# Adaptive, Real-Time Visual Simultaneous Localization and Mapping

Brian Clipp[1], Christopher Zach[1],Jongwoo Lim[2], Jan-Michael Frahm[1] and Marc Pollefeys[1,3]

Department of Computer Science[1]
The University of North Carolina
Chapel Hill, NC, USA
{bclipp,cmzach,jmf}@cs.unc.edu

Honda Research Institute USA, Inc.[2]
Mountain View, CA, USA
jongwoo.lim@gmail.com

Department of Computer Science[3]
ETH Zurich, Switzerland
marc.pollefeys@inf.ethz.ch

## Abstract

*In this paper we present a real-time simultaneous localization and mapping system which uses a stereo camera as its only input. We combine the benefits of KLT feature tracking, which include high speed and robustness to repetitive features, with wide baseline features, which allow for feature matching after large camera motions. Updating the map of feature locations and camera poses is considerably more expensive than performing KLT tracking. For this reason we use the optical flow measured by the KLT tracker to adaptively select key frames for which we do a full map and camera pose update. In this way we limit the processing to only "interesting" parts of the video sequence. Additionally, we maintain a consistent scene scale at low cost by using a GPU implementation of multi-camera scene flow, a generalization of KLT to the motion of image features in three dimensions. The system uses multiple sub-maps; scalable, bag of features recognition and geometric verification to recover from motion estimation failure or "kidnapping". This architecture allows the robot to grow the existing map online and in real time while storing all of the data necessary for an off-line optimization to complete loops. We demonstrate the robustness of our system in a challenging indoor environment that includes semi-reflective glass walls and people moving in the scene.*

## 1. Introduction

The autonomous, multi-purpose robot is an enduring dream of both science fiction aficionados and the general public. These robots might take over the day to day drudgery of domestic labor, clean up hazardous waste or assist the disabled among many possible tasks. To achieve complete autonomy in environments previously unknown to the robot a robot must be able to create a map of its environment while maintaining its pose within the map. This process is known in the robotics literature as simultaneous localization and mapping (SLAM).

A robot must use sensors to measure its environment as part of the SLAM process. These sensors may include wheel encoders, LIDAR sensors, acoustic range sensors or cameras. Cameras are an attractive option because of their low cost, low power use and passive nature. When cameras are used as the primary sensors the SLAM process is known as visual SLAM (VSLAM).

We have developed an online, real-time visual slam system for use in humanoid robots which estimates the full six degree of freedom pose of the robot. Our system performs both Kanade-Lucas-Tomasi (KLT) style differential feature tracking, and wide baseline (SIFT [12]) feature extraction/matching on the graphics processing unit (GPU). Combining the strengths of both methods speeds the SIFT feature matching process, makes our system more robust to repetitive features and allows us to detect previously visited ares in the environment (loop detection) using a vocabulary tree approach [15].

We can hide the relatively higher cost of SIFT extraction/matching and 3D pose estimation by performing those operations only for key frames. Key frames are selected based on the optical flow measured by the relatively faster KLT feature tracking. This novel, adaptive, optical-flow-based, approach to key frame selection limits the majority of the computation to more informative frames in the sequence.

The remainder of this paper will first give some background on research in visual SLAM and some recent work in the area. We will then describe our system in detail and justify the novel aspects of its design. Experimental results will then be given for a video sequence from a challenging environment which includes semi-transparent glass walls and people moving in the scene. Finally, we will conclude with our observations of open research problems that must be solved to make VSLAM a truly viable option for guiding autonomous robots.

## 2. Background

Visual simultaneous localization and mapping has been a topic of interest in the vision community for more than a decade. Early work includes that of Azarbayejani and Pentland [1] who used an extended Kalman filter framework to estimate the camera's motion, scene structure and the camera's focal length simultaneously. This seminal work does not operate in real time. Davison was the first to present a real time VSLAM system using a monocular camera in [5]. Davison's system is able to operate in small, office scale environments in real time, completing loops of this limited scale. However, as the size of the environment grows, the cubic scaling complexity of the extended Kalman filter with map size makes it too slow for real time applications.

Eade and Drummond presented the first real time particle filter approach to monocular VSLAM in [7]. Their system could also map small, office scale environments in real time. As a particle filter their approach can theoretically complete loops in environments of any size. With a particle filter the hope is that at least one particle in the distribution contains a correct representation of the map with the loop completed. For this to be the case over large and complex environments, the number of particles must be too large for real-time processing. This limitation is probably the reason why the particle filter has not been in use in many recent VSLAM systems.

Roboticists have made many strides in performing SLAM using laser ranging (LIDAR), acoustic and other sensors to map large environments [17]. Vision researchers have recently focused on extending VSLAM systems to larger, more complex environments than a desktop or a single room as well. To extend VSLAM to larger areas a method is needed to re-detect previously visited areas. A brute force approach would be to match all wide baseline features from all images to find loops. This is much too slow for real-time operation.

Nistér and Stewenius introduced vocabulary tree methods to find visually similar images in [15]. A vocabulary tree quantizes feature vectors (SIFT descriptors in our case) into visual words which are thought to be visually similar. The tree is pre-trained on a large set of general images so that it's quantization can be used on most scenes. A histogram approach taking into account the infrequency of certain visual words in the scene is used to determine likely matching images. More recent work on appearance based scene matching has been done by Cummins and Newman [4]. Their "FAB-MAP" approach learns a generative model of scene appearance and in contrast to previous approaches uses the co-appearance of features in a scene to reason about the likelihood of a scene match.

Once loops are detected in the robot's path the accumulated error in the path must be corrected to complete the loop. Loop correction in most vision-based approaches to SLAM involves some variant of sparse bundle adjustment [18] or sub-mapping. Bundle adjustment treats the process of generating a globally accurate map as a non-linear minimization problem. The minimization problem's objective function measures the difference between the expected and measured projections of the features in the scene. When the cameras are calibrated intrinsically (known focal length, center of projection, skew and pixel aspect ratio) the minimization method adjusts the feature positions and camera poses to minimize this reprojection error.

The sparsity of the bundle adjustment problem has lead to many clever optimizations such as using the Schur complement and other sparse matrix techniques. However, bundle adjustment remains orders of magnitude too slow for use in an online system if it is to globally correct the map between frames. Recently Klein and Murray [11] have demonstrated a parallel tracking and mapping (PTAM) framework. In this framework visual odometry [14] is performed in one real-time thread while a second thread performs global corrections to the map such as loop completion in an offline, background manner. In this way the map is kept locally correct in real time as new images are added but global corrections such as loop completion can be performed without real time constraints.

Sub-mapping is another option for dealing with global loop completion. In sub-map approaches such as that of Clemente *et al*. [3] a complete map is made up of many sub-maps connected by geometric transformations. Loops can be completed by adjusting the transformations between sub-maps while holding the camera poses and features within the sub-maps fixed. One major drawback of this approach is that all of the error accumulated in the sub-maps is corrected in the transformations between sub-maps. This forces all of the error into just a few parts of the map. Additionally, this loop closing method has not been demonstrated yet in a real time system.

Our system combines what we see as the best approaches in the literature to detect, track and match features; estimate the relative motion of the stereo camera since the previous time; detect loops in the robot's path and correct large scale drift in the map.

## 3. System Description

Our VSLAM system's operations can be divided into temporally local and global mapping operations. The temporally local operations estimate the change in the robot's pose since the previous frame. Sub-modules include differential feature tracking, 6DOF pose estimation, SIFT feature extraction and local geometry guided SIFT matching. Global mapping operations try to detect when the robot views a previously mapped area and correct the robot's pose to the existing map. Our system uses a map of sub-maps

approach to store our representation of the environment. At any time the robot can be moving in any one submap. After each new key frame we attempt to join the current sub-map to the other sub-maps given the new SIFT features extracted from the current frame. Using sub-maps allows us to both recover from being lost (the kidnapped robot problem) and to recover from tracking failures. These can occur due to excessive camera motion or a scene that is devoid of features such as a white wall. Figure 1 is a flow chart of the VSLAM system. In that chart temporally local operations are shown in the left column and the other two columns contain the global mapping operations.

### 3.1. Temporally Local Operations

Temporally local operations include all of the sub-modules that function in the calculation of the robot's pose relative to the last key frame. This begins with radial undistortion, followed by differential feature tracking. If there is sufficient optical flow to have another key frame, SIFT features are extracted and the 3D relative pose is estimated using the KLT features. A windowed bundle adjustment is performed on the relative pose estimate. SIFT features are then matched using the relative geometry calculated from the KLT feature correspondences as a guide. Since we use a windowed bundle adjustment these new SIFT correspondences will be included in the bundle adjustment while processing the next key frame. This completes the operations needed to perform what is essentially visual odometry [13]. We will now discuss each of these sub-operations in detail.

Our radial undistortion process is based on the non-parametric radial undistortion method of Hartley and Kang [10]. We chose this method because of problems we encountered when using cameras with large radial distortion and a polynomial distortion model. This method first finds the center of distortion and then fits a radially symmetric non-parametric curve to the radial distortion of a camera. The only other assumption made about the distortion is that it increases monotonically with distance from the center of distortion. The undistortion method is well suited to wide angle cameras whose distortion cannot be readily modeled by a polynomial function but works well on narrow field of view cameras as well. We use a GPU program to correct for the image's distortion. We store the undistortion map as a set of matched samples of distorted and undistorted radius as well as the center of distortion. Between samples we use interpolation to arrive at a continuous function.

The second step in our process, and the only step other than radial undistortion that happens in every frame, is differential feature tracking. One might wonder why we do both differential, KLT style feature tracking and also use SIFT features. By using both feature measurement methods we combine their strengths while avoiding their weaknesses. KLT feature tracking is an order of magnitude faster than SIFT extraction. However, KLT features cannot be matched over wide baselines, making KLT features unsuitable for detecting previously visited areas in the environment. We can use the optical flow measured by the KLT feature tracker to adaptively select key frames that are significantly different than the last key frame. We then only extract SIFT features and do 3D pose estimation on key frames. In this way we are able to hide the higher cost of SIFT extraction and updating the 3D pose by only performing those operations on select key frames in the sequence.

We use our own NVIDIA CUDA implementation of multi-camera scene flow [6] to perform differential feature tracking. Multi-camera scene flow is an extension of KLT tracking into three dimensions. The process can be divided into feature extraction, which occurs only in key frames, and feature tracking, which is performed between all frames. Feature extraction begins when corner features are extracted from the image using the minimum eigenvalue in the structure tensor to measure corner strength. Non-maximal suppression is performed to find only the strongest response for a given corner feature. We then use a parallel, sparse stereo matching algorithm to perform normalized cross correlation based matching across the stereo head's two cameras. The matched features are then triangulated. Corner features are tracked by determining how the feature must have moved in three dimensions in front of the camera to have generated the image measurements. This parametrization enforces the epipolar constraint between the stereo images with only slightly more than double the cost of two independent standard KLT feature tracking processes.

In addition to improved speed, using both KLT and SIFT together eliminates the all-to-all matching of SIFT descriptors which is a very costly operation. Instead we can match SIFT features using the geometry calculated with the KLT features as a guide as well as the know geometry of the stereo head. This has the added benefit of reducing problems with miss-matches due to repetitive features as well.

Our system uses KLT feature correspondences in a RANSAC framework followed by a windowed bundle adjustment [8] to find an initial estimate of the camera's motion. Because KLT features are tracked using a small image motion assumption repetitive features in the scene are unlikely to be matched in the video over time. Of course the sparse stereo match when initializing features in multi-camera scene flow can still fail, but at least we can eliminate incorrect temporal feature matches.

Since SIFT feature are extracted in the image and then typically matched based on descriptor distance alone, they can yield unreliable matches when there are repetitive features in the scene. Lowe suggests only matching features that are sufficiently distinct from other features in the scene [12]. Unfortunately, this eliminates valuable infor-

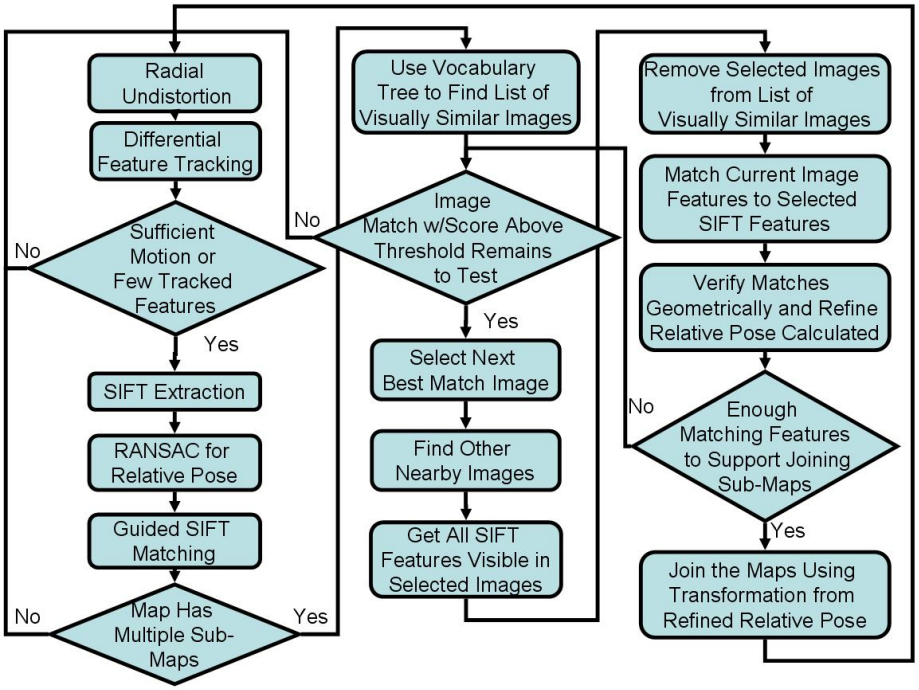


Figure 1. Flow chart of the VSLAM System. Temporally local operations are shown in the left column while global mapping operations are in the middle and right columns.

mation. The 3D geometry calculated with the KLT feature correspondences gives us a measure of where SIFT features should project in the image and hence allows us to disambiguate features that have similar descriptors (repetitive features).

Figure 2 illustrates this process. In the figure the 3D SIFT feature is shown as the large red dot and it has been triangulated from the previous stereo camera pair. The system has used the differential KLT features in RANSAC to estimate the system's rotation and translation ($R$ and $T$). Two features are in the left image of the second stereo pair that might match the 3D SIFT feature. We can use the projection of the 3D feature into the left image to guide feature matching. Note that this guided matching can be used to match features that were seen not only in the recent past but also can re-detect features that were stored in the map on a previous trip through an area. Re-detecting previously mapped SIFT features greatly reduces the accumulated error in the reconstruction as shown in Figure 6.

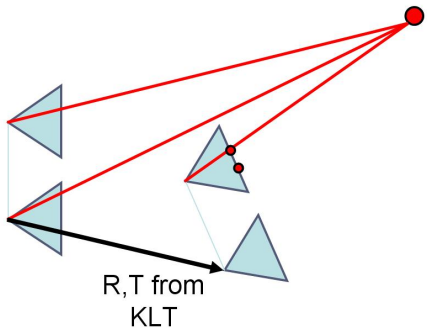


Figure 2. Repetitive SIFT feature disambiguation. Using the expected projection of the 3D SIFT feature (large red dot) and the camera motion measured with the differential KLT features we can determine which of the two features in the image (small red dots) should match the 3D SIFT feature.

### 3.2. Global Operations

Global operations allow our system to recover from "kidnapping" or loss of tracking and also join newly mapped areas to existing sections of the map. In our system these operations are performed by the same module. It uses a vocabulary tree to find areas of existing sub-maps that are visually similar to the current frame, but not in the current sub-map and geometrically verifies matched SIFT features to determine if two sub-maps should be combined into one.

The fundamental global operations are determining when two sub-maps should be joined because an area of overlap has been found and when to break sub-maps because the robot is moving into a previously unmapped area. Breaking apart sub-maps just to join them together later seems a strange thing to do in a VSLAM system and requires some justification.

Splitting sub-maps provides an effective method to deal with accumulated drift in the robot's path in a real time sys-

tem. Correcting for accumulated error using bundle adjustment is too computationally expensive to do within the real time bounds between two video frames for even relatively small loops. We create a new sub-map when the system stops matching to previously mapped features and is only performing visual odometry. A transformation between the old and new sub-maps is stored to allow the robot to navigate from one to the other. When the system comes back into an area that has been previously mapped and there has not been much error accumulated (measured by the reprojection errors) we can re-join the sub-maps. Otherwise the robot can begin to operate in the old sub-map again by localizing itself to the previously mapped features. In this case a transformation is added between the last robot pose in the new sub-map and the old sub-map it is re-entering. In this way breaking apart the map can deal with the error that accumulates in visual odometry.

The process of joining two sub-maps is shown in Figure 3. In the first pane the separation between the two sub-maps is represented by their green and blue colors. The system is currently operating in the green sub-map. The second pane shows the system finding visually similar images and extracting the features seen in those images. In the last pane the sub-maps are joined by matching the red 3D SIFT features to the features seen in the most recent frame in the current sub-map and geometrically verifying them. Since there is only one sub-map at the end of the process all cameras are shown in blue. Note that if the green sub-map had not started from an unknown position but had branched off of the existing sub-map, we would have to check the reprojection error at both ends of the green camera path before joining the sub-maps.

We begin the sub-map joining process by measuring the visual similarity between the current image and the images in the other sub-maps. Rather than store all of the images we use a bag-of-features approach to represent the images. The features are quantized into visual words using a high speed, GPU implementation of a vocabulary tree [15]. Another possible approach to detecting visually similar areas was developed by Cummins and Murray in [4]. We store inverted files which tell for a given visual word, what frames it was seen in. This set of inverted files grows as the map is extended to reflect the new key frames that are added. When we want to find out what images are similar to the current image we take the SIFT features in the current image, quantize them with the vocabulary tree and then use the inverted files to generate a histogram of likely matching images. We also weight the visual words according to their frequency in the set of images, with less frequent visual words weighed higher in the scoring because they are more visually distinctive. The scoring system returns a list of frames ordered by their likelihood of matching the current frame along with their matching scores. After calculating a list of visually

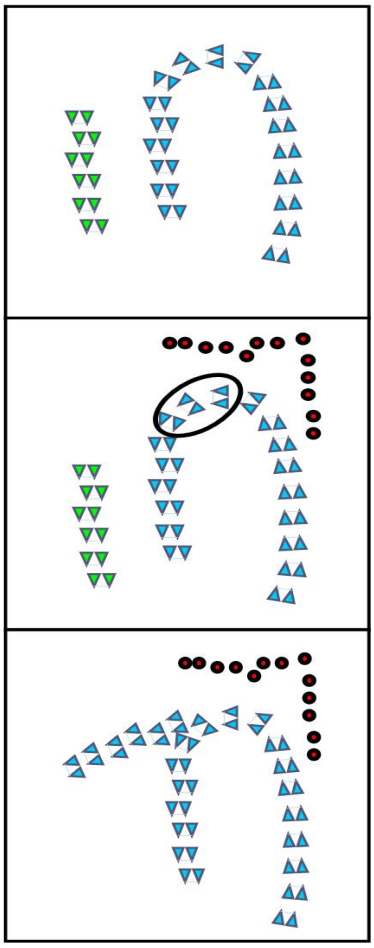

Figure 3. The process of joining two sub-maps. In the first pane the separate sub-maps are represented by the green and blue colors. The system is currently operating in the green sub-map. The second pane shows the most visually similar stereo image at the center of the ellipse and the nearby cameras are also within the ellipse. The SIFT features visible in the images in the ellipse are shown as red dots. In the final pane the SIFT features have been matched to the features extracted in the latest image in the green sub-map, the matches have been geometrically verified and the two sub-maps have been joined using the transformation calculated during geometric verification using the three point perspective pose method.

similar images we remove those images that are in the current sub-map, since those images cannot be used to join the current map to another.

Simply because two images are visually similar does not mean that they show the same scene. To ensure this we must first match the SIFT features using their descriptors and perform a geometric verification. To increase our chances of correctly joining the current sub-map to an existing sub-map we match the features in the current image, not to just the top scoring image, but also to the images that are near it in its sub-map [16]. To do this we find all of the images near the top scoring image and recover the features from the

database that are seen in each of those images.

Each of the SIFT features near the top scoring image has a 3D estimate and also has a SIFT descriptor. Simply because two descriptors map to the same visual word through the vocabulary tree does not mean they necessarily match. We use a GPU based matching algorithm to find the most likely match between the feature descriptors we see in the current image and the descriptors extracted from the map database. The GPU algorithm parallelizes the all-to-all matching of descriptors and critically, performs the dot products of the 128 float long descriptors on the graphics processor. This greatly speeds up the matching process.

Once we have matched the features in the current image to features in the other sub-map we use the three point perspective pose method [9] in a RANSAC [2] framework to geometrically verify the matches. If enough matches are found to be correct, a bundle adjustment is performed starting from the best relative pose solution found using the RANSAC procedure. The two sub-maps are then combined based on the transformation between the camera's pose in the current sub-map and the camera's pose in the other sub-map which was found using the other sub-map's features. Features which were matched and geometrically verified are combined in the database so that their 3D positions reflect the fact that the sub-maps are joined.

Breaking apart sub-maps is a much simpler process. The system simply changes the current sub-map number to one that is unused and if tracking has failed it clears all features in the new sub-map. Otherwise the features seen in the current image are added to the new sub-map. This means that a single feature can have multiple different sets of coordinates, one set for each of the different sub-maps it is stored within. This allows us to break sub-maps without loosing correspondence information between images in different sub-maps.

## 4. Experimental Results

We chose a particularly difficult office environment to demonstrate the system's robustness in this paper. The loop taken by our robot (in this case a Point Grey Bumblebee stereo camera) is approximately eighteen by ten meters. The office has two glass walls, shown in the lower left and center images of Figure 4 and on the map in turquoise in Figure 5. The walls present a large semi-reflective, semi-transparent surface which can cause problems with feature matching. However, our system can find features whose appearance is not viewpoint dependent and robustly estimate the robot's path.

Another challenging aspect of the office environment is the relative sparsity of features on the walls of the hallway shown in the upper left of Figure 4. The lack of texture on the walls means that most corner features, or extrema in difference of Gaussian space do not have a distinctive local patch around them. This would present a problem for SIFT feature matching approaches since they rely on the distinctiveness of the descriptors to accurately match features. However, the differential KLT features are easily tracked from frame to frame and allow the system to operate robustly in this part of the scene. Of course, because there are few reliable SIFT features extracted in this part of the scene sub-maps cannot be joined in this part of the map.

Processing the sequence of video shown here was done at eight frames per second, fast enough to track a camera being moved by an operator. The first pass around the loop in the map was completed in an offline process from pre-recorded data. Once that is done the system can localize the camera to the existing map in real time and also extend the map into new areas. Figure 6 shows how matching currently detected SIFT features with features already in the map (re-detecting them) can greatly reduce the accumulated error in the robot's path.

In the future we plan to make correcting for loops a background process handled by another thread. A major stumbling block to this is the open research problem of how to avoid and deal with local minima in the bundle adjustment. This is one of the reasons that global map correction for loops remains an off-line process requiring user interaction.

## 5. Conclusion

In this paper we have presented an online VSLAM system that both recovers from lost tracking and kidnapping in real time. Our system uses differential feature tracking to adaptively select key frames based on optical flow. For those key frames we extract SIFT features that allow us to detect when the robot enters a previously mapped area. This combined approach gives both high speed and allows for wide baseline feature matching.

Future work on the system will include moving the global map correction using bundle adjustment into a background process. This will allow the system to add new frames to the map at full frame rate while correcting for newly completed loops.

Developing this system has made clear some open research problems to be solved if VSLAM is to work in practice outside the laboratory. The first is primarily a hardware issue. The office environment we tested in has many windows which lead to a very high dynamic range in the scene. Even using auto exposure or auto-gain the dynamic range of the camera cannot match the scene. There are certainly many features in the scene that could not be detected because of the low dynamic range of the cameras relative to the scene, such as items on a desk in front of a window which appear almost black in the sequence because of the bright, sunlit windows behind them.

Another issue is how to deal with changes in the scene. The office we tested in has a set of tables and chairs for a

Figure 4. Sample frames from the left camera of the stereo pair for the office sequence. Note the reflective glass far wall in the lower left image and glass half wall in the lower middle image. Our system is able to robustly handle these challenging scene components.

Figure 5. The camera path (magenta and blue axes) and 3D feature estimates(blue dots) overlaid on a layout of the office environment where we ran our system. Full height walls are shown in black and half-height partitions are shown in gray. The turquoise walls are made of glass.

break area. Moving the chairs after the VSLAM system has mapped the office caused the system to no longer recognize that part of the scene. One method to deal with this is to try to extract features on 3D scene planes rather than in the image as was done by Wu *et al*. in [19]. We will be looking into ways to combine dense 3D geometry with sparse feature detection in future work with the goal of finding immovable objects in the scene such as the walls, the ceiling and floor. The system could then re-localize itself to these immovable features regardless of change in the rest of its environment.

## 6. Acknowledgements

This work was supported in part by Honda Research Institute USA Inc. We would also like to thank Arnold Irschara for the use of his vocabulary tree data.

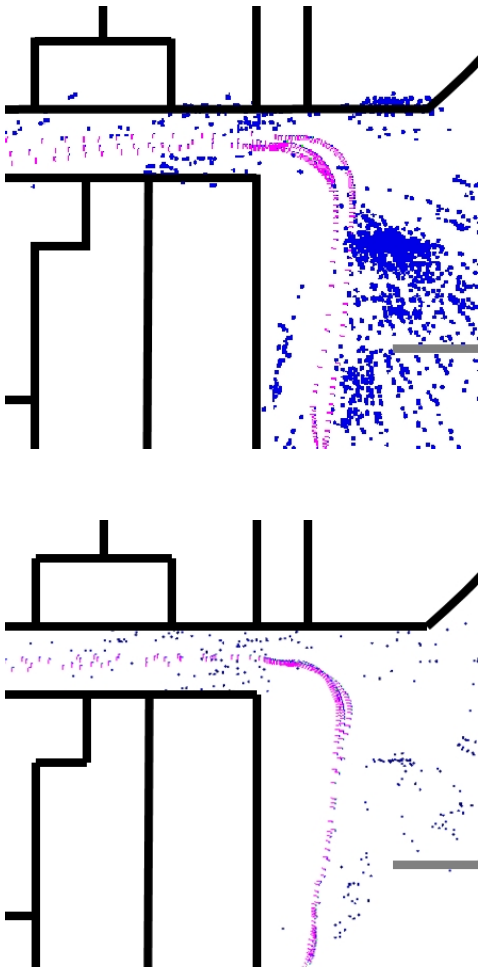

Figure 6. A comparison of the motion estimation using only visual odometry only with no re-detection of SIFT features after they go out of view (top image) and using re-detection of SIFT features (bottom image). Note the overlap in the path when areas are re-detected in the bottom vs. when no re-detection is performed in the top.

## References


[1] A. Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal length. *TPAMI*, 17(6):562–575, Jun 1995.

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

[3] L. Clemente, A. Davison, I. Reid, J. Neira, and J. Tardos. Mapping large loops with a single hand-held camera. In *Robotics: Science and Systems*, June 2007.

[4] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[5] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV*, volume 2, pages 1403–1410, Oct. 2003.

[6] F. Devernay, D. Mateus, and M. Guilbert. Multi-camera scene flow by tracking 3-d points and surfels. In *CVPR*, volume 2, pages 2203–2212, 2006.

[7] E. Eade and T. Drummond. Scalable monocular slam. In *CVPR*, pages I: 469–476, 2006.

[8] C. Engels, H. Stewenius, and D. Nister. Bundle adjustment rules. In *Photogrammetric Computer Vision*, September 2006.

[9] R. Haralick, C. Lee, K. Ottenberg, and M. Nolle. Review and analysis of solutions of the three point perspective pose estimation problem. *IJCV*, 13(3):331–356, December 1994.

[10] R. Hartley and S. B. Kang. Parameter-free radial distortion correction with center of distortion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(8):1309–1321, Aug. 2007.

[11] G. Klein and D. Murray. Improving the agility of keyframe-based slam. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 802–815, Berlin, Heidelberg, 2008. Springer-Verlag.

[12] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, November 2004.

[13] D. Nistér. A minimal solution to the generalised 3-point pose problem. In *CVPR*, pages I: 560–567, 2004.

[14] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. volume 01, pages 652–659, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[15] D. Nistér and H. Stewnius. Scalable recognition with a vocabulary tree. In *CVPR*, pages 2161–2168, 2006.

[16] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, pages 1–7, June 2007.

[17] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.

[18] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer-Verlag, 2000.

[19] C. Wu, B. Clipp, X. Li, J.-M. Frahm, and M. Pollefeys. 3d model matching with viewpoint invariant patches (vips). In *CVPR*, 2008.