# Analyzing SLAM Algorithm Performance for Tracking in Augmented Reality Systems

**Dr. Joseph J. LaViola Jr., Brian M Williamson**
**University of Central Florida**
**Orlando, Florida**
**JJL@eecs.ucf.edu,**
**Brian.M.Williamson@knights.ucf.edu**

**Dr. Robert Sottilare, Pat Garrity**
**U.S. Army Research Laboratory-Human Research and Engineering Directorate,**
**Advanced Simulation Technology Division (ARL-HRED-ATSD) Orlando, FL**
**robert.a.sottilare.civ@mail.mil,**
**patrick.j.garrity4.civ@mail.mil**

## ABSTRACT

In developing augmented reality based tutoring scenarios difficult issues can arise if the environment used is not initially known. Lacking in pre-determined fiducial markers, the tracking of the user's position and orientation relative to their starting point can easily be lost. A potential solution lies within the robotics field: simultaneous localization and mapping (or SLAM) algorithms which rely on visual tracking methods to both determine the layout of the environment and the robot's current position and orientation given the previous estimate. However, when applied to a human subject in an augmented reality environment, the agility of their movement during performance activities can lead to issues. In this paper, we discuss the framework used to test a set of SLAM algorithms and determine their capabilities of tracking a human subject performing a variety of movements. We detail the SLAM algorithms analyzed and explain their potential usage given equipment combinations that may be developed in a lab environment. We also go through each movement set, detailing the hardware used in the recording process and how the user's movements are designed to test the limits of a SLAM algorithm. By developing a networked framework, we show how the system is easily adapted to test an algorithm with minimal changes to its code and how it may be used to evaluate future SLAM research. In the end, our framework shows the capabilities and limits of SLAM algorithms when tracking a human user in an augmented reality system.

## ABOUT THE AUTHORS

**Joseph J. LaViola Jr.** is the Charles N. Millican Faculty Fellow and Associate professor in the Department of Electrical Engineering and Computer Science and directs the Interactive Systems and User Experience Research Cluster of Excellence at the University of Central Florida. He is the director of the modeling and simulation graduate program and is also an adjunct associate research professor in the Computer Science Department at Brown University. His primary research interests include pen-based interactive computing, 3D spatial interfaces for video games, human-robot interaction, multimodal interaction in virtual environments, and user interface evaluation. His work has appeared in journals such as ACM TOCHI, IEEE PAMI, Presence, and IEEE Computer Graphics & Applications, and he has presented research at conferences including ACM CHI, ACM IUI, IEEE Virtual Reality, and ACM SIGGRAPH. He has also co-authored "3D User Interfaces: Theory and Practice," the first comprehensive book on 3D user interfaces. In 2009, he won an NSF Career Award to conduct research on mathematical sketching. Joseph received a Sc.M. in Computer Science in 2000, a Sc.M. in Applied Mathematics in 2001, and a Ph.D. in Computer Science in 2005 from Brown University.

**Dr. Robert A. Sottilare** leads adaptive training research within the US Army Research Laboratory where the focus of his research is automated authoring, automated instructional management, and evaluation tools and methods for intelligent tutoring systems. His work is widely published and includes articles in the Cognitive Technology Journal, the Educational Technology Journal, and the Journal for Defense Modeling & Simulation. Dr. Sottilare is a co-creator of the Generalized Intelligent Framework for Tutoring (GIFT), an open-source tutoring architecture, and he is the chief editor for the Design Recommendations for Intelligent Tutoring Systems book series. He is a visiting scientist and lecturer at the United States Military Academy and a graduate faculty scholar at the University of Central Florida. Dr. Sottilare received his doctorate in Modeling & Simulation from the University of Central Florida with a focus in

intelligent systems. In 2012, he was honored as the inaugural recipient of the U.S. Army Research Development & Engineering Command's Modeling & Simulation Lifetime Achievement Award.

**Brian M. Williamson** is a faculty researcher at the University of Central Florida in the department of Electrical Engineering and Computer Science within the Interactive Systems and User Experience Lab under Dr. LaViola. His primary research includes 3D user interfaces with previous papers written on the RealNav system, a natural locomotion system evaluated for video game interfaces. At the University of Central Florida, Brian has published a thesis on natural locomotion, and has been published by IEEE and CHI.

**Pat Garrity** is a Chief Engineer at the U.S. Army Research Laboratory-Human Research and Engineering Directorate, Advanced Simulation Technology Division (ARL-HRED-ATSD), He currently works in Dismounted Soldier Simulation Technologies conducting research and development in the area of dismounted soldier training and simulation where he is the Army's Science and Technology Manager for the Augmented Reality for Training Science and Technology Objective (STO). His current interests include Human-In-The-Loop (HITL) networked simulators, virtual and augmented reality, and immersive dismounted training applications. He earned his B.S. in Computer Engineering from the University of South Florida in 1985 and his M.S. in Simulation Systems from the University of Central Florida in 1994.

# Analyzing SLAM Algorithm Performance for Tracking in Augmented Reality Systems

**Dr. Joseph J. LaViola Jr., Brian M Williamson**
**University of Central Florida**
**Orlando, Florida**
**JJL@eecs.ucf.edu,**
**Brian.M.Williamson@knights.ucf.edu**

**Dr. Robert Sottilare, Pat Garrity**
**U.S. Army Research Laboratory-Human Research and Engineering Directorate,**
**Advanced Simulation Technology Division (ARL-HRED-ATSD) Orlando, FL**
**robert.a.sottilare.civ@mail.mil,**
**patrick.j.garrity4.civ@mail.mil**

## INTRODUCTION

Augmented reality has been shown to create immersive environments conducive to psychomotor training scenarios (Hughes, Stapleton, Hughes, & Smith, 2005). For proper spatial awareness of a virtual object, its placement within the real-world image must be consistent and accurate. Traditionally, this has been accomplished with fiducial markers (Azuma, 1997), which use computer vision to find the marker and place the virtual object with an appropriate transformation applied. However, augmented reality that can operate in an unknown environment allows for a more versatile training system, capable of quick setup and that does not rely on the view of markers to place an object. This marker-less system requires both understanding of the environment and tracking of the user's viewpoint relative to their starting position.

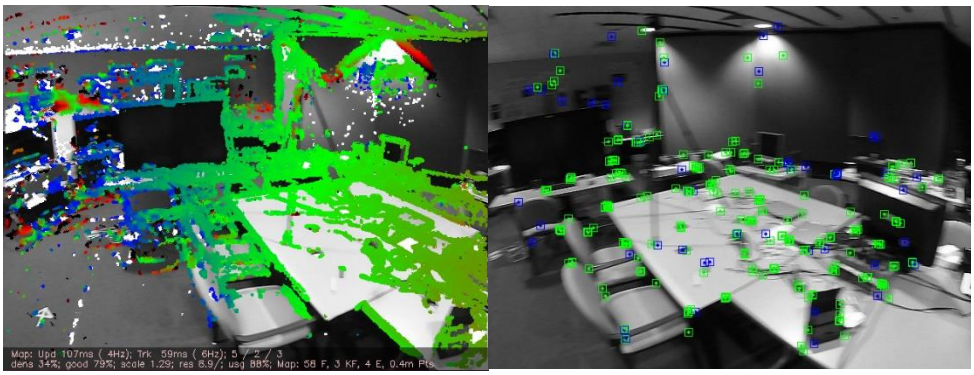

**Figure 1. Example of SLAM algorithms used in test framework.**

Simultaneous localization and mapping (SLAM) algorithms provide the means to accomplish this in the robotics community and have shown promise in augmented reality research as well (Chekhlov, Gee, Calway, & Mayol-Cuevas, 2007) (Comport, Marchand, Pressigout, & Chaumette, 2006) (LaViola et al., 2015). These algorithms take in either a color or depth frame from a sensor and use it to simultaneously understand the environment and update the pose estimate of the sensor. While showing promising results with robotic movements or running with synthetic data (Nardi et al., 2015) (Geiger, Lenz, Stiller, & Urtasun, 2013), the tracking portion of a SLAM algorithm can break down with more agile movements, such as what the human head is capable of.

In this paper, we describe a framework developed for recording data and testing it against a subset of modern SLAM algorithms, examples of which are shown in Figure 1. Due to its networking background, this framework is capable of easily interfacing with an algorithm while requiring minimal change in the source code. Furthermore, the framework can easily be expanded to include the most up to date algorithms that make their source code readily available.

We also go through an experiment run with the framework including the hardware that was used, which includes the HTC Vive (HTC, 2017), Oculus RIFT (Oculus VR LLC, 2017), Structure Sensor (Occipital Inc, 2017) and Intel RealSense (Intel Corporation, 2017). These hardware pieces were used to record data with realistic agile human movements and were run through each SLAM algorithm being tested. Preliminary results are made available.

In the next section, we go through work related to this research. In section three we walk through the development of the framework and explain its capabilities to interface with an algorithm. Section four goes through the experiment run with the framework. Section five concludes the paper and discusses future work.

## RELATED WORK

There is a rich history of utilizing virtual reality and augmented reality for dismounted soldier training. In (Witmer, Bailey, & Knerr, 1995) virtual environments were used for route learning and showed effectiveness so long as the virtual scenario could be made to match the real scenario. It is noted in the study that the virtual elements were used primarily for visual means and that little interaction with the environment was possible at the time. With (Knerr, Lampton, Thomas, Corner, & Grosse, 2003) it was observed that team leader performance improved with training within a virtual environment, suggesting that as the technology advances it could form an effective means of training. Virtual reality training was determined to be useful for decision making, situation awareness, communication, and coordination skills. In (Knerr & Lampton, 2005) virtual reality systems are used for training of military operations in an urban terrain (MOUT) scenario. It is noted that virtual reality provides a reduction in preparation time as it can quickly represent a new area without a different real world set having to be constructed, however accounting for precise movement remains an issue.

Soldier training has also made use of augmented reality research to improve performance. In (Hughes, Stapleton, Hughes, & Smith, 2005) a mixed reality application for another MOUT scenario is evaluated that makes use of blue screen technology and a see-through head mounted display (HMD). This system also made use of a set that provided audio feedback cues in order to increase immersion of the training environment. The Battlefield Augmented Reality System (BARS) (Livingston et al., 2002) also aimed to take modern technology to assist in training for MOUT scenarios. This system featured a see-through display, wearable computer, and GPS device to track the user. While both systems were effective prototypes, they relied on either an area already known by the user or external devices (GPS) to provide accurate tracking of the user.

When looking at SLAM research, potential exists of a single camera device being used to provide the tracking and mapping of an unknown area that a user may need. This can potentially allow for precise movements of a user without having to re-use the same real world set or rely upon technology that may not be available in all scenarios, such as GPS while indoors. While the SLAM algorithms originally worked with robotic based hardware, such as light detection and ranging (LIDAR) devices, it later evolved to make use of computer vision. These systems check the changes between two images to update a pose estimate of the original sensor (Davison & Murray, 2002). As different low-cost hardware became available, the algorithms evolved to incorporate them.

An easy way to distinguish the research into this field is to look at the hardware requirements of the algorithm, whether it works with a depth and color frame or can operate with a single-color sensor. For monocular SLAM, typically only RGB (red-green-blue) frames are used. The parallel tracking and mapping (PTAM) method (Klein & Murray, 2007) showed early promise in tracking for augmented reality purposes. However, this system only functioned on a desktop view and would not attempt to track as the user looked away and around the room. This was later evolved into ORB-SLAM (Mur-Artal, Montiel, & Tardos, 2015), which used oriented fast and brief (ORB) feature detection and the PTAM methods to provide precise room wide tracking. A different technique, LSD-SLAM (Engel, Schöps, & Cremers, 2014), makes use of direct image comparison rather than feature detection as was done in PTAM and ORB-SLAM. This method analyzes the intensity value of every pixel to compare rather than relying on an algorithm such as speeded up robust features (SURF) or ORB to find a spare subset of features.

Regarding SLAM algorithms with depth and RGB based sensors, RGB-D SLAM provides accurate tracking by mapping features to an exact depth location with the assumption that the RGB and depth sensor are aligned (Endres et al., 2012). ORB-SLAM also incorporated depth cameras to provide a more accurate location of feature points detected within an image (Mur-Artal & Tardos, 2016). A novel approach, originally based on Kinect Fusion

(Newcombe et al., 2011), incorporated the iterative closest point (ICP) algorithm so that depth images could be used as a part of a SLAM algorithm that creates a photorealistic mapping of the environment (Olaf et al., 2015).

## FRAMEWORK DEVELOPMENT

The goal of our framework was to allow the same data to be provided to SLAM algorithms with minimal alteration of the source code required. Since these algorithms may run on a different operating system than the framework, we created a network interface to communicate with the systems. Figure 2 shows the data flow from the test bed software to a SLAM algorithm and the expected response.

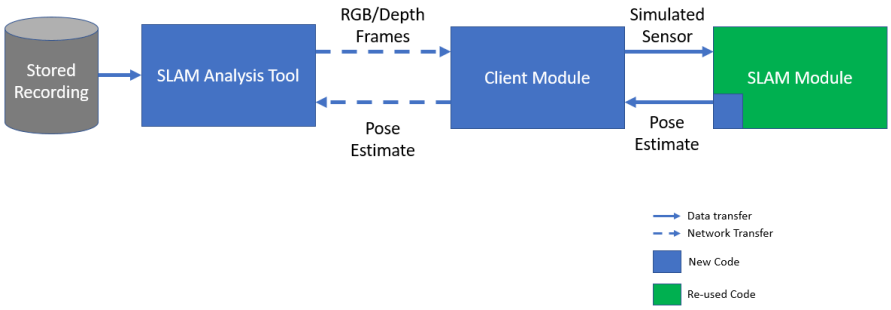


**Figure 2. Data flow diagram of framework.**

The pre-recorded data is read from the hard drive by the SLAM analysis tool which also served as a server for streaming data. A client module was developed on the platform that was running the SLAM source code and handled communication back to the framework's server via the transmission control protocol (TCP). The client module also was responsible for receiving RGB and depth frames and formatting them in a way to simulate a sensor connected to the SLAM module. Finally, the client module received the pose estimate from the SLAM algorithm and provided it back to the analysis tool server.

We were primarily concerned with accuracy analysis in this test and not real-time frame processing speeds. The justification was that a better system with more processing power could be purchased to achieve faster speeds, but the accuracy of the algorithm would remain the same. As such, the SLAM analysis tool did not send the next frame of RGB or depth data until it received a response from all clients with a pose estimate. While the server was designed to cater to several clients at once, during testing only one client module connected at a time.

The majority of the algorithms chosen ran on an Ubuntu platform with the robot operating system (ROS) installed on top of it. As such, the same client module was used which made use of the ROS messaging system both in simulating a sensor and in receiving a pose estimate result. For ease of debugging and testing, an Ubuntu virtual machine was utilized on the same system as the analysis tool. Since real-time processing was not a focus, the processing requirements of the single machine running the test was not a concern.

Once the SLAM analysis tool had received the pose estimate, it compared it to a truth position and orientation value that was recorded at the same time as the sensor frames. Slight transformations were applied to the SLAM pose estimate to align all frames of reference with the truth data. An error value was calculated for each component of position along with total distance and angular error for rotation. This accumulated as the test ran to produce a root-mean-square error (RMSE) for both position and orientation after the recorded data file was finished.

## EXPERIMENTAL DESIGN

### SLAM Algorithm Setup

To test the framework, a subset of modern SLAM algorithms was chosen based on the availability of the source code and unique parameters of the algorithm. Table 1 lists each algorithm to be used, the location of the source code, the platform it was run on and the form of sensor the algorithm works best with.

**Table 1. Subset of algorithms used for experiment.**

| Algorithm | Location | Platform | Sensor |
|---|---|---|---|
| RGB-D SLAM | https://github.com/felixendres/rgbdslam_v2 | Ubuntu, ROS | RGB, Depth |
| LSD SLAM | https://github.com/tum-vision/lsd_slam | Ubuntu, ROS | RGB |
| ORB SLAM Mono | https://github.com/raulmur/ORB_SLAM2 | Ubuntu, ROS | RGB |
| ORB SLAM RGBD | https://github.com/raulmur/ORB_SLAM2 | Ubuntu, ROS | RGB, Depth |
| InfiniTAM Depth | https://github.com/victorprad/InfiniTAM | Windows | Depth |
| InfiniTAM RGBD | https://github.com/victorprad/InfiniTAM | Windows | RGB, Depth |

RGB-D SLAM (Endres, Hess, Sturm, Cremers, & Burgard, 2014) (Endres et al., 2012) was run on Ubuntu with ROS. A client module was created on Ubuntu to stream data to two ROS topics that simulate an RGB and depth sensor. No modifications were required to the RGB-D SLAM source code as it already provides pose estimates from the internal graph optimization to a ROS topic. The client module listens to the ROS topic and replies to the server with the position and orientation information. LSD SLAM (Engel, Schöps, & Cremers, 2014) also ran on Ubuntu with ROS and was configured to listen to the ROS topics created by the client module's simulated RGB sensor. A minor modification was made to the source code to transmit the latest pose estimate back to a ROS topic that the client module was listening to. ORB SLAM (Mur-Artal, Montiel, & Tardos, 2015) (Mur-Artal & Tardos, 2016) ran the same as LSD-SLAM with a similar slight modification to transmit the pose estimate to a ROS topic. ORB-SLAM was run with the default configuration of feature extraction and vocabulary.

The InfiniTAM system (Prisacariu et al., 2014) (Olaf et al., 2015) ran on the Windows platform, the same as the SLAM analysis tool, but still made use of the network system to transmit frames and receive replies. Since ROS was not an option, the client module was built directly into InfiniTAM using its own module system. A streaming module was built as an input system that would be tried if all other sensor connections failed. This module received frames and provided them to InfiniTAM as though it came from a sensor. The code was then modified to take the latest pose estimate and provide it back to the stream module so that it could be sent to the SLAM analysis tool. An example of the InfiniTAM, and all other SLAM algorithms, running on the same data is available in Figure 3.

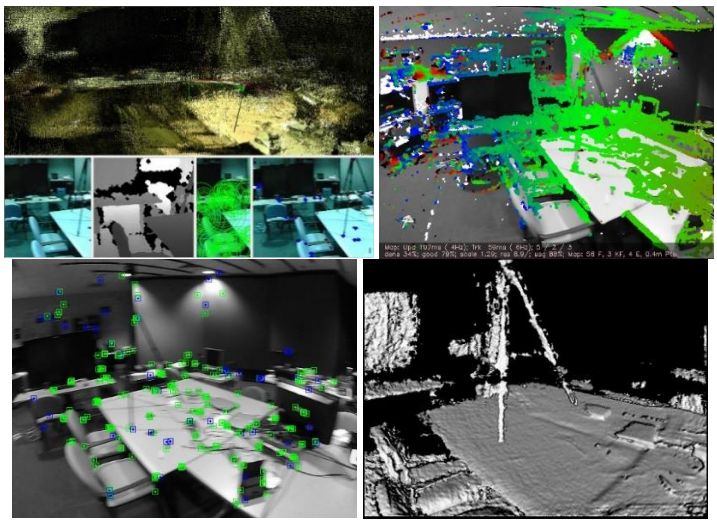

**Figure 3. An example of each SLAM algorithm processing the same data. From top left to bottom right is RGB-D SLAM, LSD-SLAM, ORB-SLAM and InfiniTAM.**

**Hardware Devices Used**

To record data for the experiments we made use of a customized head mounted display (HMD). The hardware chosen was low-weight, low-cost, commercial off the shelf (COTS) equipment meant to represent different configurations that may be available in a lab environment. The base HMD used was an Oculus Rift (Oculus VR LLC, 2017) Development Kit 2 (DK2) which provided inertial measurement unit (IMU) data. Since this was not a desktop use of the HMD, a camera was not available to provide accurate position and orientation data available from the Oculus API. Primarily the HMD was used to hold the other equipment in a manner realistic of a modified COTS HMD.

For sensors, we made use of the Intel RealSense (Intel Corporation, 2017) R200, which provided aligned RGB and depth frames. The frames were observed to be noisier than the other sensors used and provided the capability to evaluate the importance of frame alignment versus frame noise. We also used a Structure IO (Occipital Inc, 2017) sensor which provided a decent range and low noise depth frame. Two wide angle webcams were also attached to record a clean wide angle RGB image. Originally, two were attached for stereo data analysis, but this was not utilized in this experiment.

An HTC Vive (HTC, 2017) controller was used for truth data recordings. It was observed that with the light house setup accurate, low-noise data could be recorded as the user moved around the lab environment. The controller was attached to the crown of the user's head as it did not fit comfortably onto the Oculus Rift HMD. The Vive HMD was not used as its tracking would potentially have been affected by the mounting of sensors to the front of the system. Figure 4 shows the constructed HMD for this experiment. A 3D printer was used for the construction of the sensor mounts.

**Figure 4. Oculus Rift HMD modified with sensors used in the experiment.**

The SLAM analysis tool was developed to have two modes. In the first mode, data gathered from each of the above sensors played if the sensor was connected to the computer. The user could then select a file path and press the record button to gather all data from each sensor. A separate file was created for each sensor and used a custom file format that consisted of a specialized header followed by each frame of data. A playback feature was also created.

In the second mode, pre-recorded data could be selected by an operator and sent to a SLAM algorithm via any clients connected. As the clients responded with pose estimates, error values were calculated and all data was logged to a history file. Once every sensor file reached its final frame, the SLAM analysis tool calculated the RMSE values and saved them off to a different tab of the log file. Extensible markup language (XML) was used for the log files so that they could be viewed with Microsoft Excel and have full functionality available. The RMSE was calculated using equation 1

$$RMSE(x) = \sqrt{\frac{\sum_{t=1}^{t=n}\left(\left|\hat{x}_t - x_t\right|\right)^2}{n}} \qquad (1)$$

where $\hat{x}_t$ is the truth value taken at time t and $x_t$ is the pose estimate value. The RMSE was calculated for the distance of the position vector and the angular delta between quaternions. The angle was determined with equation 2

$$Q_{error} = Q_t * \hat{Q}_t^{-1}$$
$$\theta = 2 * \arccos(Q_{error}.w) \quad (2)$$

where $Q_t$ was the quaternion from the pose estimate observed at time t and $\hat{Q}_t^{-1}$ was the inverse of the truth data.

Furthermore, the test bed and analysis tool was modified to allow the selection of which hardware sensor to use in a test. This allowed us to evaluate how different configurations affected the outcomes and to analyze what should be prioritized when selecting hardware. The hardware pieces were classified as high noise RGB (RealSense R200), low noise RGB (wide angle webcam), high noise depth (RealSense R200) and low noise depth (structure IO). Also, the permutation of aligned cameras (RealSense R200) and unaligned cameras (webcam and structure IO) were evaluated.

## PRELIMINARY RESULTS

The first set of data recordings to be run through the system involved movement and orientation changes on one axis. This was used for alignment of each SLAM algorithm's coordinate system with the truth data. Another alignment data set used was for the user to look to the left and move in that direction. This was used to make sure all translations were in a world coordinate system. Once properly aligned, it was expected for most algorithms to handle these basic movements well and have a low error. Another data set was recorded in which the user casually walked around the lab environment while looking around. This was meant to indicate an average test case where a user is not exhibiting the full agility of their head movements, but is acting without care of the system they are using. Figure 5 show two examples of the tracks a user walked as recorded from the truth data.

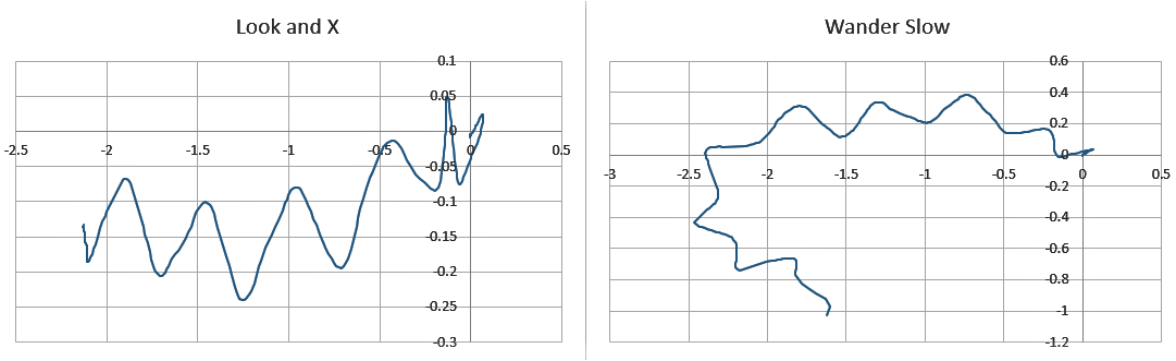


**Figure 5. Two examples of recorded translation data. In the left track the user looked to the left then moved in that direction. In the right track the user rotated and moved to the left, then turn and went to the back of the lab.**

After recording these basic data sets and making sure all coordinate systems agreed, the error could be properly calculated on every SLAM algorithm regarding the truth data recorded. Table 2 shows the statistics of RMSE values across all SLAM-hardware permutations.

**Table 2. Statistics of RMSE values for data sets across all algorithm-hardware permutations.**

| | | X Only | Y Only | Z Only | Yaw Only | Pitch Only | Roll Only | Look and X | Wander Slow |
|---|---|---|---|---|---|---|---|---|---|
| Position | Min (meters) | 0.142m | 0.057 m | 0.239 m | 0.196m | 0.092m | 0.082m | 0.502m | 0.998m |
| | Max | 3.47m | 0.721m | 1.179m | 2.248m | 9.47m | 0.387m | 4.11m | 4.117m |
| | Average | 0.601m | 0.223m | 0.419m | 0.606m | 1.766m | 0.168m | 1.62m | 2.972m |
| | Std Dev | 0.796m | 0.189m | 0.212m | 0.5727 | 2.94m | 0.09m | 0.771m | 1.045m |
| Orientation | Min (degrees) | 1.042 | 0.823 | 1.4542 | 16.791 | 9.163 | 6.858 | 10.215 | 22.45 |
| | Max | 96.564 | 155.808 | 93.184 | 113.46 | 146.757 | 55.742 | 147.462 | 147.86 |
| | Average | 19.067 | 22.403 | 12.172 | 50.07 | 49.133 | 13.701 | 58.849 | 73.11 |
| | Std Dev | 25.195 | 48.975 | 21.436 | 35.377 | 50.579 | 11.819 | 45.209 | 40.23 |

To correlate the above data with the algorithm-hardware permutation, table 3 shows for each data set which hardware and algorithm performed best and which performed worse. These can be tied directly to the minimum and maximum values from table 2.

**Table 3. Correlation of minimum and maximum errors to algorithm-hardware permutation**

| | | Best | | Worse | |
|---|---|---|---|---|---|
| | | Algorithm | Hardware | Algorithm | Hardware |
| Position | X Only | ORBSLAM RGBD | Webcam RGB Realsense Depth | RGBD-SLAM | Webcam RGB Structure Depth |
| | Y Only | LSDSLAM | Webcam RGB | InfiniTAM RGBD | Realsense RGB Realsense Depth |
| | Z Only | RGB-D SLAM | Webcam RGB Structure Depth | ORBSLAM Mono | Webcam RGB |
| | Yaw Only | ORBSLAM Mono | Webcam RGB | InfiniTAM RGBD | Realsense RGB Realsense Depth |
| | Pitch Only | ORBSLAM RGBD | Webcam RGB Structure Depth | InfiniTAM RGBD | Realsense RGB Realsense Depth |
| | Roll Only | RGB-D SLAM | Realsense RGB Structure Depth | RGB-D SLAM | Webcam RGB Realsense Depth |
| | Look and X | LSDSLAM | Realsense RGB | ORBSLAM Mono | Webcam RGB |
| | Wander Slow | LSDSLAM | Realsense RGB | RGB-D SLAM | Webcam RGB Structure Depth |
| Orientation | X Only | ORBSLAM RGBD | Realsense RGB Structure Depth | InfiniTAM Depth | Realsense Depth |
| | Y Only | ORBSLAM RGBD | Realsense RGB Structure Depth | InfiniTAM Depth | Realsense Depth |
| | Z Only | ORBSLAM RGBD | Realsense RGB Realsense Depth | InfiniTAM Depth | Realsense Depth |
| | Yaw Only | InfiniTAM RGBD | Realsense RGB Structure Depth | RGB-D SLAM | Realsense RGB Realsense Depth |
| | Pitch Only | ORBSLAM RGBD | Realsense RGB Structure Depth | InfiniTAM RGBD | Webcam RGB Structure Depth |
| | Roll Only | InfiniTAM RGBD | Webcam RGB Structure Depth | ORBSLAM Mono | Webcam RGB |
| | Look and X | LSDSLAM | Realsense RGB | RGB-D SLAM | Realsense RGB Structure Depth |
| | Wander Slow | ORBSLAM RGBD | Realsense RGB Structure Depth | InfiniTAM Depth | Realsense Depth |

These preliminary results reveal that the errors present in a SLAM algorithm can be very large (several meters at times) even with simple human movements. Note that human movements are naturally more agile and can contain more error than the robotic movements the system was designed for. For example, even a relatively slow head turn may be faster than the robotic system an algorithm was tested against. Also, some algorithms would not provide a return pose estimate if tracking became lost. For these scenarios, the sensor frame was tracked as lost and the RGB or depth feed would proceed. Any permutation that had more than 25% of the frames missing was not included in this data analysis.

When yaw and pitch movements are a part of the data set, errors were observed to be high in both position and orientation suggesting that the loss of previously recognized features is driving pose estimate errors. When the frame stays centered, such as the roll only movement, overall errors were very low.

By examining table 3, ORB-SLAM RGBD appears to be more likely to have the best performance and in the raw data it often came in as one of the top three algorithm permutations tested. InfiniTAM, either RGB-D or depth, appears to be the worse, but it should be noted that InfiniTAM does not drop frames or fail to provide a pose estimate, it always provides an estimate even if it has a high error value. When InfiniTAM maintained tracking it performed very well, suggesting that a re-localization system may be a vital addition to the process.

For frames lost, the only permutations with over 25% lost for this data set was ORB-SLAM Mono with a RealSense RGB, RGB-D SLAM with a RealSense depth and either webcam or Realsense RGB, and for some data sets LSD-

SLAM with a RealSense RGB. This suggested that noisy data can lead to very low performance if it is the primary data that the system relies upon.

It should also be noted that very few systems attempted to re-localize or to fix an estimate that was already off track. As such, error values accumulated over time as the track stayed away from the truth data. This accumulation problem was left in to the calculations since the continuous error would be present in an augmented reality scenario and noticeable by a user, even if the system at the current time was tracking well.

## CONCLUSION AND FUTURE WORK

In this paper, we showed the creation of a framework for the testing of SLAM algorithms with minimal intrusion on the original source code. A subset of algorithms was downloaded and modified to fit the framework with ease and ran against a set of pre-recorded data. Results were gathered through the analysis tool and provided guidance for how SLAM algorithms need to be improved and what hardware configurations to consider when using some SLAM algorithms.

The experiment we ran was a small data set that tested not just the SLAM algorithms, but the framework itself. In the future, a more detailed experiment should be run with more data sets that represent human movements against the latest SLAM algorithms, including updates that have been made to the algorithms in this paper. Furthermore, this experiment examined hardware configurations, but this resulted in several permutations of data to sort through and in the future, may be best performed as its own test. When testing SLAM algorithm performance, the ideal hardware may be utilized of low-noise and aligned color and depth sensors to minimize the variables being tested.

For SLAM algorithm usage with a head mounted display, we saw that there is still a long way to go to have accurate tracking in a marker-less environment. With even mild movements of the human head, large errors quickly became present in the tested algorithms, particularly as the head rotated. This makes sense since every SLAM algorithm is, in some form, comparing the current frame with a previous one, whether via direct comparison or a sparse set of features. If the present frame changes too fast, features cannot be properly matched and the deltas looked at to update the pose. In our future work, we will consider a camera that does not lose features as the head rotates, such as a 360-degree camera and how it may be integrated into existing SLAM research or utilized with a HMD.

## REFERENCES

Azuma, R. (1997). A survey of augmented reality. *Presence: Teleoperators and virtual environments, 6*(4), 355-385.

Chekhlov, D., Gee, A., Calway, A., & Mayol-Cuevas, W. (2007). Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. *IEEE and ACM International Symposium on Mixed and Augmented Reality*, 1-4.

Comport, A., Marchand, E., Pressigout, M., & Chaumette, F. (2006). Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions on visualization and computer graphics, 12*(4), 615-628.

Davison, A. J., & Murray, D. W. (2002). Simultaneous localization and map-building using active vision. *IEEE transactions on pattern analysis and machine intelligence, 24*(7), 865-880.

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., & Burgard, W. (2012). An evaluation of the RGB-D SLAM system. *Robotics and Automation (ICRA)*, 1691-1696.

Endres, F., Hess, J., Sturm, J., Cremers, D., & Burgard, W. (2014). 3-D mapping with an RGB-D camera. *EEE Transactions on Robotics, 30*(1), 177-187.

Engel, J., Schöps, T., & Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. *European Conference on Computer Vision*, 834-849.

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 1231-1237.

HTC. (2017, May 20). *Vive*. Retrieved from Vive web site: https://www.vive.com/us/

Hughes, C., Stapleton, C., Hughes, D., & Smith, E. (2005). Mixed reality in education, entertainment, and training. *IEEE computer graphics and applications, 25*(6), 24-30.

Intel Corporation. (2017, May 20). *RealSense r200*. Retrieved from RealSense developer site: https://software.intel.com/en-us/realsense/r200camera

Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. *ISMAR*, 225-234.

Knerr, B. W., & Lampton, D. R. (2005). *An assessment of the virtual-integrated MOUT training system (V-IMTS).* Orlando: ARMY RESEARCH INST FOR THE BEHAVIORAL AND SOCIAL SCIENCES.

Knerr, B. W., Lampton, D. R., Thomas, M., Corner, B. D., & Grosse, J. R. (2003). *Virtual environments for dismounted soldier simulation, training, and mission rehearsal: Results of the FY 2002 culminating event. .* Orlando: ARMY RESEARCH INST FIELD UNIT.

LaViola, J. J., Williamson, B., Brooks, C., Veazanchin, S., Sottilare, R., & Garrity, P. (2015). Using augmented reality to tutor military tasks in the wild. *Interservice/Industry Training, Simulation, and Education (I/ITSEC)*, 1-10.

Livingston, M. A., Rosenblum, L. J., Julier, S. J., Brown, D., Baillot, Y., Swan, J. E., . . . Hix, D. (2002). *An augmented reality system for military operations in urban terrain.* Washington D.C.: NAVAL RESEARCH LAB.

Mur-Artal, R., & Tardos, J. D. (2016). ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *arXiv*.

Mur-Artal, R., Montiel, J. M., & Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics, 31*(5), 1147-1163.

Nardi, L., Bodin, B., Zia, M., Mawer, J., Nisbet, A., Kelly, P. H., . . . Furber, S. (2015). Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM. *Robotics and Automation (ICRA)*, 5783-5790.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., . . . Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. *Mixed and augmented reality (ISMAR)*, 127-136.

Occipital Inc. (2017, May 20). *Structure Sensor*. Retrieved from Structure Sensor web site: https://structure.io/

Oculus VR LLC. (2017, May 20). *Oculis Rift Development Kit 2 (DK2)*. Retrieved from Oculus web site: https://www3.oculus.com/en-us/dk2/

Olaf, K., Prisacariu, V. A., Ren, C. Y., Sun, X., Torr, P., & Murray, D. (2015). Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics, 21*(11), 1241-1250.

Prisacariu, V. A., Kähler, O., Cheng, M. M., Ren, C. Y., Valentin, J., Torr, P., . . . Murray, D. W. (2014). A framework for the volumetric integration of depth images. *arXiv preprint* .

Witmer, B., Bailey, J., & Knerr, B. W. (1995). *Training Dismounted Soldiers in Virtual Environments: Route Learning and Transfer.* Orlando: ARMY RESEARCH INST FOR THE BEHAVIORAL AND SOCIAL SCIENCES.