[m]}, z^{t-1}, n^{t-1}) d\theta_{n_t}$$
$$p(s_t|s_{t-1}, u_t)$$

The above equation allows us to factorize what we need to estimate into the just the product of two well known terms: the measurement probability and the state transition or motion model probability. The only difference here is that the measurement probability is marginalized over the distribution describing the current landmark being observed. It must be noted here that the measurement model can involve a non-linear function in its more general version and hence, the paper goes on local linear approximation to compute the term $\int p(z_t|\theta_{n_t}, s_t, n_t) p(\theta_{n_t}|s^{t-1,[m]}, z^{t-1}, n^{t-1}) d\theta_{n_t}$

This allows us to estimate our particle distribution by factoring in both measurement and state transition. One final step is the importance weighting. It can be argued that the whole purpose of the mehtod was to remove this importance weighting from the system but in the process of the doing so, we are not able to accurately estimate the distribution. The root cause of why this occurs is the normalizer $\eta^{[m]}$ used in the estimation equations which turns out to be different for different particles. Hence, a reweighting process is undertaken to negate this effect and thus, we have a better distribution of particles which incorporates measurements better into the next state particles prediction than what was proposed in the original FAST SLAM.

XIII. A brief look at some other well known SLAM systems

A. PTAM:

[29] proposed to organize the computations involved in SLAM into two parallel threads - a tracking and a mapping thread. This kind of splitting allows the system to use the more expensive bundle adjustment techniques for camera pose estimation and 3D reconstruction, which is not typically used due to real time constraints. The work can be summarised as follows

* Tracking and mapping are initialized in parallel threads
* The tracking thread estimates the camera pose for every new key frame based on a subset of common points between the new key frame and the previous key frames. This is done first by coarse calculation over a broad search space using a few map points (about 50). Once a coarse estimate is obtained, a finer estimate of the pose is obtained by doing a search over a narrow search space using a lot more points (about 1000's of points).
* The mapping thread then makes the map more dense by projecting common points visible in the camera views which were not constructed by the tracking thread.
* The initial mapping when the camera starts up is obtained using a stereo pair.
* New points between consecutive frames are calculated using epi polar search

B. kimera

A team from MIT spark lab CoStar won the DARPA challange this year and one of the most recent papers from them [43](to be published in ICRA 2020) was analyzed to understand the techniques they have integrated to make this happen. Their work Kimera is open sourced as a framework. A brief overview of the whole system is given below in the following block diagram

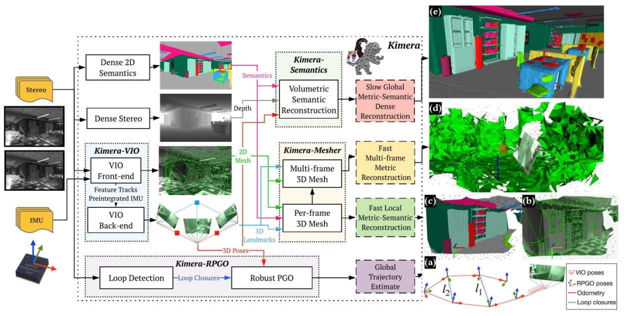


Fig. 28: A brief overview of kimera system

At its core, Kimera is stereo inertial system. A VIO system that uses stereo and inertial sensors produces a 2D mesh and a VIO back end optimises the back-end, produces the 3D poses of the camera. The stereo thread process the stereo image pair and produces a dense depth map. A pose graph optimization is used to find the optimal camera poses for the observed measurements. Then the poses and the depth are fused together to create a volumetric reconstruction (A voxel based TSDF). This reconstruction is also fused with semantic information information from a deep neural network to produce a metric-semantic dense map. Depth information is integrated into the 2D meshes to construct a 3D Mesh. A per frame 3D mesh is generated and another module integrates the 3D Meshes across frames to generate multi-frame mesh. The per frame and multi-frame meshes are very useful for robot navigation while the metric-semantic map is useful scene understanding and promoting robot interaction. A separate thread also estimates the global trajectory by using camera poses and detecting loop closure. A DBoW system is used for detecting loop closures.

A good slam solution will consider the whole of trajectory of the camera for both map building and pose estimation and this is the actual objective that reflected in the bundle adjustment cost function.

C. Semantic SLAM / Object level SLAM:

One promising research direction that has gained attraction in the recent years is the synthesis and utilization of semantic information in SLAM system. [44] is one such work which operates at object level. A concise overview of the system is shown below

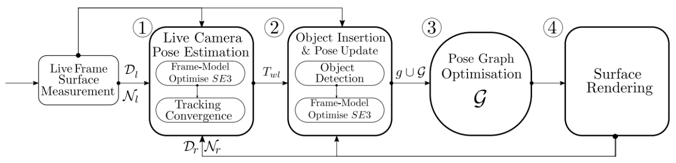


Fig. 29: Object SLAM

This system fully runs on GPU and runs at the object level. Objects are detected and placed into the map and these object serve as features for loop closure or camera pose refinement in the system. The system builds on the premise of a known object stored in a database. Each object is carefully scanned using Kinect and manually segmented. The objects are described by all point pair features, which describes the relative angles and distance between all points. These features are discretized and placed into Hash table. Newly detected objected are matched against the objects in the dataset to see if a match exists. These features cast vote for the object pose in a style that is similar to generalized Hough transform. Since the objects are constrained to be placed on the ground, the voting happens in the 3 DoF of x y positions and the yaw angle. Since we known a rough initial guess to x y positions of the object, the algorithm doesn't have to search brute force across the entire space but only in a known local space. The paper discusses efficient GPU implementation that makes this feasible in a very short time. It must be noted that objection segmentation is an easier problem here since we have the depth information and segmentation in 3D space is essentially a connected component analysis in 3D space. But this hough transform like search only leads to very coarse solutions and hence, finally the pose of the object is refined by ICP that uses point to plane metric. This is slightly different to the ICP algorithms that use a point to point distance. In a ICP that uses point to plane distance, the object is to minimize the distance between a point in the source surface to the tangent plane located in the destination surface at the point of correspondence. This metric is superior and gives better alignment results. Once the object is localize, ICP is used again, this time to localise the camera pose assuming the world to be static. The paper proposes to use an active search technique for localising object. The active search technique at its core, is about chanelling the search towards unexplored or unknown map spaces. A mask is generated in pixel space corresponding to those depth maps which are yet to be associated with an object and an object matching is carried out in these regions to efficiently search for new object. The whole system is constructed as a pose graph optimization problem where the object serve as the reference measurement landmark and the camera poses and the map points are optimized through a Bundle adjustment cost function. The system also has relocalization capabilities. When the system is lost, it starts mapping again and when it finds at least 3 objects, the new graph is matched to the existing graph to find the actual orientation of the camera.

D. *SLAM for driver less CARS:*

[42] analyzed the state of SLAM to be used in Driver less cars back in 2012. Some of the recommendations from the work were the need to have a topological maps for longer distance and a metric representation for shorter/local distances. This comes in light of the fact that self-driving cars need to keep tracks of 1000s kilometers and it simply wouldn't possible to estimate errors over such large maps and hence a map in which places which are far apart are topologically connected with no notion of metric distance could be very useful while at the same time metric distance based graph represented can be used for refining pose and map in the local scale. This is step toward life-long SLAM, where all cars could potentially collect data, share with each other and optimize it for navigation. The more expensive features that we use, the better the results in terms of accuracy. A planar feature though is more costly to estimate gives better results than a corner based feature at the expense of higher cost in computation. Therefore, point based features are normally used due to this tradeoff. Marginalization or Windowed optimization techniques provide a practical tradeoff to storage vs accuracy tradeoff. But it has to be mentioned that these system typically don't reach the accuracy that global estimation techniques reach. But global estimation techniques slow down after accumulating a lot of key frames and points. Therefore, this is also a place where a tradeoff is made depending on application.

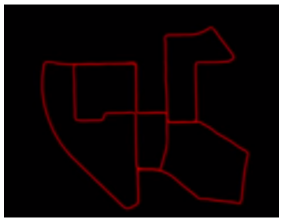
XIV. Summary and Comparison

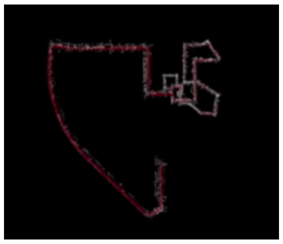
SLAM as an area is very broad and there are a lot of competing methodologies and sensor suites. What is the best SLAM method often depends on the task on hand. As SLAM continues to build on, there seems to emerging split between the AR and the robotics community. They are driven by two different objectives and hence, tend to stress more on slightly areas. Some of the primary points of distinction between the two communities are

∗ **Information management:** Information management seems to be a big concern for robotics community while the AR community is not too bothered about it. This is understandable since AR applications want to map an indoor room at the maximum. This is exactly the opposite in the robotics community now, where ideas are springing up about topological maps since there is need to maintain a map for 100s or 1000s of kilometers in case of self-driving cars.

∗ **Hardware:** The robotics community is slowly moving away from GPU and focuses more on SLAM algorithms that can be run on board in realtime. This is understandable from the demands of the robotics system. A GPU imposes a lot of stress on the robot design since it requires a cooling system and drains a lot of power. One subsection of robotics, that is very sensitive to payload being carried is the quadcoptor group since the payload carrying capacity is not as much of other robots like mobile robots. The AR community driven, which is not driven by these factors is deeply rooted in GPU.

∗ **Surface representation / Density of Maps:** Robotics community is not concerned of the quality of the reconstructed surface since the maps are a tool for robot motion planning, navigation or interaction whereas with a AR system, the quality of reconstruction is of paramount importance since its one of the most important factors in user experience. Robotics, community in fact, is happy to operate with a few feature points instead of map and the act of representing reconstruction as surfaces or meshes is relatively not deeply looked in robotics. But there is a diverging interest in the density of the area mapped in the robotics community. For a robotics navigation problem, a dense map building using SLAM is an overkill since it can be done with sparser maps more efficiently and reliably. But there are robotics applications where mapping becomes important as well, where dense maps become a necessity.


(a) Ground truth trajectory of a map


(b) A trajectory estimated by monocular system

We have looked at different sensor suites that can be used for visual SLAM. It is good to remember what are the characterisitcs and challenging when using each of them

∗ **Monocular SLAM:** While monocular SLAM is the well studied problem that purely relies on temporal stereo for structure and motion recovery, it inevitably suffers from scale drift.

The figure above shows an illustrative example for this, where the geometry of the reconstruction seems okay but the scale keeps shrinking as the trajectory progress ultimately making the map useless unless there is some correction from the loop closure to correct for the scale drift. This the reason why monocular systems often have a very specific bootstrapping sequence so that the scale is initialized to the right value but this does not eliminate the problem completely as drift can still occur. It is quite interesting that one of the works by Daniel Cremer [50] was able to exploit deep learning to simulate a virutal stereo that ended up beating many of the state of the art algorithms in a stereo system. With the infusion of deep learning, it will be interesting to see how the work on Monocular SLAM evolves.

∗ **Stereo SLAM:** Stereo slams avoid the scale drifting by finding out the depth of a few points using the static stereo constraints. But there is no free lunch as this comes with increased computation need though the cost of an additional camera is no longer a concern. It can also be noted that this system doesn't require any kind of bootstrapping to begin with since the depth of the points can be estimated directly.

∗ **Visual Inertial SLAM:** The IMU eliminate several problem in a slam system with its rich information. Firstly the scale drift is avoided in the inertial system as well due to infusion of motion estimate predicted from the IMU. But it has to be noted that this system also has to go through an initial bootstrapping sequence similar to Monocular system since the IMU parameters are initially uncalibrated at the start of the operation and the only way to initialize a point with a known depth is to follow the standard 8 point or 5 point algorithm. One more advantage of using IMU is that the pose optimization essentially reduces to a 4 DoF optimization problem since there is no drift in the roll and pitch angles. But its advantage is offset by its own sensor structure since we have to pass in the sensor noise parameters as optimization parameters. As per the inference from literature, it can be argued that the sensor noise characterization is much simpler optimization problem in a stereo SLAM or monocular SLAM since those extra 2 DoF involved in the full bundle adjustment brings in a lot more parameters to be estimated, two extra for each trajectory point than a VI system. Thus on paper, a visual inertial system should be better than a setero or Monocular system but the IMU costs are more expensive than a camera.

∗ **RGBD:** Direct depth measurement is always better than inferred depth and also allow for depth predictions to be available for every pixel in the image. Thus, RGBD sensor should be better than a stereo system. But the depth sensor measurements are only to a very limited range often only a few meters in the case of Kinect. Thus, there are good for small indoor applications where objects are pretty close but for outdoor applications where the range is much larger, the system fails and stereo wins in that aspect.

Filtering Vs Optimization: The increasing trend in the visual SLAM community is to move towards Optimization based methods. The most agreed upon method seems to be to estimate a motion only Bundle adjustment and structure only bundle adjustment locally and then do a global optimization if that seems reasonable. But there are works that have increased the power of filtering based approaches like fast SLAM [34], which can handle as much as 50,000 landmarks on an average computing platform. But at the core this debate is the objective of the work undertaken in question. If robotic navigation is the objective in question, filtering based approaches are still a powerful choice. But they fall apart if the need is to generate dense maps or to do this navigation over a large area, where the run time grows quadratically with the number of parameters in case of normal filters. A fast SLAM can do this

$O(nlog_2 K)$

Direct vs feature based methods: Direct methods appear to be way more superior at first sight but they have their own shortcomings too. Direct methods rely very much on photo consistency and small baseline assumption. This is okay as long as we are using a high frame rate camera and we are operating in environments where sudden lighting changes are unexpected. Though these methods seem to be showing good results, its still open to discussion if direct methods can perform as robustly as feature based methods under varying lighting.

Though it is very difficult to compare different SLAM algorithms under a unifying framework, a final analysis on comparison of different SLAM on three different dataset results is discussed. [2] compared three SLAM techniques on outdoor natural images. The objective was to find out how standard SLAM methods perform in outdoor environment. The whole testing was carried on a dataset collected by traveling through a lake on the boat. The algorithms compared by this method are: ORB SLAM2, LSD-SLAM and DSO. A brief summary of some of the compiled statistics is shown below

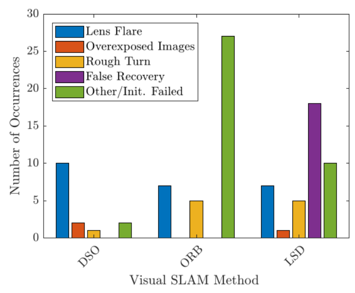


Fig. 31: Visual SLAM Method Comparison

The paper reports that ORB SLAM2 suffer from initialization problems in natural environments. Another important aspect is that the LSD SLAM seems to prone to a large number of false recoveries. Overall on failure modes, DSO seems to fare a lot better making it the most robust among the three.Another parameter that was considered for the robustness quantification was the completion rate

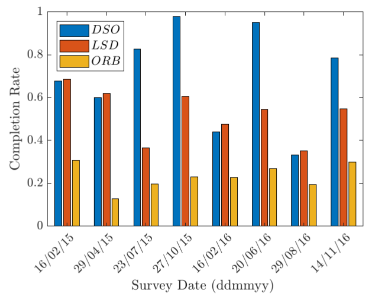


Fig. 32: Completion rates for Visual SLAM

Completion rate refers to the percentage of times the SLAM system generated the entire loop without going into a failure mode. It can once again be seen that DSO had a significantly higher completion percentage, followed by LSD and then ORB

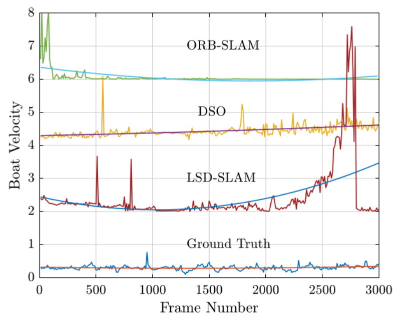


Fig. 33: Drift accumulated in the system

These velocity profiles don't have any units. Its just the initial values that the systems were initialiazed with during the bootstrapping sequence. These absolute numbers do not hold any meaning on their own rather they are just used for viewing the drfit in the system. It can be seen that ORB exhibits the least amount of drift, where as DSO exhibits a slight positive drift and LSD has a more positive drift.

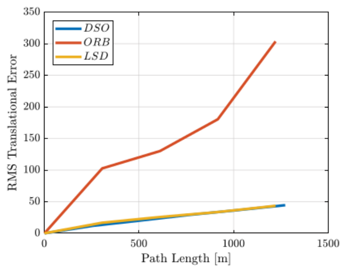


Fig. 34: RMSE error for comparing the three algorithms

RMSE is an error metric where the estimated trajectory and the ground truth trajectory are aligned and a sum over the pair wise distance between each point in the trajectory is estimated. It can be seen that LSD and DSO exhibited very low RMSE as compared to ORB, which exhibited a large error. Though the exact numbers vaired depending on the weather conditions on the day, the trend in the data didn't change drastically.

An evaluation of the same three alogrithms but in an indoor setting was carried out by [17]. The results of the analysis are shown below.

System	RMSE (m)	Mean (m)	Median (m)	Std. (m)	Min (m)	Max (m)
Cartographer	0.024	0.017	0.013	0.021	0.001	0.07
LSD SLAM	0.301	0.277	0.262	0.117	0.08	0.553
ORB SLAM (mono)	0.166	0.159	0.164	0.047	0.047	0.257
DSO	0.459	0.403	0.419	0.219	0.007	0.764
ZEDfu	0.726	0.631	0.692	0.358	0.002	1.323
RTAB map	0.163	0.138	0.110	0.085	0.004	0.349
ORB SLAM (stereo)	0.190	0.151	0.102	0.115	0.004	0.414
S-PTAM (no loop cl.)	0.338	0.268	0.244	0.206	0.001	0.768
S-PTAM (loop cl.)	0.295	0.257	0.242	0.145	0.006	1.119

Fig. 35: RMSE error for many indoor algorithms in an indoor setting

It can be seen that ORB SLAM has a lower RSME error than LSD SLAM and DSO SLAM, this proving that ORB is more reliable and robust indoors. An interesting aspect from the comparison is that ORB stereo achieves higher error than plain ORB, which is counter intuitive to what we expect. This indicates that there is scope for improvement there. It also highlights that sometimes too much information to optimise over actually results in degraded performance.

Another comparison of LSD slam with ORB slam was carried out in [16]

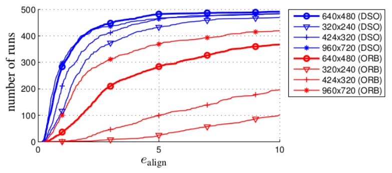


Fig. 36: Effect of changing the rectified image size

It can be seen from the figure above that LSD slam achives higher alignment error with increasing runs but one thing that can also be noticed is that LSD slam is unaffected by rectified image size. The trend of curve nearly remains the same irrepstive of recitified image size showing that LSD slam is more robust to change in rectified image sizes

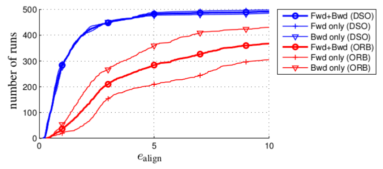


Fig. 37: Dataset motion bias

The paper also analyses the dataset bias on motion estimation for both methods. Both methods LSD slam and ORB slam were run through all sequences both forwards and backwards and both. It can be seen that ORB slma has varying alignment errors for different sequences played while LSD slam is not affected by the direction in which the sequence is played proving that LSD slam has lesser dataset bias

XV. CONCLUSION AND FUTURE WORK:

[1] presents detailed survey on slam and lists some of the open challenges in the community. Some of the challenges listed in the paper are

- **Robustness: Fail safe SLAM and recovery** Many of the SLAM techniques in action today are very susceptible to outliers. This is mainly due to the fact that most of the SLAM solvers are iterative solvers and they rely on the key idea od linearization around a key point, which fails in the case of an outlier point. An ideal slam should be fail safe and faile-aware i.e., the system should be aware of imminent failure modes and initiate a recovery rather than entering into irrecoverable failure modes.
- **Robustness: Metric relocalization:** Present loop closure techniques pure based on visual features cannot detect all loop closures reliably across day and night and across different seasons.
- **Robustness: Dynamic and deformable maps:** Most SLAM systems are built on the key idea of a static scene and rigid body assumption about objects. SLAM systems need to go beyond these assumptions by considering dynamic scenes and soft deformable objects.
- **Scalability: Map representation:** For a long term exploration, our present methods data representation such as point clouds or meshes won't scale up. We need to come up with a compact and compressed map representation for long term operation
- **Scalability: Learning, forgetting and remembering:** Another unexplored question is to decide when an information is outdated and when a information can be saved and reloaded. Present approaches just stores all data and marginisalise out data when the allocated memory is filled up.
- **Scalability:Disributed mapping:** Outlier rejections become even more difficult to handle in distributed system since the data is measured in different robot frames before fusing in into a single frame. The effect of outliers is catastrophic but this has not been cleanly solved.
- **Metric map models:semantic models** A compact way to store map would be to derive high level semantics rather than using points and surface. But not a lot has been invested into the semantic understanding of structures within a SLAM system.
- **Metric map models:Optimal representation** While we do have a lot of techniques to formulate a geometry we desire, what is not well understood is the comparison of all these methods as to which is the most optimal representation of given the geometry.
- **Metric map models: Adaptive representations** Rather than forcing the robot to use a particular representation of a map, a better strategy would be let the robot adaptively figure out the right representation based on its constraints.
- **Theoretical tools: Generality, gaurantees and verification** Apart from pose graph optimization, guaranteed theoretical results on global solutions and verification techniques have not been extended to factor graphs. Most theoretical models assume the measurement model to be isotrpic. It will be more pose if this is extended to more arbitrary noise models.

- **Active SLAM: Fast prediction of future states:** Active sLAM is the process of making decision on a robotic motion to maximize out objective of building a clean map. In order for this to happen, we must make a fast prediction about the future state, which seems too expensive for today's systems.
- **Active SLAM: When to stop it?** A clear strategy or theoretical estimate on when is the right time to stop doing active SLAM and start doing normal SLAM is still a relatively an unstudied problem.

The future opens up more challenges for the SLAM community. With the growth of SLAM, SLAM is entering a positive feedback cycle where its applications are demanding more than what its presently capable of, which in turn starts driving bigger innovation. Self-driving cars is an example of such a system which has started demanding a long term mapping solution from the SLAM community. Moreover, the growing maturity of sensors and the introduction of new sensors brings new promise. A few new sensing modalities that we left undiscussed in this paper are event-based camera and light field cameras. These sensory innovations coupled with active algorithmic development in SLAM will keep the SLAM community vibrant in research activity for the years to come.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. 2016.

[2] G. Chahine and C. Pradalier. Survey of monocular slam algorithms in natural environments. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 345–352, May 2018.

[3] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 24(5):932–945, Oct 2008.

[4] Laura A. Clemente, Andrew J. Davison, Ian D. Reid, Jos Neira, and Juan D. Tards. Mapping large loops with a single hand-held camera. In *IN PROC. ROBOTICS: SCI. SYST*, 2007.

[5] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cdric Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[6] M. Cummins and P. Newman. Accelerated appearance-only slam. In *2008 IEEE International Conference on Robotics and Automation*, pages 1828–1833, May 2008.

[7] M. Cummins and P. Newman. Accelerated appearance-only slam. In *2008 IEEE International Conference on Robotics and Automation*, pages 1828–1833, May 2008.

[8] Brian Curless and Brian Curless. A volumetric method for building complex models from range images. 1996.

[9] Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1403–1410 vol.2, Oct 2003.

[10] A. J. Davison. Active search for real-time vision. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 66–73 Vol. 1, Oct 2005.

[11] A. J. Davison and D. W. Murray. Simultaneous localization and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):865–880, July 2002.

[12] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29:2007, 2007.

[13] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *I. J. Robotics Res.*, 25(12):1181–1203, 2006.

[14] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. In *arXiv:1607.02565*, July 2016.

[15] Jakob Engel, Thomas Schoeps, and Daniel Cremers. Lsd-slam: large-scale direct monocular slam. volume 8690, pages 1–16, 09 2014.

[16] Jakob Engel, Vladyslav Usenko, and Daniel Cremers. A photometrically calibrated benchmark for monocular visual odometry. 07 2016.

[17] Maksim Filipenko and Ilya Afanasyev. Comparison of various slam systems for mobile robot in an indoor environment. 09 2018.

[18] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, May 2014.

[19] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In Lydia E. Kavraki, David Hsu, and Jonas Buchli, editors, *Robotics: Science and Systems*, 2015.

[20] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual-inertial odometry. 2017.

[21] Jorge Fuentes-Pacheco, Jose Ascencio, and J. Rendon-Mancha. Visual simultaneous localization and mapping: A survey. *Artificial Intelligence Review*, 43, 11 2015.

[22] Dorian Galvez-Lopez and Juan Tardos. Bags of binary words for fast place recognition in image sequences. *Robotics, IEEE Transactions on*, 28:1188–1197, 10 2012.

[23] X. Gao, R. Wang, N. Demmel, and D. Cremers. Ldso: Direct sparse odometry with loop closure. In *iros*, October 2018.

[24] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[25] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Transactions on Intelligent Transportation Systems Magazine*, 2:31–43, 2010.

[26] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *UIST '11 Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, October 2011.

[27] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research, IJRR*, 31(2):216–235, February 2012.

[28] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *Robotics, IEEE Transactions on*, 24:1365 – 1378, 01 2009.

[29] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, Nov 2007.

[30] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, Shanghai, China, May 2011.

[31] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:2006, 2006.

[32] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. , 293:133–135, Sep 1981.

[33] Nicholas Molton, Andrew Davison, and Ian Reid. Locally planar patch features for real-time structure from motion. 01 2004.

[34] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598. AAAI, 2002.

[35] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), Acapulco, Mexico*, 2003.

[36] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, April 2007.

[37] R. Mur-Artal and J. D. Tards. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[38] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tards. Orb-slam: a versatile and accurate monocular slam system. *CoRR*, abs/1502.00956, 2015.

[39] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges,

David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinect-fusion: Real-time dense surface mapping and tracking. In *IEEE ISMAR*. IEEE, October 2011.

[40] T. Qin, P. Li, and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, Aug 2018.

[41] Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors, 2019.

[42] German Ros, Daniel Ponsa, and Antonio M. Lopez. Visual slam for driverless cars: A brief survey. In *in IEEE Workshop on Navigation, Perception, Accurate Positioning and Mapping for Int. Veh*, 2012.

[43] A. Rosinol, M. Abate, Y. Chang, and L. Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020. arXiv preprint arXiv: 1910.02490, https://www.youtube.com/watch?v=-5XxXRABXJsfeature=youtu.be, https://github.com/MIT-SPARK/Kimera, https://arxiv.org/pdf/1910.02490.pdf.

[44] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, June 2013.

[45] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo Taylor, and Vijay Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, PP, 11 2017.

[46] S. Thrun and M. Montemerlo. The graphslam algorithm with applications to large-scale mapping of urban structures. In *International Journal on Robotics Research*, pages 403–430, 2005.

[47] Kanade Takeo Tomasi Carlo. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9:137–154, 11 1992.

[48] R. Wang, M. Schwörer, and D. Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *International Conference on Computer Vision (ICCV)*, Venice, Italy, October 2017.

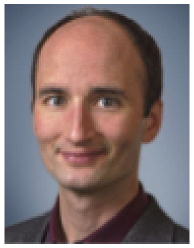
[49] Brian Williams, Mark Cummins, Jos Neira, Paul Newman, Ian Reid, and Juan Tards. A comparison of loop closing techniques in monocular slam, 2009.

[50] N. Yang, R. Wang, J. Stueckler, and D. Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *European Conference on Computer Vision (ECCV)*, September 2018. accepted as oral presentation, arXiv 1807.02570.

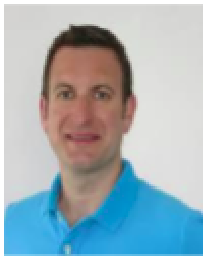
[51] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6612–6619, July 2017.

XVI. Appendix

Active researchers in the area


Daniel Cremers, TUM, Germany
Optimisation, Direct methods

Andrew Davison,
Imperial College, London
Probabilistic methods, AR
applications

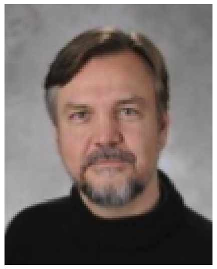

Cyrill Stachniss, University
of Bonn, Germany,
Particle Filters, Semantic
SLAM, SLAM for precision
robots

Sebastian Thrun,Stanford, US.
Mathematical framework of
SLAM systems. (Focuses on AI,
in which SLAM is one of his
interests)

Michael Kaess, CMU, US.
Factor graphs, Sparisty
representations,
Incremental updates

luca carlone, MIT, US
DARPA Competitions, Semantic
SLAM, VI SLAM systems

John Leonard, MIT, US
SLAM for underwater robots


Frank Dellaert, Georgia tech, US,
Monto Carlo methods, Particle filters

Note: The researchers given here and the research topics given under each researcher is by no means exhaustive. These are few researchers who were source of papers during my survey and these are few areas I read a paper about them. SLAM is a pretty big area. I am sure I must have left out someone important or some topics important under some of these reseachers