Title goes here

Anonymous CVPR submission

Paper ID Occular Tension

Abstract

The overall problem of Visual Simultaneous Localization and Mapping (VSLAM) and its different aspects have been researched in the computer vision community extensively. This paper critiques three papers on VSLAM and provides an analysis of the innovative contributions of each. Robust Large Scale Monocular Simultaneous Localization and Mapping proposes a framework for implementing VSLAM using monocular cameras. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects proposes a method for VLSAM using prior knowledge of the environment and object recognition. Good Features to Track for Visual SLAM (GF-SLAM) focuses on the aspect of finding good visual features to use in VSLAM based on their temporal observability. This paper also provides an analysis of the implementation of the GF-SLAM paper and the recreation of the paper's key experiment.

1. Introduction

The problem of estimating a 3D model the environment as sensed by a camera as well as estimating the camera's trajectory is Visual Simultaneous Localization and Mapping (VSLAM). The computer vision community has researched and provided innovative solutions that focus on different aspects of the overall problem. In this paper, we analyze the following three CVPR conference papers on VSLAM, discussing their innovative solution to a particular aspect of VLSAM or the problem as a whole, implementation, and subsequent conclusions: 1) Robust Large Scale Monocular Visual SLAM [1], 2) SLAM++: Simultaneous Localisation and Mapping at the Level of Objects [2], and 3) Good Features to Track for Visual SLAM [3]. We also provide an in-depth analysis of our implementation of Zhang *et al.* [3] and the conclusions drawn from the reproduction of the paper's key experiment.

2. Robust Large Scale Monocular Visual SLAM

2.1. Problem Statement

The paper focuses on the problem of using calibrated monocular cameras to perform VSLAM while making the algorithm robust, accurate, and scalable. Monocular VSLAM comes with the challenge of not being able to observe the scale of the scene of the environment. In order to overcome this, loop closures (which occur when the camera returns to a previously observed location) need to be detected. This is an issue in large environments where many scenes look alike, and results in an erroneous 3D model if loop closures are not detected properly. Thus, the paper focuses not only on the general problem of monocular VSLAM but also tackles a key subproblem of dealing with loop closure.

2.2. Innovative Contribution

To solve the problem of monocular VSLAM, the authors propose a framework consisting of three parts: 1) a Structure from Motion (SfM) algorithm based on the *Known Rotation Problem* [?] is used to estimate submaps which are parts of the camera trajectory and the unknown environment [?], 2) a loopy belief propagation algorithm is used to efficiently aligns many submaps based on a graph of relative 3D similarities to produce a global map that is consistent up to a scale factor, and 3) an outlier removal algorithm that detects and removes outliers in the relative 3D similarity graph is used to reject wrong loop closures.

2.3. Proposed Method

The paper proposes a four-part framework to implement the innovations that solve monocular VSLAM: keyframe selection, submap reconstruction, pairwise similarity estimation, and large scale relative similarity averaging.

Keyframe selection: For each frame in the captured video, Harris Points of Interest (PoI) are detected and tracked using a Lucas-Kanade tracker. When the Euclidean distance between the PoI of the current frame and previously selected keyframe is greater than a specified threshold, the frame is selected as a keyframe.

Submap reconstruction: Consecutive keyframes are clustered, and using the *Known Rotation Problem*, a SfM algorithm is applied to each one by first extracting the SURF PoI [?] from all member keyframes. Loops are closed inside of each submap by matching these PoI between pairs of keyframes. The epipolar geometry is then calculated using the 5-point algorithm with RANSAC and bundle adjustment [?] between consecutive pairs of images using the SURF matches and tracked Harris PoI. The local 3D orientations are then extracted are used to estimate the global 3D orientation. With this, known tracks of PoI are built and a linear program is used to solve the *Known Rotation Problem* to estimate the camera pose at each keyframe and the associated 3D point to reconstruct the submap [?].

Pairwise Similarity Estimation: Loop closures in the reconstructed submaps are detected by first applying a bag of words approach on the SURF descriptors of the 3D points of all submaps to give each submap a unique descriptor. After that, the relative 3D similarities between each keyframe and its 10 nearest nieghboors is estimated by matching SURF descriptors with the 3D points of each submap using a k-d tree, and then using the 3-points algorithm with RANSAC and nonlinear refinement on those matches.

Large Scale Similarity Averaging: To align the submaps by estimating thier global 3D similarity to the global reference frame, a cost function on relative similarities is minimized by transforming the problem to a graph inference problem. Outliers in the graphs (representing wrong loop closures) are rejected by the *outlier removal algorithm* in which loop closures are incrementally checked by finding the shorted loop of inliers and adding them to the overall graph of inliers if their cycle error and covariance are within specified bounds. Once the outliers are removed, the *loopy belief propagation algorithm* performs the graph inference by accumulating the measurements and variances on temporal subgraphs of the original graph as it builds up final average global similarity. This algorithm is parallelized, so it can be applied on a large scale of submaps.

2.4. Experimental Evaluation

To evaluate the proposed VSLAM framework, the authors compared its performance with that of state of the art algorithms [?] and [?] on the TUM and KITTI datasets and with four different cameras with different resolutions on indoor videos they captured. Each experiment used the same optimized parameters for the various parts of the algorithm. When evaluating the results of the algorithms with respect to ground truth, a 3D similarity obtained from the minimum distance between the estimated and actual camera trajectories was used. When compared to the [?] on the TUM RGB-D dataset, the author's approach resulted in a lower RMSE for camera trajectories than [?]. When compared to [?] on the KITTI dataset, the author's algorithm estimated camera trajectories that were closer to ground truth and thus performed better. When compared to [?] on their own videos, the proposed method outperformed [?] with respect to the ground truth motion of the camera. In addition, the paper discusses the limitations of the framework in not being able to estimate a pure rotation of the camera, the necessity for the sensed environment to be static, and the necessary for consecutive relative similarities to be outlier free. However, the framework still has reasonable performance when applied to datasets that involve some moving objects.

2.5. Subsequent Conclusions

The performance evaluation of the method shows that the authors' proposed monocular VSLAM framework does substantiate their claim. Robust, independent submap generation is achieved by the visual odometry approach based on the *Known Rotation Problem*, and these submaps can be processed and aligned to form the global map and camera trajectory estimates with loop closure through the outlier removal and loopy belief propagation algorithms. Even with the described limitations, the evaluations show the innovative framework does provide a robust, accurate, and scalable solution to loop closure and the overall problem monocular VSLAM.

3. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects

3.1. Problem Statement

The authors of this paper propose an approach to the VSLAM problem using a combination of the KinectFusion algorithm along with an efficient graph based 3D object recognition system. According to them, this approach offers several advantages over existing VSLAM systems in operation that operate at the level of low level primitives (i.e. points, lines, etc).

3.2. Innovative Contribution

The VSLAM problem has been approached from the perspective of 3D object recognition before. However these methods generally reveal huge amounts of wasted computational effort via repeated low level geometry processing of the 3D objects. To counter this, the authors propose the building of pose graph maps based on an "object-oriented" approach that directly encodes the positions of recognized 3D structures. With each new measurement, the graph is continually optimized with new measurements from the sensors and allows for efficient tracking of the camera system based on recognized landmarks. In addition to this, the algorithms make the assumption that the world has "intrinsic symmetry in the form of repetitive objects" thereby allowing for the the objects in a scene to be identified and segmented as salient repeated elements. The algorithm leverages this repetetiveness along with the efficient use of GPU architectures to provide a real time processing system.

3.3. Proposed Method

Creating an Object Database: The authors first create a database of repeatedly occurring objects via known KinectFusion algorithms. These are objects that are subsequently recognized and used in their VSLAM process.

SLAM Map Representation: The authors represent the world via a graph where each node stores the 6DOF pose of discovered objects relative to a fixed world frame as well as an annotation of the type of the object from the earlier created database.

Real-Time Object Recognition: This portion of the method recognizes objects in the world based on standard mesh recognition algorithms. The implementation is parallelized on GPUs to allow the real-time detection of multiple instances of multiple objects. These correspondences are obtained via the use of Point-Pair Features (PPFs) which are four dimensional descriptors.

Camera Tracking and Object Pose Estimation: The iterative closest point (ICP) algorithm is used to to track the pose of the camera model based on the earlier computed object based locations. A Huber penalty function is used to in this optimization process. Criteria is developed to ensure successful convergence of the tracking error.

Graph Optimization: The poses of the static object is now viewed as a graph optimization problem which minimizes the sum over all the measurement constraints based on the known features of each object.

Relocalization: The system accounts for a loss in camera tracking by re-initilazing localization based on matching at least 3 of the objects seen in the previously tracked long-term graph.

3.4. Experimental Evaluation

The authors reference a video submitted along with this paper to CVPR as a better description of the advantages of their method.

Loop Closure: Small loop closures are detected and compensated for by the ICP algorithm. Larger loop closures are compensated for via the use of the relocalization method.

Large Scale Mapping: Scaled mapping of a large room (15mX10mX3m) was obtained along with the mapping of 34 different objects around the room. The algorithm uses no priors regarding the original placement of these objects.

Moved Object Detection: The algorithm also displays the ability to track these objects while they, themselves are in motion.

System Statistics: The algorithm displays the amount of storage used as compared to the more traditional KinectFusion algorithm. The given mapped rooms is stored in about 20MB of space as compared to the 1.4GB used by KinectFusion. The resultant compression ratio is 1/70 which is a dramatic improvement.

3.5. Subsequent Conclusions

The paper makes several bold claims with regard to its own contributions to the literature. The graph based optimization method does indeed seem novel and the system has significantly large data compression ratio when compared to the KinectFusion algorithm. UnfortunatelyThe experimentally evaluated conclusions are quite sparse when compared to the dense and well written introduction and methodology. The biggest issue pertains from the fact that no standard metric is used to compare the system's advantages and efficiently to other 3D object recognition based VSLAM approaches. Thus, though several claims are made about the paper's VSLAM advantages over other methods, there are no easy ways to determine the validity of these claims.

4. Good Features to Track for Visual SLAM

4.1. Problem Statement

This paper focuses on the problem of feature selection for VSLAM since not all features detected and tracked by conventional methods contribute to the accurate estimation of camera trajectories and the overall map. Only a subset of all tracked and detected features actually produces a good result for VSLAM, and furthermore, there are many different ways of detecting these features. As VSLAM applications grow to include large-scale reconstruction from on large sets of features, an efficient algorithm that can be applied to features generated by any detection method is necessary. Thus, this paper focuses on finding the best features out of any given set of previously detected features as a preprocessing step before VSLAM occurs.

In exploring different methods for VSLAM proposed in recent conference papers, we chose to re-implement the methods and key experiment of this paper.

4.2. Innovative Contribution

To solve the problem of finding the best features to use for VSLAM given a set of detected features, this paper proposes a new SVD-based algorithm for more rigorous feature selection to be used during the localization process. The paper develops a mechanism by which such measured features can be ranked with an observability score. The ranked feature set can then be used to generate a reduced feature set with better a better estimation accuracy. The authors call their system the Good Features algorithm (GF-SLAM for short). The innovative contribution of this paper is in the three main aspects of GF-SLAM: 1) the system observability measure for features, 2) the rank-k temporal update of observability scores, and 3) a submodular learning method for completing the set of good features if the initial ranking of scores fails to provide the desired quantity of good features for a given application of VSLAM.

System Observability Measure: To counter the use of extraneous features in the localization process of a VSLAM system, the paper develops a metric of observability scores based on the temporal observability for each detected feature in each captured video frame. Features are represented by their coordinates in the image plane as well as the corresponding world coordinates, allowing this measure to be applied to features extracted from video frames by any detection method. The general idea is that features with higher temporal observability are scored higher and the set of "good features" is created from all features whose scores are above a specified threshold based on the application of VSLAM. These good features are then called anchors.

Rank-k Temporal Update of Observability Score: The generation of feature observability scores requires the computation and storage of large observability matrices that increase over time. The actual act of creating these matrices is inefficient and can lead to delays in the SLAM process. To counter this, the authors propose a new algorithm to generate the observability matrix based on an incremental SVD of the system's observability matrix applied per time step.

Submodular Learning for Feature Grouping: The ranking and thresholding of feature observability scores may not always produce an adequate number of anchors. Thus, this paper proposes a submodular learning algorithm that greedily chooses the best of the remaining features with scores below the threshold to complete the anchor set based on an SVD of the augmented system observability matrix of the anchor set and possible upgraded features.

4.3. Proposed Method

Dynamical Measurement Model of VSLAM System: The authors use a constant velocity motion model that describes the motion of a VSLAM system in standard euclidian three space, $SE(3)$, as the basis for their algorithm. The method requires the position, and orientation as well as the linear and angular velocities of the camera model of the VSLAM system:

$$\boldsymbol{x}^W_{R_k} = (\boldsymbol{r}^W_{R_k} \quad \boldsymbol{q}^W_{R_k} \quad \boldsymbol{v}^W_{R_k} \quad \boldsymbol{\omega}^W_{R_k})^T \tag{1}$$

where $\boldsymbol{x}^W_{R_k}$ is the state vector of the camera model, $\boldsymbol{r}^W_{R_k}$ is its position, $\boldsymbol{v}^W_{R_k}$ is its velocity, and $\boldsymbol{\omega}^W_{R_k}$ is its angular velocity. All these quantities are with reference to the world frame as denoted by the superscript W at time k. The state vector is updated with each time segment by the SLAM system.

The algorithm requires access to the following properties of the features that are located by the SLAM system:

$$\boldsymbol{p}_i^{R_k} = (p^{R_k}_{x_i} \quad p^{R_k}_{y_i} \quad p^{R_k}_{z_i})^T \tag{2}$$

$$\boldsymbol{h}_i^{R_k} = \begin{bmatrix} h_{x_i} \\ h_{y_i} \end{bmatrix} = distort([.]) \tag{3}$$

where $\boldsymbol{p}_i^{R_k}$ is the pinhole projection of the features in homogenous coordinates, and $\boldsymbol{h}_{x_i}$ is the projection of these coordinates on to the pinhole plane in two dimensions. These coordinates are obtained after correcting for distortion which involves the intrinsic parameters of the camera.

Computation of System Observability Measure: To compute the observability score for each feature, the authors define a System Observability Matrix (SOM), Q_{SOM}. Q_{SOM} is defined for a fixed number of frames, τ, of the SLAM process. When the current frame iteration grows larger than τ, the oldest elements of the now fixed size Q_{SOM} are replaced directly by the newest feature values. τ is thus an important input parameter to the system as it governs the efficiency of the overall algorithm. Q_{SOM} itself is built as a function of the current frame number, j, where $j < \tau$.

$$\boldsymbol{Q}_{SOM}(j) = [\ Q_1^T \mid Q_2^T \mid ... \mid Q_j^T\ ] \tag{4}$$

The replacement process for when $j > \tau$ is described in a later section. Each Q_j^T is built and appended to Q_{SOM} as the frame number increments. Each Q_j^T itself has the following form:

$$Q_j^T = [\ H_j^T \mid (H_jF_j)^T \mid ... \mid (H_jF_j^{n-1})^T\ ] \tag{5}$$

where F is the process matrix that relates the motion model defined in (1) from one state to the next. H is the measurement Jacobian where the derivatives are arranged as follows (with features and anchors abbreviated as "feat." and "anch." respectively):

$$\begin{bmatrix} \text{image feat. w.r.t. camera state} & \text{image feat. w.r.t. world feat.} \\ \text{image anch. w.r.t. camera state} & 0 \end{bmatrix}$$

Once Q_{SOM} is computed, the temporal observability scores for each feature is computed as:

$$score(f) = \sigma_{min}(Q_{SOM}\ for\ feature\ f) \tag{6}$$

Rank-k Temporal Update of Observability Score: As can be seen from the preceding section, the computation of Q_{SOM} is an extensive operation. To simplify matters the authors present an iterated SVD approach to creating Q_{SOM} when $j > 3$ and $j < \tau + 1$.

Fundamentally, Q_{SOM} is created via (4) for the first three frames. For subsequent frames, the SVD of Q_{SOM} and combined with corresponding calculated Q_j in (6) to form the new Q_{SOM} . Again, this new definitions of several matrices and involves diagonalization of newly constructed matrices. Please refer to the original text for the complete implementation details. The net gain by the algorithm is the creation of Q_{SOM} by a rank-2r update instead of creating a whole new matrix.

For $j > \tau + 1$, Q_{SOM} is updated via replacement. The newest Q_j in (6) is computed and placed in the positions occupied by the oldest elements of the matrix. As an example, if at time k, $Q_{SOM}(k) = [\ Q_1^T \mid Q_2^T \mid ... \mid Q_k^T\ ]$, then at time k+1, the oldest (i.e. the first element) is replaced, $Q_{SOM}(k+1) = [\ Q_{k+1}^T \mid Q_2^T \mid ... \mid Q_k^T\ ]$.

Submodular Learning for Feature Grouping: If the set of anchors created after the thresholding of the feature observability scores is smaller than desired, a greedy submodular learning algorithm upgrades features with with the next best scores to anchors to complete the anchor set. This is done by creating an SOM of just the anchors and candidate rows for each remaining feature. The algorithm iterates through each candidate feature, appending it's row to the end of the anchor SOM and then taking the SVD of that matrix to get the observability score for that feature via (6) with the modification that Q_{SOM} is now the anchor and candidate row SOM. The feature whose row produces the maximum observability score is upgraded to be an anchor and removed from the set of features. This greedy algorithm recomputes the anchor and candidate SOMs and repeats the process until the the desired number of anchors is obtained.

4.4. Experimental Method

The authors implemented their feature reduction algorithm on readily available simulated VSLAM data that was created by the authors of [31]. The robot is made to move in a circular in a simulated environment of dimensions 12mX12m with 72 landmarks forming a square. The VSLAM process uses an Extended Kalman Filter to estimate the locations of each landmark and subsequent estimate its own position in the map. In the original EKF algorithm, the simulated robot detects as many landmarks that it can detect. The authors apply their SVD ranking approach to select a certain subset of the number of features selected. By changing the number of features used as well as the observation noise, they run the VSLAM simulation several times and compare their results to that of the original algorithm. They run each simulation 40 times and count the number of times their algorithm outperforms the original. The noise values used are $\sigma = \{0.5, 1.0, 1.5, 2.0, 2.5\}$. The number of features used are $K = \{3, 4, 5, 6, 7, 8, 10, 12\}$. The results are provided in Table 1.

Metric Used	Num. Succesful	Accuracy
Translation Error $\sum_k \|\|\Delta \boldsymbol{r}_{R_k}^W\|\|_2$	37/40	92.5%
Rotation Error $\sum_k \|\|\Delta \boldsymbol{\theta}_{R_k}^W\|\|_2$	33/40	82.5%

Table 1: Author's experimental results.

4.5. Our Implementation

We chose to re-implement this ranked features algorithm. The first step was to download and run the original VSLAM EKF algorithm [31] that was used by the authors in their experimentation. The code was readily available to download and use online on the author's page.

Algorithm Structure: The algorithm simulates a robot moving in a circular trajectory surrounded by 72 detectable landmarks. A monocular camera is placed on the robot for visual sensing purposes. Broadly the algorithm is divided in to the following sections:

1. Data Generation - The trajectory to be followed by the robot is generated. Each landmarks's position is also generated. Care is taken to seed the random number generator at a fixed point so as to ensure consistency across measurements.

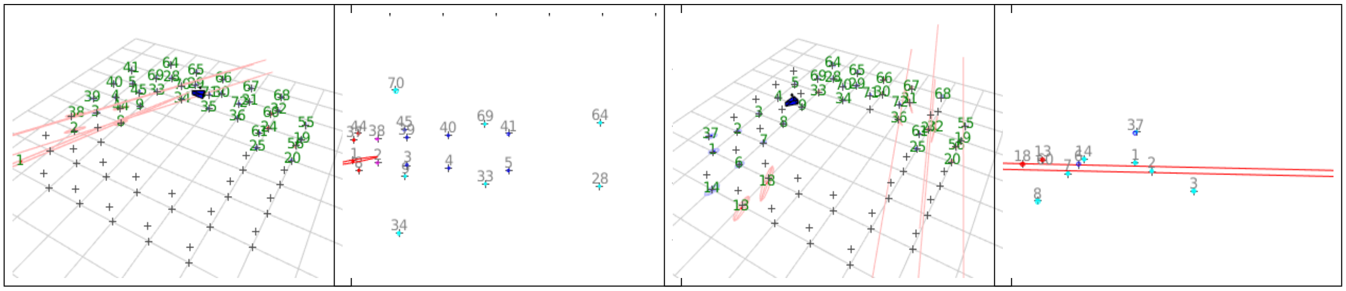
2. Simulation - The simulated robot is moved ahead on the predefined trajectory. The measurement vectors are generated based on the location of the landmarks located by the robot. In the terms of our implementation, we consider the features to be these landmark positions.

3. Localization - The robot's current position and orientation is localized by sending all the detected features to an Extended Kalman Filtering Algorithm.

4. Visualization - The corresponding position of the landmarks, detected features, actual robot's position and orientation and estimated robot's position and orientation are visually updated on a three dimensional plot.

Calculating $\boldsymbol{Q_{SOM}}$: The first step to generating the observability matrix was to map the features detected by the SLAM algorithm to their corresponding $\boldsymbol{p}_i^{R_k}$'s (2) and $\boldsymbol{h}_{x_i}$ (3) indices.This is done between step 2 and 3 by extracting the corresponding elements from the algorithms pinhole perspective calculations. The camera's corresponding position and orientation, $\boldsymbol{x}_{R_k}^W$ (1) , is also extracted. These features are now used to create a corresponding H matrix which in turn is used to iteratively build up Q_{SOM} as described in (4) and (6) . Q_{SOM} is created to contain information from 10 iterations of the EKF algorithm at a time i.e. $\tau = 10$. As described previously, Q_{SOM} is initially created iteratively, then via the iterative SVD and finally via the replacement method.

Feature Scoring: Step two provides us with a list of features that has been detected by the robot. Each of these features have corresponding rows in the Q_{SOM} matrix. For each feature, the corresponding rows are iteratively extracted and concatenated to form a feature matrix. The SVD is computed for each of these matrices and the minimum positive singular value is extracted as a corresponding score



(a) VSLAM (all features) (b) Pinhole View (all features) (c) VSLAM (10 ranked feat.) (d) Pinhole View (10 ranked feat.)

Figure 1: Experimental Results: (a) VSLAM is performed using all the features. All the landmarks (+ signs) are detected by the robot and used for localization. (b) Shows the pinhole view of the robot in the first scene. These pixel coordinates form the $\boldsymbol{h}_{x_i}$ that (3) used in the algorithm. (c) VSLAM is performed with the the 10 highest ranked features. In the figure this is apparent from the fact that all the landmarks in the vicinity of the robot are not labelled. (d) Corresponding pinhole view. As expected, fewer features are being used for the localization process.

of each feature by (4). The resultant scores are then used to sort the original features as well as the corresponding $\boldsymbol{p}_i^{R_k}$'s (2) and $\boldsymbol{h}_{x_i}$'s (3). A subset of the scored features with scores above an application-specified threshold is designated as the anchor set and is passed on to the localization step and the algorithm proceeds as normal.

Submodular Learning: We implemented submodular learning to complete the anchor set as described by the paper's proposed greedy algorithm. This portion of the implementation also runs only if the resultant set of anchors after thresholding the feature scores is less than 2 below the desired number of anchors.

Experimental Implementation: The experiment described by the authors is recreated exactly. 40 experiments are performed at a fixed seeded location. The experiment is run over 200 iterations for each combination of pixel noise and number of anchors from the follow sets: $\sigma = \{0.5, 1.0, 1.5, 2.0, 2.5\}$ and $K = \{3, 4, 5, 6, 7, 8, 10, 12\}$. The positional and euclidian 2-norms are used to compare the results of the original algorithm vs. the ranked features method.

Experimental Results: Table 2 results obtained in terms of the positional norm. Table 3 provides the results obtained in terms of the Euclidian norm. ???The ranked algorithm performed better then the original in 8/40 cases (20%) in terms of the positional norm. In terms of the euclidian norm it performed better 14/40 cases (35%). For a complete tabulation of all the results please refer to the attached appendix.

4.6. Subsequent Conclusions

As can be seen, the results we obtained do not agree with that of the authors' though the experiment is reproduced exactly. We believe the reason for this an incorrect creation of the Q_{SOM} matrix. This paper turned out to be particularly difficult to re-implement. The main reason behind this was the convoluted mathematical definitions and notations used by the authors especially with regard to the contents and creation of Q_{SOM}. We tried to contact the authors via email but are unfortunately yet to hear back from them. They also do not provide any of the software or code used.

In conclusion, based on the metric they define, the author's feature ranking algorithm appears to successfully aid SLAM processes. Unfortunately, their results do not appear to be easy to reproduce. For this algorithm to achieve more success in the community it is our suggestion that the authors clean up their mathematical notation as well as release commented software and code which performs the algorithm successfully.

σ	Avg. $(\sum_k \|\|\Delta \boldsymbol{r}^W_{R_k}\|\|_2)_{all}$	Avg. $(\sum_k \|\|\Delta \boldsymbol{r}^W_{R_k}\|\|_2)_{rnk}$
0.5	6.3591	7.0524
1.0	4.8382	8.9654
1.5	6.1305	10.1870
2.0	6.5744	7.0839
2.5	6.4082	8.5952

Table 2: Position Based Results

σ	Avg. $(\sum_k \|\|\Delta \boldsymbol{\theta}^W_{R_k}\|\|_2)_{all}$	Avg. $(\sum_k \|\|\Delta \boldsymbol{\theta}^W_{R_k}\|\|_2)_{rnk}$
0.5	0.7082	7.2744
1.0	0.9237	5.8175
1.5	0.8389	8.5750
2.0	7.9474	4.7680
2.5	8.0217	6.1176

Table 3: Euclidian Norm Based Results

5. Conclusion

Conclusion goes here.