2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)

AgileSLAM: A Localization Approach for Agile Head Movements in Augmented Reality

Brian M. Williamson[*] University of Central Florida
Andres Vargas[†] University of Central Florida
Pat Garrity[‡] Army Research Laboratory
Robert Sottilare[§] Army Research Laboratory
Joseph J. LaViola Jr. [¶] University of Central Florida

Abstract

Realistic augmented reality systems require both accurate localization of the user and a mapping of the environment. In a markerless environment this is often done with SLAM algorithms which, for localization, pick out features in the environment and compare how they have changed from keyframe to current frame. However, human head agility, such as seen in video gaming tasks or training exercises, poses a problem; fast rotations will cause all previously tracked features to no longer be within the field of view and the system will struggle to localize accurately. In this paper we present an approach that is capable of tracking a human head's agile movements by using an array of RGB-D sensors and a reconstruction of this sensor data into 360 degrees of features that is fed into our SLAM algorithm. We run an experiment with pre-recorded agile movement scenarios that demonstrate the accuracy of our system. We also compare our approach against single sensor algorithms and show a significant improvement (up to 15 to 20 times better accuracy) in localization. The development of our sensor array and SLAM algorithm creates a novel approach to accurately localize extremely agile human head movements.

Index Terms: Human-centered computing—Human Computer Interaction—Interaction Paradigms—Mixed / Augmented Reality; Computing Methodologies—Artificial Intelligence—Computer Vision—Tracking

1 Introduction

Accurate markerless head tracking in augmented reality (AR) is a known difficult problem when compared to head tracking in a pre-mapped space [1,2]. In several cases [6,25], visual Simultaneous Localization and Mapping (SLAM) algorithms developed for robotics provide an adequate starting point for this form of head tracking. However, when considering agile human head movements that are expected in a highly physical task, the existing algorithms lose tracking. This is especially true for large rotational motions as features that were being tracked become lost [22]. These large rotations can be expected of a head mounted system being used in a video game or training exercise that requires extreme movements, such as a first person shooter style game.

Our goal is to develop a hardware system coupled with a SLAM algorithm that can achieve accurate localization and dense mapping while being subject to extreme human head movements. In this paper we focus on the localization problem to provide a proof of concept that our approach produces improved results. We process the feed of an RGB-D sensor array containing four cameras arranged in a square pattern. The sensor array was chosen over a single omni-directional camera so that accurate and dense depth data could be made available to our SLAM algorithm. We extract features from the array of imagery and reconstruct them into their world coordinates to determine a pose estimate since the last keyframe. We hypothesize that by using a 360-degree (horizontal) field of view sensor array, we can significantly reduce localization errors during extreme movements.

To demonstrate the accuracy of our approach we ran an experiment comparing the pose estimate from our system to truth data gathered from a marker based tracking system. We also ran the same motions through existing single sensor SLAM algorithms to show the difficulty of tracking extreme head movements, especially when there are high bursts of angular velocity.

While our SLAM algorithm will continued to be iterated upon, the localization method that we detail still provides valuable novel research. The following are the contributions of this paper:

- An approach to convert the feed from an array of RGB-D sensors to a localization system that provides accurate pose estimates.
- The creation of a data set that includes extreme agile movements.
- A demonstration of the accuracy of our hardware and software system against the data set we recorded.

2 Related Work

Many problems exist within augmented reality research, some of which are the need for accurate tracking, the necessity of environment mapping and the stability of the devices used [1]. For tracking, many techniques have been previously presented with high accuracy being observed via SLAM algorithms in a marker-less environment.

2.1 SLAM Development

The SLAM problem in robotics aims to determine an efficient method for a mobile robot to both map its surroundings and know its location. Two key algorithms that were developed to solve the problem are the extended Kalman filter (EKF-SLAM) and the Rao-Blackwellized particle filter (FastSLAM) [12]. Initially the focus was on the use of advanced robotic hardware such as LIDAR and range finders, but recent advances in computer vision resulted in the development of vision based SLAM algorithms. These systems began by constructing sparse landmark based maps and determining pose estimates from the feature changes [10]. To keep the hardware requirements minimal, researchers developed monocular based SLAM algorithms that operated on sparse maps in a system called MonoSLAM [11]. In addition, progress was made in improving tracking accuracy as usage of these algorithms were beginning in the augmented reality research space [25]. In [21] researchers developed parallel tracking methods, which have high accuracy in a small area. These visual algorithms relied on evolving feature extraction

[*]e-mail: brian.m.williamson@knights.ucf.edu
[†]e-mail: andres.vargas@knights.ucf.edu
[‡]e-mail: patrick.j.garrity4.civ@mail.mil
[§]e-mail: robert.a.sottilare.civ@mail.mil
[¶]e-mail: jjl@cs.ucf.edu

DOI 10.1109/ISMAR-Adjunct.2018.00025

using advancements in computer vision. Commonly used examples are SIFT [23], the optimized version SURF [3] and an alternative method, ORB [31].

The use of SLAM algorithms has a long history of real time tracking. In [25], researchers attempted to determine the depth of a feature from a monocular image by mapping imaginary particles along a line toward the object and re-weighting the particles as more information comes in. The active wearable would use this estimate for mapping and relocalization. In [8] researchers recognized that a human wearing a camera poses a much more difficult problem than a robot using one, due to the agility of the user. They attempted to use the parallel tracking and mapping method [21] to generate several maps that the system uses in an augmented reality application which showed some improvement, but overall localization failed when mapping a large scale environment.

Over time, several promising SLAM applications have emerged for both augmented reality and robotics usage. In [13] a RGB-D sensor was incorporated to create a highly accurate point cloud along with improved accuracy to the localization methods. [14] developed a dense tracking system for monocular sensors using methods described in DTAM [29] which has been shown to solve motion blur issues while tracking. In [27] PTAM [21] was expanded to incorporate ORB features along with several other modern advancements in computer vision, which in the next iteration [28] added RGB-D and stereo sensors.

2.2 Omni-Directional SLAM Research

The use of an omni-directional sensor with a SLAM algorithm is not in itself novel. In [7] LSD-SLAM is modified to incorporate a wide angle camera which provides improved results in their localization experiments and the creation of semi-dense point clouds. They test with hand held motions and show great accuracy as they map a room, but their demonstrated recordings reveal slow methodical movements rather than the fast extreme ones in our problem domain. In [24] a pair of omni-directional cameras are used to create stereo depth maps where the images overlap. While some localization takes place it is not shown how well the system works over a range of motions. [32] and [26] both made great usage of dense omni-directional RGB-D sensors for autonomous vehicles, but the motions they were primarily concerned with were mimicking a vehicle rather than human agility. [26] contained a sensor array most similar to our own, but it was designed with a large distance between the stereo cameras as it was intended for vehicular mounting rather than the human head. We note two primary differences between these research projects and our own. First, the sensor arrays were either very large or were only creating map points at regions of overlap, which were relatively sparse. Second, they focused on either slow methodical movements used in environment mapping, or the movements expected of a robot or vehicle, which did not have high angular velocities. To the best of our knowledge we present the first usage of an RGB-D sensor array designed for the human head that can track highly agile human head movements while simultaneously generating a dense 360-degree point cloud with each frame.

3 AgileSLAM Design

Our system design is built around an array of RGB-D sensors arranged into a square pattern. The individual sensors provide both an RGB frame from which features may be extracted along with a pixel aligned depth frame which can translate the feature to accurate world coordinates.

We did not expect our sensor array to provide perfect coverage, as may be seen with an omni-directional camera, and intrinsic blind spots were not considered detrimental to our localization attempts. If the localization approach is accurate, the blind spots in the environment map will be naturally filled in as the user moves around the room. We create two separate images by concatenating the sensor feeds. The first image is a composition of the RGB frames and the second is the matching composition of depth data.

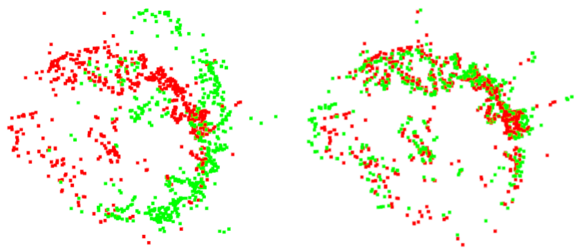

(a) Features Unaligned (b) Features Aligned

Figure 1: An example of feature points placed into world coordinates relative to the sensor array and the result of the optimal rotation and translation calculations. Green points are the current frame, red points are the keyframe.

For each frame we run SURF [3] to detect unique features then convert each feature's pixel coordinate to its world coordinate relative to an individual sensor. The conversion is trivial for the Z value, which is given in the depth data. X and Y coordinates can be determined using projection equations that multiply the point by a matrix containing the focal and central points of the camera. If a feature's depth value could not be determined by the sensor it was immediately discarded.

We also run the frames through a Laplacian filter and take the standard deviation to estimate the blurriness of the image [5]. This presents us with a single number representing the variance within the image, which is lower for blurrier images than clear ones. If no keyframe has been set then the initial frame is evaluated by the number of features present and blurriness value and set as the first keyframe if it qualifies.

If a keyframe is present we match the SURF features between the current color image and the keyframe's composite RGB image. The matching features are then given their world coordinates relative to the sensor array. Because the imagery is composed by concatenation, the horizontal pixel coordinate of a feature also determines the transformation applied to its world coordinate by multiplying by a transformation matrix based on

$$f(u) = \begin{cases} 0 < u < w, & \text{Forward Sensor} \\ w < u < 2w, & \text{90 Degree Sensor} \\ 2w < u < 3w, & \text{180 Degree Sensor} \\ 3w < u < 4w, & \text{270 Degree Sensor} \end{cases} \tag{1}$$

where u is the horizontal pixel coordinate of the feature and w is the width of a single sensor image. This process outputs two sets of features whose points are relative to the sensor array's center, one from the keyframe's coordinates and one from the current frame.

In order to select the best matches and reduce the number of points we run the sets through RANSAC [15] and cull the outliers. Because the features were determined to be quality matches and the world coordinates have a high degree of accuracy due to the sensor depth feed, we find the localization problem reduced to two sets of data points which differ by the change in rotation and translation of the sensor array. We use the Kabsch algorithm [19] to determine the optimal rotation matrix and translation vector. We calculate the centroid of each data set with

$$centroid = \frac{1}{N}\sum_{i=1}^{N}\vec{X} \tag{2}$$

where $\vec{X}$ is a 3D point from the dataset of N points. The covariance matrix H is then constructed with

$$H = \sum_{i=1}^{N} (\vec{X}_{cf}^{i} - centroid_{cf})(\vec{X}_{kf}^{i} - centroid_{kf})^{T} \tag{3}$$

where $\vec{X}_{cf}$ is the 3D point from the current frame's set of points and $\vec{X}_{kf}$ is the 3D point from the key frame's set of points. The covariance matrix is decomposed using singular value decomposition (SVD) [16] and the final rotation and translation are calculated with

$$\begin{aligned} [U,S,V] &= SVD(H) \\ R &= VU^{T} \\ t &= -R \times centroid_{cf} + centroid_{kf} \end{aligned} \tag{4}$$

where U, S and V are matrices from the H decomposition and R and t are the optimal rotation and translation. This composes our pose estimate of the current frame in relation to the keyframe as shown in Fig. 1(a) and Fig. 1(b).

We then evaluate the state of the frame to produce a recommendation regarding the pose estimate. If our analysis determines the frame to be blurry, or there are very few matching features after running the RANSAC algorithm, it is recommended to reject the pose. If the number of matching features are low, but the imagery has consistently not been blurry, a new keyframe is established. This method of evaluation is simplistic and will be expanded upon in future iterations, but served its purpose for our localization experiments.

With the user's pose estimated, the entire RGB frame is combined with the depth data to produce a point cloud. Similar to the transformations used in Equation 1, each point's frame of reference is transformed from relative to an individual sensor to relative to the sensor array. This allows for every frame to produce a 360-degree (horizontal) dense point cloud. If a point is not near another one, it is added to the mapping of the environment.

4 Experimental Setup

With the localization system developed we proceeded to test our hypothesis regarding accuracy and to create baseline metrics for our system in regard to extreme head movements. We designed an experiment involving agile movements with a focus on large rotation deltas. For our sensor array we used four ZED stereo cameras which produce aligned RGB and depth images using two stereo sensors. In our evaluation the depth imagery was more accurate at a distance (greater than two meters) than up close, which is ideal for room-wide SLAM applications. The sensor allows for multiple resolution configurations, however the lowest resolution (672x376) was used to ensure the single USB controller could maintain four simultaneous streams. Furthermore the lower resolution option provided a larger horizontal field of view (~90 degrees) which was also ideal for achieving as much coverage as possible with only four sensors.

4.1 Apparatus

The rig used in our experiment contained four full sized ZED cameras along with a Kinect and HTC Vive tracker, which when combined was larger and heavier than we desired of a head mounted system. A lighter weight version of the ZED sensor is now available that provides the same camera specification as the full sized camera, but this was not available at the time of our experiment. As such, we mounted all of the sensors to a platform (shown in Fig. 2) to be held close to the head. While this was not worn on our head during the experiment, it still allowed for agile motions that simulated the problem we wanted to evaluate.

As movements were performed point cloud data was retrieved from the ZED SDK and recorded to the hard drive. This pre-recorded data afforded us the ability to use the same data through iterations

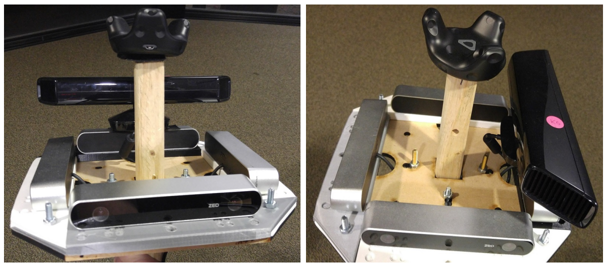

(a) Front View (b) Side View

Figure 2: Sensor setup for experimentation.

of algorithm development. For truth data a HTC Vive Tracker was placed at the center of the rig and recorded alongside ZED frame data. We created a recording tool that synchronized the Vive pose data to the four ZED sensor images.

We also compared our approach to existing and readily available single sensor SLAM algorithms. For this portion of the experiment we attached a Microsoft Kinect to the front of our rig so that RGB and depth data could be provided to these algorithms following the same motions as our sensor array. The Kinect was chosen as it is a familiar commercial sensor that was used in RGB-D SLAM's own evaluation [33] and produced RGB imagery of a comparable resolution (640x480) to what the ZED was set to. This provided us with confidence that any localization errors experienced by the single sensor SLAM algorithms would not be caused by our sensor choice, but rather the motions we were performing. Similar to the ZED sensor array data, the RGB and depth frames from the Kinect were saved to a hard drive to be streamed to the algorithms and compared to the corresponding truth data. A checkerboard recording was taken with the Kinect to retrieve the intrinsic camera parameters: the focal point, principal point and distortion parameters.

The SLAM systems chosen for comparison were LSD-SLAM [14], a monocular dense feature tracking system and RGB-D SLAM [13] a sparse feature RGB-D tracking system. These were chosen based on the availability of the source code and results from internal testing which showed promising accuracy given slow methodical movements. The algorithms were run on an Ubuntu virtual machine with the robot operating system (ROS) [30].

4.2 Data Sets

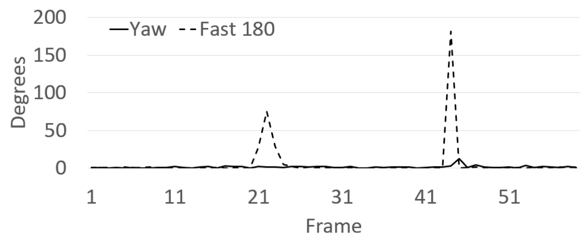
We recorded seven data sets used for frame of reference calibration to the truth data and five data sets to simulate agile head movements.[1] The average length of the recordings were 10-20 seconds depending on the movements performed. The data sets are described in Table 1 with the top portion going over the calibration data sets and bottom portion going through the extreme agility data sets.

For calibration scenarios, the sensors were moved in a slow methodic manner similar to a cautious user slowly mapping an environmnt. The agile simulation scenarios moved in speeds that would be seen of the human head engaged in extreme, but realistic movements, such as in a first person shooter video game task. Table 2 provides the average velocity and angular velocity for every scenario. Averages were accumulated as the sensor moved via a simple high pass filter which excluded periods in which it was stationary. As can be seen, calibration scenarios moved at a slow pace with an average angular velocity near 14 degrees per second and an average velocity of 0.1 meters per second. Agile scenarios had a much higher speed with an average angular velocity of 210 degrees per second and an average velocity of 1 meter per second. While these orientation changes may appear excessive, the human head can move at burst

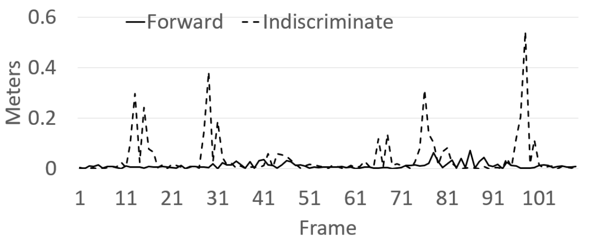
[1] Available for download at http://www.eecs.ucf.edu/isuelab/downloads.php.

Table 1: Descriptions of data sets recorded.

Data Set Name	Description
Translation	Movement only on a single axis (x,y,z).
Rotation	Angular movement only on a single axis (x, y, z).
Look and Forward	Turn 90 degrees and move forward.
Turn and Duck	Simultaneously turn 90 degrees and duck down.
180 Fast Turn	Quickly turn around 180 degrees. Turn back.
Eye Track	Simulate head movements of a user tracking a moving object with their eyes.
Indiscriminate Movements	Simulate fast rotation and position changes similar to what may be seen in a first person shooter.
Flag Movement	Extreme movements that go beyond head agility. Simulate a hand held device being rotated and translated.



(a) Angular Rate of Change Comparison



(b) Translation Rate of Change Comparison

Figure 3: Examples of the rate of change between calibration data sets and extreme movement data sets.

speeds of up to 780 degrees per second [17]. Fig. 3(a) and Fig. 3(b) help to visualize the difference between slow methodical movements and the agile movements from our problem domain. The agile data sets that we recorded are meant to push the SLAM algorithms to a worse case scenario.

4.3 Design

Each algorithm was set up to receive a frame from its expected sensor and as much time as needed to calculate a pose estimate. We did not concern ourselves with the speed of any algorithm as we are focusing on overall accuracy and assume that better hardware can account for real time performance needs. For RGB-D SLAM and LSD-SLAM, Kinect frames, along with intrinsic parameters, were read from the hard drive and sent to the virtual machine via a local TCP connection. A client tool and ROS node built for this experiment received the data and published it to the ROS topics that the algorithms were configured to listen to. The only changes made to LSD-SLAM and RGB-D SLAM source code was to publish pose estimates back to a ROS topic. This topic was subscribed to by our client tool which would send the pose estimate to our data server for comparison to the truth data. AgileSLAM was incorporated directly into the data server and did not require this process.

Table 2: Average velocity and angular velocity of sensor array for each data set.

Scenario	Velocity (m/s)	Angular Velocity (deg/s)
Forward	0.118453	14.66209
Sideways	0.105685	13.04024
Up	0.112185	12.97311
Look and Forward	0.100345	14.98031
Yaw	0.096846	18.35225
Pitch	0.117617	12.52419
Roll	0.138613	15.63995
180 Fast Turn	0.190872	325.6126
Eye Track	0.741853	170.3189
Flag	1.183397	207.1455
Turn and Duck	1.171908	149.7151
Indiscriminate	1.055420	198.8264

Once an algorithm provided its pose estimate, we compared it to truth data recorded with that frame. Error was accumulated into a root mean square error (RMSE) calculation to be evaluated once the data set was completed. Error between quaternions was determined by

$$\begin{aligned} Q_e &= Q_d * Q_t^{-1} \\ \theta &= 2 * acos(Q_e.w) \end{aligned} \quad (5)$$

where Q_d is the quaternion from the pose estimate and Q_t is the truth orientation. The position errors were calculated as the distance between the truth data and pose estimate in meters. After we determined the error for that frame, the next frame was streamed to the algorithm.

The truth data had a frame of reference based on the location of the markers that the system relied upon, while the SLAM algorithms would begin at a frame of reference relative to where the system was when the algorithm started. We performed a rotation and translation of the truth data to transition its frame of reference from the marker's positions to its own starting point. The calibration scenarios were then used to make sure movements on a particular axis had the same meaning between the SLAM algorithms and the truth data.

5 Results

The error experienced by each SLAM algorithm through our scenarios can be seen in Table 3 and Table 4. As expected the errors during the calibration scenarios were low and this allowed them to be used to verify the frame of reference alignment between the algorithm's pose estimate and truth data.

For the single sensor SLAM algorithms, the agile movement scenarios had a large amount of error if tracking was maintained at all. If a system lost frames, but was able to resume tracking, the error that may have been applied during lost tracking was not applied to the average. For position error we see relatively low RMSE values for all of the SLAM algorithms during calibration, with AgileSLAM's maximum RMSE at 0.16 meters, RGB-D SLAM at 0.90 meters and LSD-SLAM at 0.20 meters. As rotation changes and features became lost, this error increased. For AgileSLAM, it is most noticeable with roll and pitch as the sensor array has a limited vertical field of view. With the agile simulations, AgileSLAM maintained a low position error, while RGB-D SLAM and LSD-SLAM had very large errors. AgileSLAM had a peak of 0.32 meters during the scenario Indiscriminate Movements. By comparison RGB-D SLAM had an average error of 2.9 meters during that scenario and LSD-SLAM lost tracking. RGB-D SLAM and LSD-SLAM's error peaked with the eye tracking scenario with errors of 5.12 meters and

Table 3: RMS Error Distance in Meters. X indicates tracking was lost for $>$40% of frames and was not counted.

Scenario	AgileSLAM	RGB-D SLAM	LSD-SLAM
Forward	0.0547	0.0852	0.1460
Sideways	0.0461	0.1585	0.1600
Up	0.0531	0.0868	0.1630
Look and Forward	0.0989	0.8163	X
Yaw	0.0549	0.3617	0.1700
Pitch	0.1690	0.8993	0.2000
Roll	0.1369	0.1768	0.0897
180 Fast Turn	0.0933	2.0988	0.8330
Eye Track	0.1396	5.1240	3.7804
Flag	0.2884	4.9099	X
Turn and Duck	0.0661	2.3348	0.6443
Indiscriminate	0.3207	2.9017	X

Table 4: RMS Error Rotation in Degrees. X indicates tracking was lost for $>$40% of frames and was not counted.

Scenario	AgileSLAM	RGB-D SLAM	LSD-SLAM
Forward	2.4908	2.2355	2.767
Sideways	2.4778	2.5646	3.042
Up	3.6471	4.8335	6.868
Look and Forward	4.7664	11.142	X
Yaw	6.2878	11.519	14.44
Pitch	17.393	23.343	18.98
Roll	14.872	16.586	13.40
180 Fast Turn	4.6179	95.911	101.97
Eye Track	11.454	95.325	132.67
Flag	24.131	83.151	X
Turn and Duck	2.3032	41.005	65.744
Indiscriminate	16.093	79.097	X

3.53 meters respectively. To compare, AgileSLAM had an average position error of 0.13 meters during the same scenario.

For rotation error similar results are seen. AgileSLAM had a peak orientation error of 17 degrees during calibration and RGB-D SLAM peaked at 23 degrees. LSD-SLAM had relatively low orientation errors during calibration, but lost tracking during the look and forward scenario, which was not expected. While the calibration data set moved slowly, LSD-SLAM appeared to struggle with tracking when the yaw movement brought a large featureless television into the frame. With the agile simulations, AgileSLAM maintains a low orientation error with a peak of 24 degrees during another worse case scenario, the flag movement scenario. RGB-D SLAM and LSD-SLAM show very high errors in almost every agile data set with tracking being lost by LSD-SLAM in the two more difficult scenarios, Flag and Indiscriminate Movements.

It should be stressed that the data recorded involved extreme movements that LSD-SLAM and RGB-D SLAM were not designed for. Furthermore, the lab environment used contained large black television screens which can cause difficulties with vision based localization. Our results are intended to show the difficulty of the problem domain and its worse case scenarios more than showing limitations of existing algorithm performance.

6 Discussion and Future Work

In this paper we have shown the creation of a SLAM system designed to work with a RGB-D sensor array that provides improved localization and dense environment mapping during extreme head movements. We also constructed an experiment to show the accuracy of our system in regard to these movements and compared our results with existing single sensor algorithms.

We see that our sensor array and SLAM algorithm, AgileSLAM,

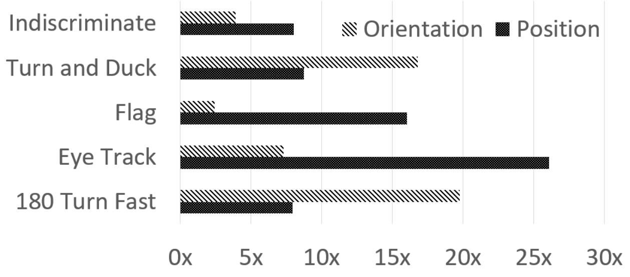


Figure 4: Percentage of error improvement between AgileSLAM and the next best single sensor SLAM algorithm.

was able to maintain a significantly lower error in both rotation and position during calibration and agile movement scenarios over a single sensor system. Fig. 4 illustrates the percentage of improvement in the error seen for both orientation and position estimates compared to the next best single sensor prediction. This is expected as the sensor array is capable of maintaining view of features even if some areas are not ideal or have changed quickly between frames. This supports our hypothesis that accurate localization can be achieved by the usage of multiple sensor feeds and provides us with promise in moving forward in the research.

6.1 Limitations

Our initial prototype of AgileSLAM was able to demonstrate an improvement in localization during extreme movements, however, it still had a peak RMSE of 0.3 meters and 24 degrees, which would be jarring to an augmented reality experience. During calibration of pitch and roll, we also saw angular errors of 16 degrees, which show the limitation of the sensor array's vertical field of view. We believe the introduction of a linear motion model to be used when frames are rejected and an IMU could greatly reduce these errors.

AgileSLAM also did not run in real time during the experiments. We believe there are optimizations that can be implemented, such as offloading data processing to the GPU and culling unnecessary steps given a frame's evaluation. Also better hardware can be utilized to significantly lower processing time.

Our algorithm is in its initial stage and there is the possibility that as it grows in complexity to accommodate versatility, localization results may change. However, we do not believe they will become significantly worse, or degrade to the equivalent of a single sensor system.

We also note that our sensor array proved to be slightly too large for an actual head mount. This was partially because of our incorporation of a truth tracker and a Kinect sensor for the experiments we ran, but the full sized Zed cameras created a significant weight at 0.7kg and a perimeter of 0.7m. Future iterations can make use of a stripped down version of this sensor, the Zed Mini, without sacrificing any of the other technical specs. With the Zed Mini the system would weigh 0.25kg and have a perimeter of 0.5m.

6.2 Future Work

The system as it stands is in its preliminary stages, but represents novel development and a proof of concept to solve extreme agile movement scenarios. In future iterations we plan on the implementation of lessons learned from other SLAM algorithms [9,13,14,20,27] which would include using bag of visual words [34] for keyframe re-localization and loop closure [18].

We are also looking to integrate better frame evaluation alongside linear motion models for when a frame cannot provide an accurate estimate. As points are integrated into the mesh, they may also be evaluated to provide corrections to the original pose estimate via methods such as iterative closest point [4]. We also intend to examine the system with less sensors in different configurations, such as three sensors arranged into a triangle or two sensors forming a spearhead. Lowering the number of sensors needed would both lower the weight and allow for higher image resolutions to be streamed. The use of four sensors will function as our baseline, best case configuration, to which these other configurations can be evaluated against. We would also like to compare against other omni-directional SLAM systems, such as [7] and [32]. Such future comparisons would also modify algorithms as needed to match the hardware we are using for a direct comparison, rather than making use of different sensors such as the Kinect.

6.3 Conclusion

Our initial step has shown promise in greatly reducing localization errors during agile movements. Given this, we believe that proceeding forward will result in a system that can track natural human head movements with accuracy in the centimeter range. AgileSLAM will have future iterations, but it has already shown promise in solving the localization problem while undergoing extreme movements.

Acknowledgments

This work is supported in part by NSF Award IIS-1638060, Lockheed Martin, Office of Naval Research Award ONRBAA15001, and Army RDECOM Award W911QX13C0052.

References

[1] R. T. Azuma. A survey of augmented reality. *Presence: Teleoperators and virtual environments*, 6(4):355–385, 1997.

[2] M. Bajura and U. Neumann. Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications*, 15(5):52–60, Sep 1995. doi: 10.1109/38.403828

[3] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *ECCV*, pp. 404–417, 2006.

[4] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611, pp. 586–607. International Society for Optics and Photonics, 1992.

[5] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. In *Readings in Computer Vision*, pp. 671–679. Elsevier, 1987.

[6] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic. Augmented reality technologies, systems and applications. *Multimedia Tools and Applications*, 51(1):341–377, 2011. doi: 10.1007/s11042-010-0660-6

[7] D. Caruso, J. Engel, and D. Cremers. Large-scale direct slam for omnidirectional cameras. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 141–148. IEEE, 2015.

[8] R. Castle, G. Klein, and D. W. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *2008 12th IEEE International Symposium on Wearable Computers*, pp. 15–22, Sept 2008. doi: 10.1109/ISWC.2008.4911577

[9] A. J. Davison and D. W. Murray. *Mobile robot localisation using active vision*, pp. 809–825. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. doi: 10.1007/BFb0054781

[10] 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, Jul 2002. doi: 10.1109/TPAMI.2002.1017615

[11] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007. doi: 10.1109/TPAMI.2007.1049

[12] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[13] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the rgb-d slam system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 1691–1696, May 2012. doi: 10.1109/ICRA.2012.6225199

[14] J. Engel, T. Schöps, and D. Cremers. *LSD-SLAM: Large-Scale Direct Monocular SLAM*, pp. 834–849. Springer International Publishing, Cham, 2014. doi: 10.1007/978-3-319-10605-2_54

[15] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. doi: 10.1145/358669.358692

[16] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970.

[17] G. E. Grossman, R. J. Leigh, L. Abel, D. J. Lanska, and S. Thurston. Frequency and velocity of rotational head perturbations during locomotion. *Experimental brain research*, 70(3):470–476, 1988.

[18] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987.

[19] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[20] O. Khler, V. Adrian Prisacariu, C. Yuheng Ren, X. Sun, P. Torr, and D. Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE transactions on visualization and computer graphics*, 21(11):12411250, November 2015. doi: 10.1109/tvcg.2015.2459891

[21] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225–234, Nov 2007. doi: 10.1109/ISMAR.2007.4538852

[22] J. J. LaViola Jr, B. M. Williamson, R. Sottilare, and P. Garrity. Analyzing slam algorithm performance for tracking in augmented reality systems. In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, 2017.

[23] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. doi: 10.1023/B:VISI.0000029664.99615.94

[24] C. Ma, L. Shi, H. Huang, and M. Yan. 3d reconstruction from full-view fisheye camera. *arXiv preprint arXiv:1506.06273*, 2015.

[25] W. W. Mayol, A. J. Davison, B. J. Tordoff, and D. W. Murray. Applying active vision and slam to wearables. In *Robotics Research. The Eleventh International Symposium*, pp. 325–334. Springer, 2005.

[26] M. Meilland, A. I. Comport, and P. Rives. Dense omnidirectional rgb-d mapping of large-scale outdoor environments for real-time localization and autonomous navigation. *Journal of Field Robotics*, 32(4):474–503, 2015.

[27] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015. doi: 10.1109/TRO.2015.2463671

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

[29] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2320–2327. IEEE, 2011.

[30] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, vol. 3, p. 5. Kobe, Japan, 2009.

[31] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pp. 2564–2571, Nov 2011. doi: 10.1109/ICCV.2011.6126544

[32] D. Scaramuzza and R. Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE transactions on robotics*, 24(5):1015–1026, 2008.

[33] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580, Oct 2012. doi: 10.1109/IROS.2012.6385773

[34] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pp. 197–206. ACM, 2007.