Monocular Visual SLAM on the AR.Drone

Julian Straub - JulianStraub@gatech.edu

Problem Description

This special problem investigates in the feasibility of performing monocular visual Simultaneous Localization and Map building (SLAM) on a quadcopter, namely the AR.Drone, to enable autonomous navigation in an outdoor environment. Such an aerial vehicle could help in scenarios where a area is to dangerous or inaccessible for humans. For example the drone could help to build a fast map of a disaster zone, like a collapsed building, to provide first responders with the necessary overview over the situation without risking human lives. Another scenario would be the exploration and surveillance of areas that are contaminated with radiation.

A totally different field for deployment of such a system would be in agriculture to collect data about plant growth and overall condition automatically. Statistics generated from such widespread measurements could help regulate the supply of water and nutrients in a way that the plants always get optimal conditions for growth. This would ultimately result in a bigger harvest, especially in regions that are naturally adversary to growing crop.

A third scenario in which an autonomous quadcopter could be of use is building and park surveillance. Although public buildings are usually well equipped with cameras observing the surrounding area, a fast and movable camera would add a lot of flexibility to the security infrastructure. It could for example be used to continuously track possible intruders once they have been spotted by one of the installed cameras. Public parks present a different challenge, since they are usually large and do not have surveillance infrastructure. That this is a problem, especially when thinking of crime and drug dealing in parks at night, is obvious. A fast and mobile camera on a quadcopter would be a way to monitor public parks and showing presence of police, which would foster security.

Related Work

SLAM is the problem of estimating the robots position while constructing a map of the surrounding environment without any prior information. It is one of the key problems in robotics to enable truly autonomous agents. There are efficient algorithms [6, 15] that are able to solve SLAM given sensors like laser range finders that are able to provide rich data about the structure of the space and enough computational resources to process this data. Those algorithms were developed for ground vehicles, but are not restricted to that.

In fact there has been recent work [10, 2, 1] to adopt those techniques to a flying system. All of those aerial vehicles however rely on laser scanners which require a significant amount of energy and are of considerable weight and cost. Especially the last two facts disqualify the adoption of such a solution, since the AR.Drone does not allow heavy additional load.

There are already attempts [16] to utilize a single camera as the only sensor to provide feedback for the control of a helicopter. This system however does not solve SLAM but uses visual flow to navigate. Monocular visual SLAM is performed by Steder [17] on a flying blimp. This system is limited to slow movements and a small region of surveillance.

Looking at the problem of the pure monocular SLAM problem, where the camera is simply moved by a human, we find recent progress. Eade [7] and Davison [4] are among the first that have been able to demonstrate monocular SLAM in real-time. Both systems perform well in small areas with a quite small number of landmarks.

The latest research in visual SLAM is representing the SLAM problem in a graph that is then optimized to get the most likely map and trajectory. Graph based algorithms give easier insight into the structure of the SLAM problem and allow exploiting conditional independence between the involved random variables. This can lead to faster algorithms compared to the Extended Kalman Filter SLAM systems [18]. A general framework for such graphical inference algorithms is GTSAM, which is being developed at Georgia Tech. It contains a batch [5] and a incremental solver [12] for Smoothing and Mapping (SAM), as the graph based SLAM problem is also called. Examples of systems that utilize graph based inference algorithms to solve visual monocular SLAM have been proposed by Eade [8] and Konolige [13]. Both systems are able to deal with large areas and a high number of landmarks. This is achieved by operations on the graph that reduce the complexity of the optimization problem.

Approach

The AR.Drone quadcopter is equipped with an inertial measurement unit (IMU), a downward- and a forward-

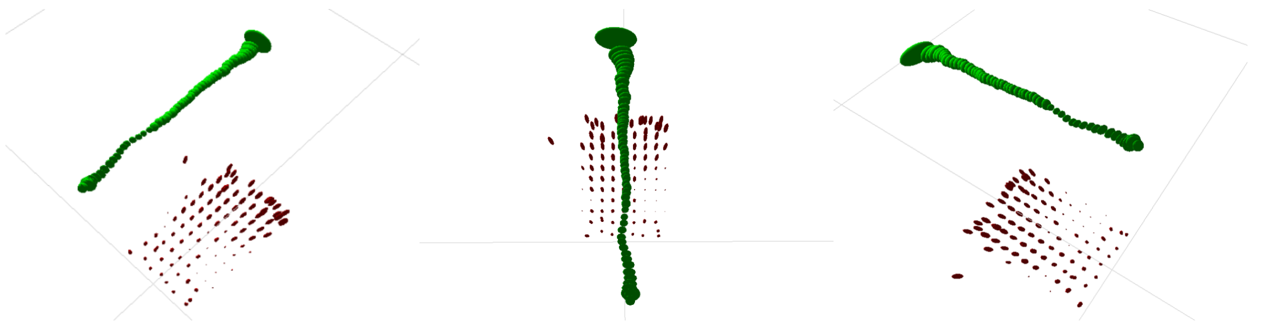
Figure 1: Estimates of camera positions (green ellipses) and feature positions (red ellipses) after performing monocular visual SAM during moving the AR.Drone manually over a checker board.

facing camera and ultrasound altimeter. The fast dynamics of the quadcopter are compensated by its on-board microprocessor and need not to be dealt with. Once the WLAN connection with the AR.Drone is established we are receiving new frames from the down-ward facing camera every 55ms and the latest IMU and ultrasonic sensor measurements every 60ms. For the purpose of easier testing of the involved algorithms a program was created to store all data received from the AR.Drone during a flight. The C++ implementation of monocular visual SAM described in the following does currently only operate on recorded data sets. However, it would be a straight forward implementation to feed the live data streams into the algorithms.

We use SURF [3] features as visual landmarks. From each new frame we find those features using the OpenCV2.2 implementation of this feature extraction algorithm. Data association between the extracted features in two consecutive frames is established using Nearest Neighbor classification on the 128bit SURF feature descriptors. Wrong data associations are eliminated by running Random Sample Consensus (RANSAC) [9] on the set of associations with a pinhole camera model. The frame-to-frame feature matches are then fed into a book-keeping algorithm that keeps track of feature associations over several frames.

In the SAM approach to SLAM the goal is to find the best estimate of trajectory of the camera and the positions of all landmarks given measurements on the landmarks and the pose of the camera. In our case the measurement on the landmarks is the projection of the 2D feature position in image coordinates into the 3D world. This projection is only determined up to scale, since the landmark could be anywhere on the line that originates in the focal point of the camera and intersects at the feature position with the image plane. This scale issue, however, can be fixed using the altimeter readings. The assumption is made, that all features lie in a plane underneath the quadcopter in the measured distance. The IMU data is used to get a measurement on the relative motion between two data packets.

The graphical model that comprises the structure of this

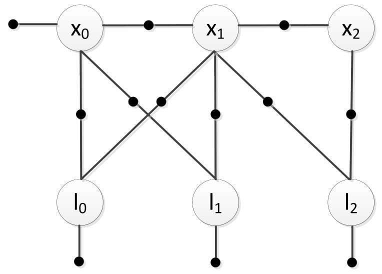

Figure 2: Factor graph of the monocular visual SAM problem, where x_i denotes a camera pose and l_j denotes a landmark position.

SAM problem is depicted in Figure 2 in the form of a factor graph [14]. Each node represents a random variable. The connections, the so called factors, symbolize a mathematical relation between the involved variables. The SAM problem at hand has two different binary factors: A projection factor connecting a landmark l_j with the camera pose x_I in which it was observed and relative pose factors between consecutive camera poses. Also, the assumption that all landmarks lie in a plane is enforced by priors on the landmarks, that constrain their position only in the z-direction but not in the x-y-plane. And finally the first pose x_0 is constrained to take on the height, measured by the altimeter.

Since the sensor data arrives gradually over time in a realistic scenario, the factor graph has to be built incrementally as new measurements arrive from the quadcopter. iSAM [12] is an incremental SAM solver algorithm that is designed to efficiently manage the adding of new factors and variables. We are using the GTSAM C++ implementation of iSAM to build up the SAM problem as we obtain new sensor data and to retrieve the current trajectory and landmark position estimates whenever necessary.

The quadratic error functions that iSAM uses internally, lead to unstable behavior in the presence of wrong data associations. This is because even a small number of outliers can draw the estimated solution away from the real solution when using quadratic error functions. Hence, in order to stabilize the solution, we implemented an algorithm that finds projection factors with a high error and deletes them from the factor graph. This basically is an Expectation Maximization (EM) algorithm where the two classes are correct associated features and wrong associated ones.

Using the elimination algorithm on the current factor graph, it is possible to query the marginal Gaussian distribution on any landmark position or any camera pose. In displaying all the individual position and pose distributions as one-sigma ellipse around the 3D mean in peekabot, we are visualizing the quality of the obtained solution for evaluation purposes.

Evaluation

The evaluation concentrates on the performance of the monocular visual SLAM algorithm measured by the positioning and mapping accuracy as well as computational speed. The accuracy of mapping and localization is shown qualitatively by overlaying the estimates with the real ground truth layout of the area that was traversed. Although this does not give quantitative results, this way of showing results is widely used in the SLAM community. Computational speed of the the main algorithms in the SLAM system is measured on a external 2.2GHz dual core laptop that is in control over the AR.Drone. It has to be noted that while for all the experiments data was collected using the AR.Drone, we processed the data for the evaluation offline.

Figure 1 shows the estimate of the camera trajectory and the feature positions of an experiment in which the AR.Drone was moved manually in a straight line with nearly constant height over a checker board. The relative pose measurements were given as a straight movement with constant speed in constant height, since the AR.Drone does only output the direct pose estimate if it is actually flown. The same approach was taken for the second experiment, but this time the camera's view captured randomly scattered paper. The resulting estimates are displayed in Figure 3. Referr to the two videos submitted with this paper to see the how the final estimate is incrementally built up.

Figure 4 displays the projection of the feature estimates of both experiments onto the respective real image as taken by the AR.Drone. Note that SURF features are not necessarily found on corners. In the case of the checker board SURF features are found in the middle of the squares.

As for the timing results, we found that the dominant time is spent on eliminating factors with high errors. On average the algorithm needs about a second per frame to remove those factors. In the worst case we measured up to three seconds though. Second most time was needed to compute an update of iSAM after adding new factors. Figure 5 displays plots of update times of iSAM for the two different different scenarios that were evaluated.

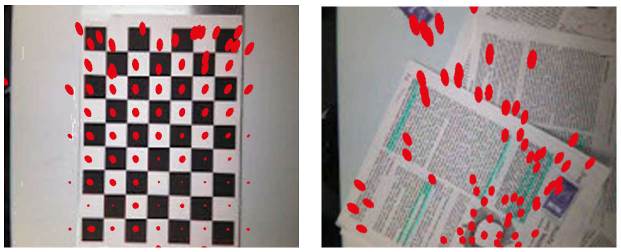

Figure 4: Projection of the feature estimates of both experiments onto the respective real image.

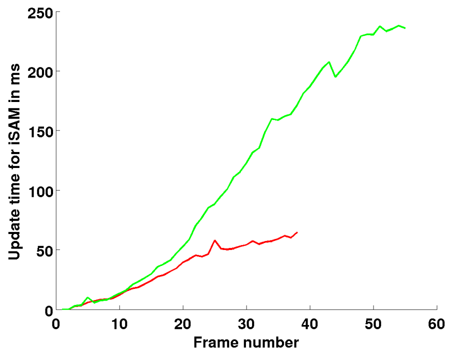

Figure 5: Timing of the update of iSAM in red for the scenario with the scattered paper and in green for the checker board example.

Conclusion

All in all a robust monocular visual SAM system has been implemented. The estimation results displayed in Figures 1 and 3 show qualitatively a good solution for the camera trajectory as well as for the feature positions. By depicting the projection of the feature positions onto the real image in Figure 4 we were able to show that the feature position estimates match the positions in the real underlying images. The fact that the results of the second experiment, which involved cluttered images rather than the regular pattern of a checker board, are of good quality as well, proves the robustness of the system.

While quality and robustness of the visual SAM system could be shown, the timing results are not convincing yet. The main problem was the necessity to remove factors with high error to achieve stability. As mentioned beforehand, the code that performs the elimination of such

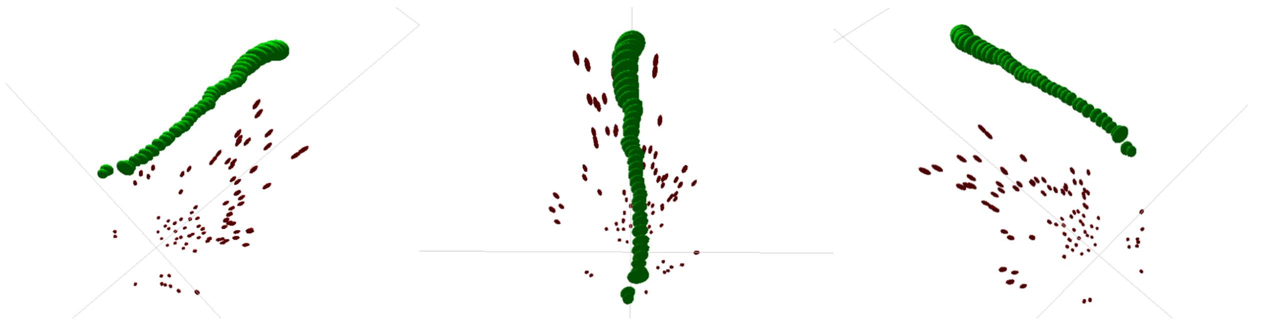

Figure 3: Estimates of camera positions (green ellipses) and feature positions (red ellipses) after performing monocular visual SAM .

factors consumes up to three seconds. This is because it has to compute all errors of all factors in order to find the one with maximal error, which is then removed. After the removal, the whole factor graph has to be resolved by iSAM. Furthermore, most of the time several factors have to be removed per frame. Hence the whole procedure is repeated several times. The core problem here is the usage of the quadratic error function which is not robust against outlier. Deploying a robust error function would make the whole solution robust against outliers caused by wrong data associations and hence eliminate the need to remove factors with high errors.

The next biggest time consumption is caused by iSAM itself during the necessary update of the factor graph at each frame. The monocular visual SAM system would greatly benefit from the improvements made upon iSAM in the second version iSAM2 [11]. iSAM2 features fluid relinearization and incremental variable reordering, which eliminate the need to perform time-consuming complete reordering and relinearization after each update.

References

[1] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy. Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments. In *SPIE*, Orlando, FL, USA, 2009.

[2] A. Bachrach, R. He, and N. Roy. Autonomous flight in unstructured and unknown indoor environments. In *European Conference on Micro Air Vehicles (EMAV 2009)*, Delft, Netherlands, 2009.

[3] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded Up Robust features. *Computer Vision–ECCV 2006*, pages 404–417, 2006.

[4] 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, 2007.

[5] F. Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Reasearch*, 25:2006, 2006.

[6] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17:229–241, 2001.

[7] E. Eade and T. Drummond. Scalable monocular SLAM. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*. IEEE Computer Society, 2006.

[8] E. Eade, P. Fong, and M. E. Munich. Monocular graph SLAM with complexity reduction. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3017–3024. IEEE, 2010.

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communication ACM*, 24:381–395, June 1981.

[10] S. Grzonka, G. Grisetti, and W. Burgard. Towards a navigation system for autonomous indoor flying. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 2009.

[11] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Shanghai, China, May 2011. To appear.

[12] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 2008.

[13] K. Konolige and M. Agrawal. FrameSLAM: From Bundle Adjustment to Realtime Visual Mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, 2008.

[14] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, pages 498–519, 2001.

[15] M. Montemerlo, S. Thrun, D. Roller, and B. Wegbreit. FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1151–1156, San Francisco, CA, USA, 2003.

[16] S. P. Soundararaj, A. K. Sujeeth, and A. Saxena. Autonomous indoor helicopter flight using a single onboard camera. In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, pages 5307–5314, Piscataway, NJ, USA, 2009.

[17] B. Steder, G. Grisetti, C. Stachniss, and W. Burgard. Visual SLAM for flying vehicles. *IEEE Transactions on Robotics*, 24(5):1088 –1093, 2008.

[18] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Real-time monocular SLAM: Why filter? In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2657–2664. IEEE, 2010.