Implementing Simultaneous Localization and Mapping

Eric Quist, Randy Beard

I. Introduction

FOR autonomous agents to successfully navigate, they must be able to recognize their current position in the world. While many systems rely on the Global Positioning System (GPS) for absolute positioning, there are many GPS-denied environments such as indoors or in urban areas where GPS reception is severely degraded or not available.

The vision community has developed a Simultaneous Localization and Mapping (SLAM) framework[8], [7], [4], [35] that utilizes laser range finders and vision sensors to both recognize previously viewed locations and use successive images to estimate changes in an agent's pose[29]. The SLAM approach provides a valuable framework that allows for other non-optical sensors in addition to, or in place of, the optical sensors for which it was designed.

This paper will provide an overview of existing SLAM techniques and a brief review of several implementations. This will be done by describing the SLAM problem formulation, followed by the details involved in implementing both the image recognition and motion estimation approaches. The application of SLAM using other sensors, such as radar, will also be discussed.

II. An Overview of SLAM

A. SLAM formalization

The SLAM problem formalizes a method for simultaneously estimating position and error of the controlled agent to observed, stationary objects. While many algorithms have been proposed to perform simultaneous localization and mapping, the probabalistic formulation of each algorithm is relatively consistent. The problem consists of multiple observed states and landmarks (as shown in blue in Figure 1). The intent of SLAM is to estimate the true states and landmark positions from the observations (as seen in white).

The precise formulation of this problem involves the following quantities:

- x_t - The true state vector of the unmanned vehicle at time t
- m_i - The true position vector of landmark i
- u_t - The control vector applied at time t
- c_{it} - The observation of landmark i at time t. In the event of a single landmark observation at time t, c_t is used.

The collection of the above quantities can be represented in the following sets:

The authors are with the Department of Electrical and Computer Engineering, Brigham Young University, Provo, Utah, e-mail: {quist, beard}@byu.edu

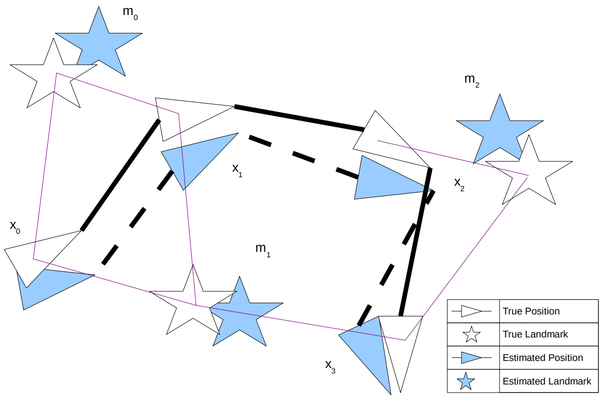

Figure 1. SLAM Problem Diagram: The true states and landmark positions are shown in white triangles and stars respectively. The measured states and landmarks positions are shown as blue arrows and stars respectively.

- $X_{0:n} = \{x_0,\ x_1,\ ...,\ x_n\}$ - The set of all unmanned vehicle states from start to time n
- $M = \{m_0,\ m_1,\ ...,\ m_l\}$ - The map, describing the set of all l landmark states
- $U_{1:n} = \{u_1,\ u_2,\ ...,\ u_n\}$ - The set of all unmanned vehicle controls applied from start to time n
- $C_{0:n} = \{c_0,\ c_1,\ ...,\ c_n\}$ - The set of all landmark observations from start to time n

The problem involves seeking to estimate the vehicle state and map states using the observations, controls, and initial state. Let

$$P(x_n,\ M \mid U_{0:n},\ C_{0:n},\ x_0) \tag{1}$$

be the probability of state and the current map, given the applied controls, landmark observations, and initial state. It is assumed that the vehicle motion model, $P\left(x_k \,|\, x_{n-1},\ u_n\right)$, is a Markov model, which results in the **Time-Update** predictive step

$$\begin{aligned} &P(x_n, M \,|\, U_{0:n}, C_{0:n-1}, x_0) = \\ &\int P(x_n| \, x_{n-1}, u_n)P(x_{n-1}, M \,|\, U_{0:n-1}, C_{0:n-1}, x_0)dx_{n-1}\ , \end{aligned} \tag{2}$$

which predicts the next state. An application of Bayes Rule shows that Equation1 can be refactored as

$$P(x_n,\ M \mid U_{0:n},\ C_{0:n},\ x_0) = \frac{P\left(c_n \,|\, x_n,\, M\right) P\left(x_n,\ M \mid U_{0:n},\ C_{0:n-1},\ x_0\right)}{P\left(c_n \,|\, C_{0:n-1},\, U_{0:n}\right)},$$

which is referred to as the **Measurement Update**. The Measurement Update describes the predicted vehicle and map states as a function of the prediction estimate and the new landmark observation(s). Combining the Measurement Update with the Time Update, results in Equation 1 being reordered to be

$$P(x_n,\ M \mid U_{0:n},\ C_{0:n},\ x_0) = \frac{P\left(c_n | x_n, M\right) \int P(x_n |\, x_{n-1}, u_n)}{P\left(c_n \,|\, C_{0:n-1},\, U_{0:n}\right)} \cdot P(x_{n-1}, M | U_{0:n-1}, C_{0:n-1}, x_0) dx_{n-1}, \quad (3)$$

which allows for the recursive nature of the state estimation to be seen, as the new state estimate becomes a function of the vehicle model, the observation model, the new landmark observations, and the previous state estimate, $P(x_{n-1}, M \,|\, U_{0:n-1}, C_{0:n-1}, x_0)$.

While this formulation allows for accurate estimation, if landmarks are never revisited, the accuracy of the estimate will diverge. By re-visiting and recognizing previously viewed landmarks, **loop-closure** occurs, which results in convergence of the estimated vehicle and landmark states, significantly increasing the estimation accuracy.

It is important to note that landmark recognition may involve map landmarks that the agent has never visited, but were stored in the agents memory to provide absolute positioning. When these landmarks are visited, any inaccurate biases in the map may be resolved by correlating the map to these landmarks where the position is precisely known.

B. SLAM Implementations

The liturature provides a variety of SLAM implementations. Accordingly, a brief discussion of several, more common, implementations allows for better understanding of the strengths and weaknesses of SLAM as a whole.

1) EKF-SLAM: The most common SLAM implementation is the Extended Kalman Filter SLAM (EKF-SLAM) algorithm which models the noise as Gaussian, and implements a Kalman Filter to estimate the system state. It's main limitation is its inability to overcome initial motion estimates and it's inability to perform in real-time scenarios. Additionally, EKF-SLAM is well known to be inconsistant, allowing for a converging covariance matrix, while the data may not justify such confidence.

2) FastSLAM: FastSLAM[26] was the first SLAM algorithm to implement a particle filter heuristic to estimate the motion. FastSLAM simplifies the model by means of a Rao-Blackellised (RB) Filter, which allows for real-time, direct representations of non-linear process models involved. Even with the simplification allowed by the RB Filter, FastSLAM is still unable to handle the large datasets that often occurs in GPS-denied environments. FastSLAM has also been shown to be inconsistant in its estimation.

3) GraphSLAM: GraphSLAM[36] poses SLAM in information-state form, which allows for large data samples while also taking advantage of the sparsity of the covariance matrix. In the information matrix, each vehicle state is described. Each landmark is also represented in reference to the vehicle state where it was visible. Initially, the algorithm populates the information matrices for all positions using only control data $u_{1:n}$.

Once initial position estimations are in place, GraphSLAM updates the information matrices using the observation model for each observed landmark for each location at which it was observed. This update step linearizes the non-linear dynamics, while taking into account all measurement data. GraphSLAM then temporarily reduces the size of information matrices, eliminating all landmark observations. Once the observations are removed, the algorithm uses least-squares optimization to calculate the precise vehicle position. The vehicle position estimates are in turn used to estimate the landmark locations.

For loop closure, GraphSLAM iteratively uses the SLAM posterior and the current position estimate to calculate the probability that the newly acquired landmark has been previously viewed. If the probability exceeds some threshold, the relationship between the newly acquired data and previously viewed landmark is recorded. By identifying previously viewed landmarks allows, the constantly updating information matrices converge.

4) FrameSLAM: Konolige's FrameSLAM[14], [15] approach varies from EKF-SLAM, FastSLAM, and GraphSLAM in that it was designed to be able to use only visual data to localize and map an area. Using Visual Odometry (VO) for motion estimation, FrameSLAM compares features from successive images, calculating the state changes that would correspond to the changes is the imagery. Minor modifications to the VO algorithm provides a baseline for matching revisited landmarks.

To correlate imagery, a Bayes net is formed containing all frames as features. Reduced versions, or *skeletons[34],* are then created to simplify the estimation step, while maintaining accuracy. A FrameSLAM Bayes nets consists of image frames represented as point transformation from world to camera coordinates $c_i = [x_i,\, y_i\,, z_i,\, \phi_i,\, \psi_i,\, \theta_i]^T$, features $q_j = [x_j,\, y_j,\, x_j]^T$represented by their 3-D position, and the system states z_{ij}. The same state transformations that represent VO transformations can be similarly used to describe other sensor measurements such as IMUs. To create a skeleton, FrameSLAM reduces the graph, marginalizing coordinates, features, and/or states into synthetic constraints that represent the accumulative affect of the removed variables.

For loop-closure, FrameSLAM uses a combination of image features and VO to determine if a feature is revisited. While pose prediction is relied upon to consider similar landmarks, the features are relied upon as well, thus resulting in a significantly more accurate SLAM model.

5) VSLAM: VSLAM[17], [16], an algorithm developed by Konolige et al uses the VO and skeletons found in FrameSLAM, while adding Nister's vocabulary tree[30]. Rather than limiting the considered re-visited features to those in the predicted area, it uses a vocabulary tree and image database to perform the place recognition (PR).

The vocabulary tree provides a methodology to describe features. The image comparison uses the Term Frequency Inverse Document Frequency (TF-IF) approach[30], [32] to compare the newly acquired image to previously acquired images resulting in multiple candidate images. The relative geometric positioning of the features of both the candidate images and the newly acquired image are then compared to validate that the images are truly the same.

Additionally VSLAM addresses the memory concerns involved with SLAM. It adds a methodology for image deletion, thus removing duplicate images. His deletion method also considers the age of the acquired image, thus allowing for long-term scenery changes as well as short-term occlusions.

III. Placement Recognition using an Image Database

A. Feature Descriptors

Understanding image recognition for placement recognition relies on the ability to describe attributes, or features, contained in an image. Such descriptors can be grouped into two classes. Point descriptors describe changes around individual image points. Lowe [19] proposed such a descriptor, termed a Scale Invariant Feature Transform (SIFT). SIFT repeatedly uses a smoothed version of an image, comparing it to an even more smoothed version of the same image. The descriptor is the vector describing the differences between each smoothing level. Bay et al [5] recently developed a related descriptor, termed SURF (speeded up robust features), which improves computational efficiency and transformational invariance. The Gradient Location-Orientation Histogram (GLOH), proposed by Mikolajczyk[23] more directly extends SIFT by creating a log-polar location grid using the SIFT descriptor.

Region descriptors emphasize regions that contain areas of similar or differing contrast, color, or other characteristics. The Harris-Affine detector and Hession Affine Detectors[25] both create an elliptical region for each detected interest point. A maximally stable extremal region (MSER) detector, developed by Matas et al[22], describes a threshold that maximizes or minimizes the differences between external and internal intensities for a specified region. Tuytelaars and Van Gool [23]proposed the extrema-based region detector, EBR, which describes the intensity function of rays emanating from instensity extremum.

Each descriptor, both point and region, offers its own set of advantages and disadvantages. Significant work[24], [23] has been done to fully characterize the many descriptors, thus providing a base-line comparison of each. The results suggest that using multiple descriptors concurrently often allows for more accurate image recognition,.

B. A "Bag of Words" model

Building on feature extraction, early work involved comparing all features of all database images to that of the query image, identifying the database image with the most-similar features. While this is often accurate, it cannot be done real-time on possibly small agents. To address this, Sivic[33] proposed a "bag of words" approach similar to the way internet searches are performed. In his implementation, a representative set of training images is collected. Using the k-means clustering algorithm[21], the features from these images are used to partition the feature space into k clusters, or "words." Each feature in the image database is then assigned to its most similar word. The same word assignment is then performed on the features extracted from the query image.

Once features are described by a finite set of words, the images are compared using a weighted tf-if vector[3], or "term frequency-inverse frequency" vector. This involves creating a vector $V_i = [\ t_0 \quad t_1 \quad ... \quad t_{k-1} \quad t_k\ ]$ describing each image, i. Each element t_j is defined as

$$t_j = \frac{n_{ji}}{N_i} log \frac{M}{m_j}\ . \tag{4}$$

It weighs n_{ji}, the number if times descriptor j is found in image i, in comparison to N_i, the total number of words found in image i, and M, the number of images in the map, in comparison to m_j, the number of and occurances of descriptor j in the map. This emphasizes the occurance and non-occurance of repetitive words in individual images or areas. The tf-if vector from the query image is then compared to the tf-if vector from each image in the database. The most similar image is identified as the match.

The bag of words model has performed remarkably well. Its main limitation is its inability to perform real-time image description, as each feature must be compared to each word, prior to image comparison.

C. Scalable Recognition with a Vocabulary Tree

Nistér and Stewénius addressed the computational complexity involved in image comparison using their Vocabulary Tree *[30]* approach. It is built on Sivic's work using visual words, but rather than clustering all words into a single, flat set of *visual words*, they iteratively cluster the words resulting in a *vocabulary tree*. To do this, they first group the entire set of image descriptors D, into m clusters (where m is much less than the k used in Sivic's algorithm). Each descriptor d_i, $d_i \in D$, is then assigned to the word w_j with the closest cluster center. This results in m disjoint subsets of $D = d_1 \cup d_2 \cup ... \cup d_m$. Each word cluster d_j is then subdivided into m new clusters. Again, each descriptor $d_i \in d_j$ is assigned to the word closest word center from the newly created clusters, such that $d_{j1} \cup d_{j2} \cup ... \cup d_{jk} = d_j$. The depth L of the tree is user-selectable, and as the vocabulary tree is generated from training data, it can be generated prior to an agents mission.

The use of weighting for a tree structure is more complicated as the use of the tree leaves or leaves and branches must be considered. In Nistér's paper, many comparison methodologies were explored, such as altering L, m, the comparison metric, discarding very common or uncommon words, and the precise weighting approach. The results can be summarized as indicating that the use of a vocabulary tree in image recognition performed in a similar fashion than that of a flat vocabulary. It also indicates that using only leaf nodes in the image vector results in better recognition.

By using a vocabulary tree, the hierarichal nature decreases the feature comparisons required to identify the most similar word. While recognizing the most similar word using a flat vocabulary tree has $k * N_I$ complexity, using the same number of leaf-words (ie. $m^L = k$) results in word recognition complexity of only $LN_I m = LN_I k^{1/L}$. Both Nistér*[30]* and Konolige*[16]* report to have real-time database creation and localization using the Vocabulary Tree algorithm.

IV. Motion Estimation using Optical Imagery

The use of optical imagery to estimate motion seeks to extract the 3-Dimension change in motion from at least two, 2-Dimension images. This is done by correctly identifying multiple features in each image. The feature locations are then used to estimate the motion required to generate the sequence in imagery.

A. Feature Comparison and Correspondence

Correctly identifying common features in multiple images requires a much more precise comparison methodology than that found in the vocabulary tree. While the vocabulary tree seeks to find similar words, accurate motion estimation hinges on the ability to precisely identify specific, common features in each image. Many methodologies exist for such comparison including kNN Classification, k-Means, randomized trees, and others. For any methodology to function, a metric must be used for comparison.

Lowe discusses much of this presentation of the SIFT descriptor[20]. He indicates that the use of Euclidean distance to compare features yields the best results. The dimensionality (64 or 128 elements) of both SURF and SIFT descriptors prohibit them from benefiting from many of the existing comparison algorithms, though Lowe successfully used a Best-Bin-First (BBF) algorithm[2] which essentially sorts the descriptors in the feature space. Descriptors are then compared to the closest features. Rather than an exhaustive approach, comparing features that have the same vocabulary tree word classification[9], [6] yields similar results, while limiting the number of required comparisons.

While finding the best feature is useful, using such a comparison exclusively can result in incorrect matches. Nistér's original approach was a dating scheme where a match only occurs when both parters believe the other is the best match[29]. Lowe, on the other hand, suggested the use of a distance ratio[20], also discussed by Amato[1]. He defines the distance δ from a feature f_i to it's nearest neighbor $NN_1(f_i, d_j)$ in image d_j as compared to the distance to it's second nearest neighbor $NN_2(f_i, d_j)$. When the resulting distance ratio

$$\sigma(f_i, d_j) = \frac{\delta(f_i, NN_1(f_i, d_j))}{\delta(f_i, NN_2(f_i, d_j))}$$

is greater than some threshold T (Lowe and Amato use 0.8), the features are judged to be indescernible. On the other hand, when $\sigma(f_i, d_j) < T$, the features $NN_1(f_i, d_j)$ and f_i are considered the same feature.

B. Visual Odometry

Performing VO relies on a set of acquired Images $I_{0:n} = \{I_0,\ I_1,\ ...,\ I_n\}$. The m feature points found in image I_j are defined to be $F_j = \{f_{j,1},\ f_{j,2},\ ...,\ f_{j,m}\}$, where the location of the feature $f_{j,k}$ is defined as $f_{j,k} = \begin{bmatrix} x & y & z \end{bmatrix}^T$. The camera's *intrinsic parameters*

$$K = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

is used to map the features to the 2-D image, resulting in

$$p_{j,k} = \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KL_{j,k} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix},$$

where λ is a scaling factor, u and v represent the 2-D location in image I_j of feature $f_{j,k}$. The resulting features (in 2-D) at time j are $P_j = \{p_0,\ p_1,\ ...,\ p_k\}$.

The intrisic parameters matrix K is also used to generate the normalized image coordinates

$$\tilde{p} = \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}.$$

The Rigid-body transformation matrix is defined as

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$

for the rotation matrix $R_{k,k-1} \in SO(3)$ and the translation matrix $t_{k,k-1} \in \mathbb{R}^{3x1}$. The resulting collection of transformations $T_{0:n} = \{T_{0,1},\ T_{1,2},\ ...,\ T_{n-1,n}\}$ correspond the estimated movement by the agent. Each camera pose C_k is the result of the prior camera pose and the cooresponding transformation,

$$C_k = T_{k,k-1} C_{k-1}$$

resulting in a collection of camera poses $C_{0:n} = \{C_0,\ C_1,\ ...,\ C_n\}$.

C. Representing Dimensionality

Handling the dimensionality, whether 2-D or 3-D, of the problem often involves range calculations. For this, stereo vision naturally provides some depth information, yet as the distance between the observer and the observed features grows, the range accuracy deteriorates. Many other difficulties exist such as rough terrain, blurred imagery, and non-stationary image features.

1) Motion Estimation for 3-D-to-3-D: 3-D-to-3-D comparisons require that both previously acquired and newly acquired features already have 3-D positions defined prior to motion estimation. Such representations are almost exclusively performed using stereo vision systems.

2) Motion Estimation for 2-D-to-3-D: 2-D-to-3-D comparisons rely on previous 3-D feature position estimates generated by triangulation of previous images (or information gleaned from other sensors). These 3-D features are compared to to the newly acquired 2-D image, estimating the pose change that would result in the 2-D image. The solution of the 3-D-to-2-D problem requires at least three feature correspondences, resulting in at most four possible solutions (using four feature correspondences results in a single, unique solution).

The model calculates the transformation T_k, using the previously viewed, and localized, features $F_{k-1}$and the newly acquired features P_k. It is important to recognize that using a monocular camera requires the use of at least three prior feature observations to calculate the range information necessary to represent F_{k-1} accurately[12], [31].

The T_k selected minimizes the re-projection error between each new feature $p_{k,i}$ and the transformed, previously viewed feature $\hat{p}_{k-1,i} = T_k f_{k-1,i}$, selecting the most similar features

$$arg\ min_{T_k} \sum_i \|p_{k,i} - \hat{p}_{k-1}, i\|^2 .$$

The number of features used to calculate the translation, or change in perspective, is often referred to as the *perspective from n points* (PnP) problem. Considerable work has been performed both in minimizing the number of points needed, as well as increasing the accuracy given a fixed number of features, resulting in a wide variety of solutions[13], [27]. For visualization, a single, linear PnP solution using $n \geq 6$ can be found using a problem, where each $f_{k-1,l} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ point has the form[31], [11]

$$A_1 P_k^{'} \quad = \quad \begin{bmatrix} 0 & x \\ 0 & y \\ 0 & z \\ 0 & 1 \\ -x & 0 \\ -y & 0 \\ -z & 0 \\ -1 & 0 \\ x\tilde{v} & -x\tilde{u} \\ y\tilde{v} & -y\tilde{u} \\ z\tilde{v} & -z\tilde{u} \\ \tilde{v} & -\tilde{u} \end{bmatrix}^T \begin{bmatrix} P_k^1 \\ P_k^2 \\ P_k^3 \end{bmatrix} \quad = \quad 0 \quad ,$$

where $P_k^{j^T}$ is the j^{th} row of $P_k = [R_k|t_k]$. The 2-D feature coordinates are represented in their homogenous homogeneous coordinates $\tilde{p}_{k-1,l} = [\tilde{u}, \tilde{v}, 1]^T$. Combining the equations from all six features results in

$$\begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \end{bmatrix} P_k^{'} = 0 \ ,$$

where solving for P_k results in the desired motion estimate.

While the P6P is linear, the minimal solution (for 2-D-to-3-D) involves three points[10], resulting in the *perspective from three points* (P3P) algorithm. Kneip et al has developed an approach that performs the estimation very efficiently[13]. All approaches to the P3P implementation results in 4 possible pose solutions, which can be reduced to a single solution using another single feature.

3) Motion Estimation for 2-D-to-2-D: 2-D-to-2-D approaches do not estimate the position of the features in 3-D, but rather seek to estimate the motion using only 2-D image comparison. Using only the dimensionality of the involved images, there is no need to triangulate across multiple, prior features. This approach focuses on calculating the essential matrix

$$E_k = \lambda t_k \hat{R}_k \ ,$$

where $t_k = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$and the skew-symmetric matrix

$$\hat{t_k} \simeq \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

for some scalar λ. The normalized feature points $\tilde{p}_{k,i}$ and $\tilde{p}_{k-1,i}$ are then be used to calculate E_k, where $\tilde{p}_{k,i}^T E_k \tilde{p}_{k-1,i} = 0$.

A simple, brute-force approach to solving for E_k involves using a Least-Square Estimator for all matched feature points. For a single feature, this involves re-factoring

$$\begin{bmatrix} \tilde{u}_{k,i} & \tilde{v}_{k,i} & 1 \end{bmatrix} E_k \begin{bmatrix} \tilde{u}_{k-1,i} \\ \tilde{v}_{k-1,i} \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} \tilde{u}_{k,i} & \tilde{v}_{k,i} & 1 \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} \tilde{u}_{k-1,i} \\ \tilde{v}_{k-1,i} \\ 1 \end{bmatrix} \qquad (5)$$

as

$$A_{k,i} e = \begin{bmatrix} \tilde{u}_{k,i} \\ \tilde{u}_{k-1,i} \\ \tilde{u}_{k,i}\tilde{v}_{k-1,i} \\ \tilde{u}_{k,i} \\ \tilde{v}_{k,i}\tilde{u}_{k-1,i} \\ \tilde{v}_{k,i}\tilde{v}_{k-1,i} \\ \tilde{v}_{k,i}\tilde{u}_{k-1,i} \\ \tilde{v}_{k-1,i} \\ 1 \end{bmatrix}^T \begin{bmatrix} e_{11} \\ e_{12} \\ \ldots \\ e_{33} \end{bmatrix} . \qquad (6)$$

Combining the $A_{k,i}$ for each of the m common features selected in images I_k and I_{k-1}results in

$$A_k = \begin{bmatrix} A_{k,1} \\ A_{k,2} \\ \ldots \\ A_{k,m} \end{bmatrix} .$$

Using A_k, E_k can now be estimated using the least-square estimation equation

$$\hat{e} = min_e e^T A^T A e \ .$$

The least-squared approach, while simple is computationally complex, as A_k is 9 by m. It also heavily weights incorrect correspondence pairs. Recognizing that A_k is only rank 8, the 9th dimension is loss as A_k as it is only calculated to scale, allows for the use of only eight point correspondences being necessary[18]. While this decreases the complexity, the heavy weighting of errors remains. Running RANSAC on top of the *eight-point algorithm* allows for the essential matrix with the largest number of inliers to be the chosen matrix.

Nistér developed a 5-feature algorithm[28] for a calibrated camera, which will estimate motion. The resulting motion estimate does not result in a single solution, but four 4 possible solutions of R and $\hat{t}$, defined as

$$R = U\left(\pm W^T\right)V^T$$

and

$$\hat{t} = U\left(\pm W^T\right)SU^T \ ,$$

where

$$W = \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} .$$

The correct solution is chosen by comparing the resulting point positions. Only one solution allows for both 3-D point positions to be viewed by both cameras.

The actual pose rotations can be extracted by noticing from the Rotation matrix

$$\begin{aligned} R_v^b &= \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{bmatrix} \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \end{aligned}$$

by noting that $r_{13} = -\sin\theta$. The other elements may then be used as $\cos\theta = \sqrt{r_{11}^2 + r_{12}^2}$. Knowing both $\sin\theta$ and $\cos\theta$ allows for the precise value of θ. Once known, $c\theta$ can be used in conjunction with r_{11} and r_{12} to calculate ψ. ϕ can also be calculated using $c\theta$, r_{23}, and r_{33}. In the event θ happens to be 0 or π, the trigonometric identities $\sin(u \pm v) = \sin u \, \cos v \pm \cos u \, \sin v$ and $\cos(u \pm v) = \cos u \, \cos v \mp \sin u \, \sin v$ can be used along with the values found at r_{21}, r_{22}, r_{31}, r_{32} to solve for ϕ and ψ.

4) Resolving the Scale Factor in 2-D-to-2-D: The use of the 2-D-to-2-D approach, involves ignoring the unknown scale found in the homogenous representations of each point, rather than a precise Euclidean reconstruction. To accurately propagate motion across multiple, consecutive image pairs, it is necessary to calculate the scale factor resolution. One approach[31] to resolving the scale factor involves selecting two image feature pairs $f_{k-1,i}$,$f_{k-1,j}$ and $f_{k,i}$, $f_{k-1,j}$ and defining the distance ratio

$$r_{k,i} = \frac{\|f_{k-1,i} - f_{k-1,j}\|}{\|f_{k,i} - f_{k,j}\|}.$$

While the distance ratio from a single pair of features may result in large error, using the mean value of all ratios$r_k = avg_i \ r_{k,i}$ may resolve the error. Alternatively, RANSAC can be used to eliminate the outliers, resulting in $r_k = median_{i \in inliers} \ r_{k,i}$. The scale can then be used to calculate the image translation $t_k = r_k \hat{t}_k$.

D. Bundle Adjustment

Previous approaches assume that only two consecutive images are being compared. Bundle adjustment[11] enhances the Maximum Likelihood (ML) algorithm to minimize the projection matrices and 3D feature points, while also minimizing the distance between the measured and projected feature points. It considers common features across multiple images and results in a projective reconstruction in three dimensional space.

The implementation of bundle adjustment is computationally intensive as it is estimating over the camera's 11 degrees of freedom and each 3-D point's 3 degrees of freedom, resulting in 14 parameters. Some work has simplified its complexity, yet a large amount of computation is still necessary. Bundle adjustment is also limited by the accuracy of the initial pose estimate. For these reasons, it is often used as a local optimization step performed after the motion estimation.

The main limitation for implementing bundle adjustment with graph-based SLAM is that for accuracy, it requires many consecutive images, which complicates the state estimation that would be used for pose estimation.

V. Conclusions and Future Worke formu

In this paper, the existing research Simultaneous Localization and Mapping is explored. SLAM allows for agents to estimate their pose while exploring new terrain. While the probabalistic formulation of SLAM is similar across all approaches, each algorithm has different benefits and liabilities. Using a *Vocabulary Tree* in conjunction with a bag of words allows for real-time image recognition. *Visual Odometry* allows for either 2-D-to-3-D or 2-D-to-2-D image and feature comparison.

The formulation of SLAM provides an architecture that allows for a variety of sensors, including radar. Specifically, we intend to replace the vision sensor in SLAM with a Synthetic Aperture Radar (SAR). Using radar also adds range information that cannot be obtained from vision alone. SAR, unlike optic sensors, provides an active sensor that remains unaffected by night, fog, or even rain, thus providing a more robust solution than optics.

References

[1] Giuseppe Amato, Fabrizio Falchi, and Claudio Gennaro. Geometric consistency checks for kNN based image classification relying on local features. In *Proceedings of the Fourth International Conference on SImilarity Search and APplications*, pages 81–88. ACM, 2011.

[2] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[3] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co. Inc, 1999.

[4] Tim Bailey and Hugh F. Durrant-whyte. Simultaneous Localization and Mapping (SLAM): Part II State of the Art. *Robotics and Automation Magazine*, (September):108–117, 2006.

[5] Herbert Bay. *From wide-baseline point and line correspondences to 3D*. PhD thesis, Swiss Federal Institute of Technology, 2006.

[6] Gabriella Csurka, Christopher R Dance, Lixin Fan, Jutta Willamowski, Cédric Bray, and De Maupertuis. Visual Categorization with Bags of Keypoints. In *Proceedings Workshop on Statistical Learning in Computer Vision*, 2004.

[7] Hugh F. Durrant-whyte and Tim Bailey. Simultaneous Localization and Mapping (SLAM): Part I the essential algorithms. *Robotics and Automation Magazine*, 13(2):99–110, 2006.

[8] Hugh F. Durrant-Whyte, D. Rye, and E. Nebot. Localisation of automatic guided vehicles. In G. Giralt and G. Hirzinger, editors, *Robotics Research: The 7th International Symposeum*, volume 25, pages 613–625. Springer Verlag, 1995.

[9] Ethan Eade and Tom Drummond. Unified loop closing and recovery for real time monocular SLAM. In *British machine vision conference*. Citeseer, 2008.

[10] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[11] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge U.K.: Cambridge Univ. Press, second edi edition, 2004.

[12] Thomas S. Huang and Arun N. Netravali. Motion and Structure from Feature Correspondences : A Review. *Proceedings of the IEEE*, 82(2):252–268, 1994.

[13] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, volume 24, pages 2969–2976. IEEE, 2011.

[14] Kurt Konolige and Motilal Agrawal. Frame-frame matching for realtime consistent visual mapping. In *Proceedings International Conference on Robotics and Automation*, 2007.

[15] Kurt Konolige and Motilal Agrawal. FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, October 2008.

[16] Kurt Konolige and James Bowman. Towards lifelong visual maps. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1156–1163. Ieee, October 2009.

[17] Kurt Konolige, James Bowman, JD Chen, Patrick Mihelich, Michael Calonder, Vincent Lepetit, and Pascal Fua. View-based maps. *The International Journal of Robotics Research*, 29(8):941, 2010.

[18] HC Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, Vol. 293(No. 5828):133–135, 1981.

[19] David G. Lowe. Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157 vol.2, 1999.

[20] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, pages 1–28, 2004.

[21] James MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium*, 233(233):281–297, 1967.

[22] J Matas. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, September 2004.

[23] K. Mikolajczyk, T. Tuytelaars, C. Schmid, a. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1-2):43–72, October 2005.

[24] Krystian Mikolajczyk and Cordelia Schmid. Performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10):1615–30, October 2005.

[25] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, J Matas, F Schaffalitzky, T Kadir, and Luc Van Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1-2):43–72, October 2005.

[26] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the National conference on Artificial Intelligence*, pages 593–598. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[27] Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. Accurate non-iterative o (n) solution to the pnp problem. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[28] David Nistér. An Efficient Solution to the Five-Point Relative Pose Problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(6):756–770, 2004.

[29] David Nistér, Oleg Naroditsky, and James Bergen. Visual Odometry. In *Computer Vision and Pattern Recognition 2004*, volume 11, January 2004.

[30] David Nistér and Henrik Stewenius. Scalable Recognition with a Vocabulary Tree. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, pages 2161–2168, 2006.

[31] Davide Scaramuzza and Friedrich Fraundorfer. Visual Odometry: Part I The First 30 Years and Fundamentals. *IEEE Robotics & Automation Magazine2*, 18(December):80–92, 2011.

[32] Josef Sivic and Andrew Zisserman. Video Google: a text retrieval approach to object matching in videos. *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1470–1477 vol.2, 2003.

[33] Josef Sivic and Andrew Zisserman. Video Google: a text retrieval approach to object matching in videos. *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1470–1477 vol.2, 2003.

[34] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Skeletal graphs for efficient structure from motion. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[35] Sebastian Thrun and John J Leonard. Simultaneous Localization and Mapping. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, volume 23, pages 693–716. Springer, August 2008.

[36] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y. Ng, Zoubin Ghahramani, and Hugh F. Durrant-Whyte. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *The International Journal of Robotics Research*, 23(7-8):693–716, August 2004.