Learning a Context-Dependent Switching Strategy for Robust Visual Odometry

Kristen Holtz, Daniel Maturana, and Sebastian Scherer

Abstract Many applications for robotic systems require the systems to traverse diverse, unstructured environments. State estimation with Visual Odometry (VO) in these applications is challenging because there is no single algorithm that performs well across all environments and situations. The unique trade-offs inherent to each algorithm mean different algorithms excel in different environments. We develop a method to increase robustness in state estimation by using an ensemble of VO algorithms. The method combines the estimates by dynamically switching to the best algorithm for the current context, according to a statistical model of VO estimate errors. The model is a Random Forest regressor that is trained to predict the accuracy of each algorithm as a function of different features extracted from the sensory input. We evaluate our method in a dataset of consisting of four unique environments and eight runs, totaling over 25 minutes of data. Our method reduces the mean translational relative pose error by 3.5% and the angular error by 4.3% compared to the single best odometry algorithm. Compared to the poorest performing odometry algorithm, our method reduces the mean translational error by 39.4% and the angular error by 20.1%.

1 Introduction

Autonomous aerial vehicles are often desired for performing tasks that are dangerous or impossible for humans. From urban search-and-rescue missions to remote exploration of nuclear disaster sites, these tasks often take UAVs to unknown environments that are challenging due to their diverse and dynamic nature. Among these challenges is the likely inability of external communication, including limitations on the availability and reliability of GPS data. This requires all perception,

Kristen Holtz · Daniel Maturana · Sebastian Scherer
Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA, 15213, USA
e-mail: {kholtz, dmaturan, basti}@andrew.cmu.edu

processing, and decision-making to be made onboard. The unpredictability of the environment further contributes to the need for a more robust system that is capable of recovering from unanticipated faults. The infeasibility of considering all possible exception and errors beforehand has led to research in fault-tolerant control (FTC) and fault-tolerant perception [16].

Fault-tolerant perception can pose an especially difficult problem due to the vast diversity of environments that occur in the real world. This diversity means that for many tasks, a single method is rarely the best in all situations; instead, different methods excel in different kinds of environments. This phenomenon was shown in the task of Visual Odometry (VO) by Fang and Scherer [5], who compared the performance of different VO systems using RGB and depth data. They found that VO systems that used both kinds of information performed better, on average, than systems using only depth information. However, in dark or smoky environments, the depth-only systems would fail significantly less often than the other systems.

This motivates the main contribution of this paper, a practical and flexible framework for fault-tolerant state estimation. The main idea of our framework is to use an ensemble of algorithms, and dynamically switch between them as the vehicle moves between environments. Switching is performed by periodically selecting the algorithm expected to be the most accurate in the current context, according to a statistical model of the accuracy of each algorithm. The model is trained to predict the accuracy of the motion estimates of each algorithm as a function of various features describing different aspects of the environment and vehicle state.

We evaluate our framework in a benchmark with data from two sensors covering four challenging, real-world environments. As further contributions, this evaluation empirically shows that:

1. It is possible to improve on the performance (in terms of accuracy and robustness) of the single best algorithm in an ensemble by dynamically switching between algorithms.
2. Algorithm performance is correlated with observable characteristics of the environment and vehicle state, so it is possible to predict which algorithm will perform best in each context from the sensory input.
3. Our proposed switching strategy results in more accurate and robust estimates than any of the individual VO algorithms.

2 Related Work

Several state estimation methods can report some form of variance or confidence estimate together with their state estimate, e.g. Censi's method for ICP [4]. These variances are often used for soft fusion, e.g. in a Kalman Filter [10]. They can also be used for fusion. For example, Tomic et al [14] use the self-reported variances of a stereo odometry algorithm and laser odometry algorithm to switch between them as the vehicle navigated between indoor and outdoor estimates. Compared to our approach, these methods have the advantage of not requiring any extra training step;

however, this comes at a cost in terms of flexibility. These methods cannot take advantage of extra information our method can incorporate seamlessly. Additionally, our method does not require running an algorithm to predict its performance; this can provide significant computational benefits by turning the unused odometry algorithms "off". Finally, only some specific methods can self-report their variance, while our method is applicable to any algorithm, whether or not it has this capability.

Leishman et al [12] propose a system that dynamically switches between different odometry methods based on context-dependent variables. The switching is based on an ad-hoc strategy based on manually selected quality thresholds for each modality. In contrast, our method uses machine learning to learn this strategy, which considerably simplifies adapting the method to different environments, sensors or algorithms.

An algorithm that has several similarities to ours is CELLO [15]. This method predicts a covariance matrix for each method as a function of past training data. The covariance matrix is predicted with a nonparametric estimator similar to nearest neighbors methods. Compared to CELLO, our method makes various choices that make it simpler and more practical. Instead of predicting a full covariance matrix, we predict error magnitudes, which are simpler to learn and sufficient in many cases. We choose a random forest-based regressor, which is faster than and more robust than nearest neighbor methods [3]. In addition, our evaluation is more exhaustive, as it has different and more challenging environments.

3 Approach

The system implementing our proposed method can be decomposed into various components:

Sensor Suite In our framework sensors serve two purposes: to serve as input for the VO estimates and to capture characteristics of the environment that will allow the model to predict which algorithm will be the most reliable in any given context.

Algorithm Ensemble Our method requires a set of base algorithms performing the same task (VO, in this paper). The algorithms should be diverse in terms of their performance characteristics across different environments.

Features In order to describe aspects of the environment that are potentially relevant for predicting accuracy, we extract various features from the sensor data and estimated vehicle state.

Error Prediction Model Using data annotated with ground truth, we train a statistical regression model to predict the accuracy of each algorithm in the ensemble from the extracted features.

Switching Planner At each time step, the switching planner selects which algorithm to run based on the predicted accuracy of each method and potentially other factors, such as computational cost.

We note that there is considerable flexibility in regards to the concrete implementation of each component. The concrete choices for the VO system proposed in this

paper are outlined in Figure 1. While this system worked well in our experiments, this framework easily accommodates variations for each component.

Below we describe selected components in more detail.

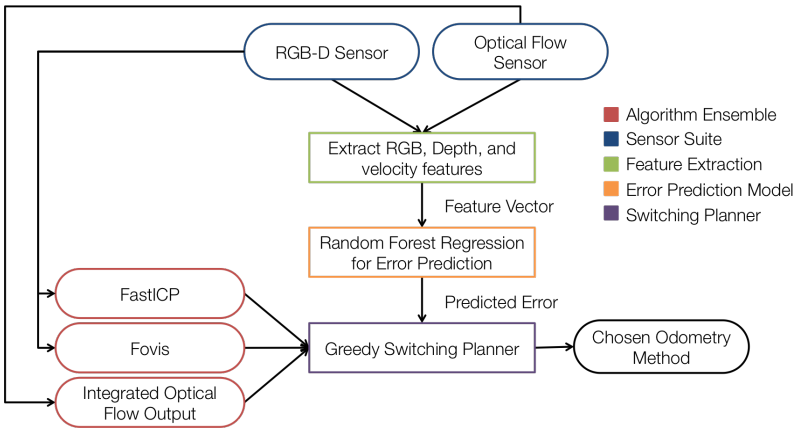


Fig. 1: **Framework Outline –** The adaptive architecture framework allows the robotic system to switch between different visual odometry methods. To choose whereto and when to switch between methods, we predict the error associated with each method given a feature vector extracted from current sensory information and state.

3.1 Sensor Suite

In the experiments for this work we use two different sensors. The first is a forward-facing RGB-D sensor, which combines a visible light camera with an active structured light system to create registered RGB and Depth (RGB-D) images.

The second is a specialized camera for optical flow [7], which faces downwards. The camera has an attached ultrasonic range sensor to estimate height relative to the ground.

Between these two sensors we have four channels of information: RGB, depth, ground optical flow and height relative to the ground. Each of these channels provide informative and complementary cues about the environment and vehicle state.

3.2 Algorithm Ensemble

A diverse yet powerful set of algorithms is crucial to maximizing the robustness and accuracy of our system as a whole. For this work we selected three representative VO methods, each using different subsets of the data:

Fovis [8] This VO algorithm uses the RGB and depth data from the RGB-D sensor. It works by detecting sparse keypoints from the RGB image and their 3D positions relative to the camera using the depth image. Then motion is estimated by robust matching of keypoints across frames using appearance information and geometric constraints.

FastICP [1] This method relies solely on depth data. It converts each depth image to a point cloud and estimates motion between frames by registering the point clouds to a local surface map using point-to-plane Iterative Closest Points (ICP).

Optical Flow [7] This method uses optical flow and height measurements to estimate motion. Unlike the other two methods, this method assumes the vehicle maintains constant height in each motion.

3.3 Feature Extraction

For each of our sensor modalities we extract multiple features designed to summarize various potentially relevant characteristics of the environment. We chose these features as they are compact, efficient to calculate and commonly used in the computer vision and point cloud processing literature.

Below we describe each feature according to the type of sensor data it describes; see Table 1 for a summary. The number corresponding to each feature will also appear in parenthesis with the description of that feature in the following text.

3.3.1 RGB Image Features

Note that as we do not expect color to be a discriminative feature in this context, we convert images to grayscale before further processing.

Luminance (9) We expect the luminance of the environment to be a useful predictor of algorithm accuracy, as methods that rely on RGB information will likely fail in dark environments. We use the Mean Intensity (9) of the grayscale image as a simple estimate of luminance.

Corners (2, 3) Keypoint-based algorithms such as Fovis rely on the presence of corner-like features in the environment; therefore, the quantity of these features may be a good predictor of the success of these algorithms. While we could use the results of Fovis' own corner detection step, this would entail running Fovis itself, and one of our goal's objectives is to reduce computation by only running algorithms we will use. Instead we simply run two corner detector algorithms from OpenCV,

Table 1: Image features in the feature vector will be referred to according to the numbers in this table. * indicates a GLCM statistic.

RGB Image	Depth and Point Cloud	State
1. Contrast*	10. Contrast*	18. Translational Velocity
2. Harris Corners	11. Correlation*	19. Angular Velocity
3. Shi Tomasi Corners	12. Valid Depth Ratio	
4. Correlation*	13. Energy*	
5. Edge Ratio	14. Homegeneity*	
6. Energy*	15. Linear-ness	
7. Entropy	16. Scattered-ness	
8. Homogeneity*	17. Surface-ness	
9. Mean Intensity		

the Harris (2) and Shi-Tomasi (3) detectors and include the number of corners from each as in the feature vector.

Edges (5) The presence of strong intensity edges is correlated with certain kinds of environments; for example, a cluttered indoor scene will probably have a larger number of edge pixels than an empty hallway. Hence we include the number of Sobel Edge (5) pixels in each image in the feature vector.

Texture (1, 4, 6-8) The presence of salient intensity textures in an environment may aid in the extraction and tracking of keypoints, and is also a strong cue to distinguish physical environments (for example, outdoors and indoor scenes have very different textures). To succinctly describe image texture we include the entropy (7) of each image, calculated using a histogram of the 256 possible intensity values, and features extracted from the gray-level co-occurrence matrix (GLCM) [6] of the image over four different angles (0°, 45°, 90°, and 135°). The GLCM counts how often every possible combination of gray-level pixel values occurs next to each other, and statistics of this matrix are popular texture features. The statistics we use are contrast (1), correlation (4), energy (6) and homogeneity (8).

3.3.2 Depth Features

Valid depth ratio (12) Our RGB-D sensor reports depth as a 16-bit image in which pixels are set to a special value if depth estimation is unsuccessful, deeming them invalid. If a large amount of the depth image is invalid, likely meaning it was out of range of the depth sensor, then any method using depth information may not be reliable. To quantify this we include the Valid Depth Ratio (12) of each depth image was computed to estimate the amount of information in that image.

Saliency (15-17) The three-dimensional shape of the environment may be a useful predictor of algorithm performance. For example, it might distinguish between environments that are underconstrained in depth information – particularly long, clear corridors – and environments that have several interesting depth features to track. To capture this we compute global saliency features [11], a coarse but efficient measure of shape. These features operate on point clouds, so we first project the

pixels with valid depth to a 3D point cloud $\{X_i\} = \{(x_i, y_i, z_i)^\top\}_{i=1}^N$. A 3×3 covariance matrix $\sum_{i=1}^N (X_i - \bar{X})(X_i - \bar{X})^\top / N$ is computed, λ_0, λ_1, λ_2 are extracted, such that $\lambda_0 \geq \lambda_1 \geq \lambda_2$. The three saliency features of the pointcloud are the scatteredness $f_{\text{scatter}} = \lambda_2$ (16), the linear-ness $f_{\text{linear}} = \lambda_0 - \lambda_1$ (15), and the surface-ness $f_{\text{surface}} = \lambda_1 - \lambda_2$ (17).

Depth Texture (10, 11, 13, 14) GLCM statistics, as described for the RGB texture features, were also extracted from the depth image.

3.3.3 State Features

The velocity, as reported by the visual odometry algorithms, was also included in the feature vector. Specifically, the magnitudes of both the translational (18) and angular (19) velocities of the currently active odometry method were used as features. This was added as a possible predictor for motion blur, which could affect the performance of visual odometry algorithms.

3.4 Error Prediction Model

The task of the error prediction model is to predict the trajectory errors of each algorithm at each time step given the feature vector described in the last section.

We formulate the problem as a regression problem, for which various methods may be used. Below we describe our chosen methodology.

Error Evaluation The output of our algorithm is an prediction of the trajectory errors each algorithm will make. We chose a metric based on the relative pose error (RPE) at a given time, described by [13] for VO evaluation. RPE measures the local accuracy of a trajectory, as compared to a ground truth trajectory, over a specified time interval. This measures how far the trajectory drifts in the given time interval.

$$\mathbf{E}_i := \left(\mathbf{Q}_i^{-1}\mathbf{Q}_{i+\Delta}\right)^{-1}\left(\mathbf{P}_i^{-1}\mathbf{P}_{i+\Delta}\right) \tag{1}$$

Equation (1) calculates RPE at time step i, over the time interval Δ. Here $\mathbf{Q}$ refers to the ground truth path and $\mathbf{P}$ refers to the estimated trajectory. In all experiments in this paper we use a time interval of $\Delta t = 2$ s.

As predicting structured matrices in nontrivial, instead we choose to predict two scalar quantities: the *translational error*, extracted from (1) as the Euclidean norm of the translational portion of $\mathbf{E}_i$, and the *angular error*, extracted as the absolute value of the angle of rotation from the rotation matrix portion of $\mathbf{E}_i$.

Error Prediction To predict the trajectory errors given the feature vector, we proceed as follows. First, we learn an independent regression model for each method and for each type of error (translational and angular). While joint prediction of the errors could potentially be more accurate, performing the predictions independently allows us to use virtually any off-the-shelf regression algorithm. Another advantage of using several different regressors instead of a joint regressor is that VO methods

can easily be added to the algorithm ensemble without affecting regression models that have already been learned.
Regression was used instead of the potentially simpler classification. One advantage of regression over classification is that we are able to determine the magnitude of an error before switching away from a method. Regression commits less strongly to which method might be the least accurate at any time. Regression gives us more nuanced information that allows more informed decisions. For example, if two methods are performing well, classification may indicate to frequently switch between the two. With regression, we may be able to determine that the cost of switching is not worth the slight decrease in error.
For the regression model we choose Random Forests (RF) [2]. The RF algorithm is an ensemble learning method that contains many decision trees, each contributing a vote towards an answer. We chose to use RF compared to other regression algorithms because random forests are efficient, robust, and empirically among the most accurate algorithms in many problems [3]. They are also able to predict feature importance.

3.5 Switching Planner

Decision Method We obtain both translational and angular error from evaluating the RPE as above. We considered two methods of combining angular and translational error to decide which visual odometry method to use. These two methods are shown in 2 and 3. Here ε_i^τ is the translational error for method i, and ε_i^α is the angular error for method i.

$$\text{method} = \arg\min_i(\varepsilon_i^\tau \varepsilon_i^\alpha). \tag{2}$$

$$\text{method} = \arg\min_i(\beta\varepsilon_i^\tau + \varepsilon_i^\alpha). \tag{3}$$

Ultimately we decided to use the additive metric, as in equation 3, with $\beta = 0.5$. This showed a larger decrease in both translational and angular error than metric 2, or the metric 3 for for either $\beta = 0.25$ or $\beta = 1$.

Greedy Switching Planner We aim to improve VO estimation without greatly increasing the computational cost, as it is important for our method to run online. Therefore, it is important that we do not run multiple odometry methods in parallel, as that can be very computationally expensive. As Fovis relies on keyframes for motion estimation, and our depth-only method relies on building a map from previous point clouds, it would introduce error into the system to instantaneously switch from one method to another.

We therefore allow three image frames of overlap between the two visual odometry methods, during which both methods would be making motion estimates. We do not use the newly started visual odometry method until the overlap is completed. A more complicated switching planner may be implemented, possibly by considering the benefit of switching before committing to a switch.

4 Experiments

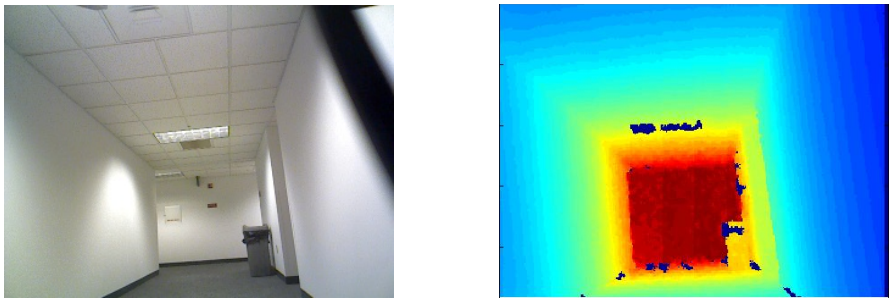
Datasets We test our framework in a variety of conditions that would be challenging for any individual algorithm. We looked specifically at four different environments. The *basement* datasets were taken in a dark, cluttered hallway (see Figure 2a). This environment is particularly challenging for any algorithm relying on light-dependent RGB images or the limited optical flow information available. The *hallways* datasets were taken in an area with brightly lit, blank hallways that may be challenging for algorithms relying on depth information (see Figure 2b). A depth cloud will be underconstrained, and therefore forward motion may be difficult to detect. The *spacious* environment included a large spacious room in which few depth features are available due to the limited range of the depth sensor (see Figure 2c). Lastly, the *cluttered* datasets were taken in an area with many objects detectable in both depth and RGB images (see Figure 2d).
We extract the feature vector and predict the VO error in real time, and switch between algorithms based on the predictions. For this paper, we trained RF regressions using one complete, > 60 *s* dataset from each of the four environments and tested on the remaining four datasets, again one from each environment.

Ground Truth Because we wanted data from a variety of environments, the use of motion capture systems was infeasible. Instead, localization was performed on the datasets, matching the depth point cloud with dense 3D maps of the environment. However, due to the challenging nature of these environments, localization failed in many cases. In order to get a close estimation of ground truth, points of the path that were accurately localized were manually selected. These points were then used as landmarks for the path. The most accurate odometry method for that dataset was then smoothed using iSAM [9].

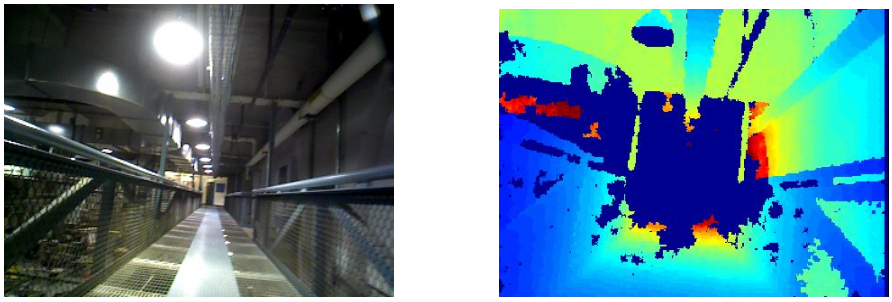
4.1 Feature Importance

The importance of each variable in the feature vector can be estimated during the training process of a random forest. We also collected information on the computation time for each feature. Computation time and variable importance were compared to determine if any features were not worthwhile to compute. In Figure 3, the maximum importance across all six forests for each variable is plotted against the variable computation time. We see that the mean intensity of the RGB image has a higher maximum importance than the other features and relatively low computation time. It is also clear that the corner detection methods are the most time-consuming and only of moderate importance. However, the computation time remained under 3 ms for these features, which is within the limits given by the image frequency of 15 Hz. Therefore, we kept all features while moving to the next step.

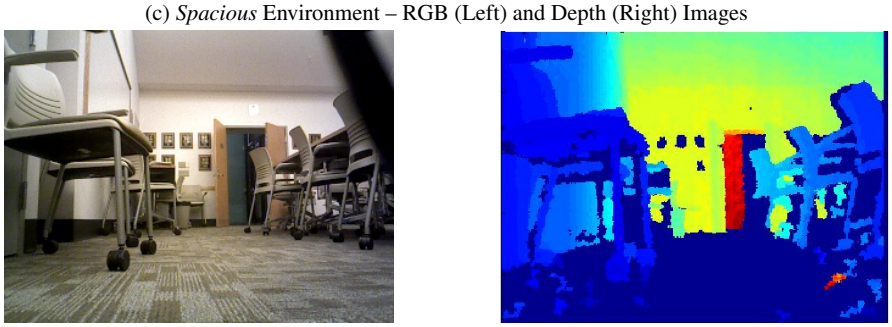
(a) *Basement* Environment – RGB (Left) and Depth (Right) Images



(b) *Hallways* Environment – RGB (Left) and Depth (Right) Images



(c) *Spacious* Environment – RGB (Left) and Depth (Right) Images



(d) *Cluttered* Environment – RGB (Left) and Depth (Right) Images

Fig. 2: **Environment Examples –** Four different, unique indoor environments were explored. Sample RGB and depth images are shown.

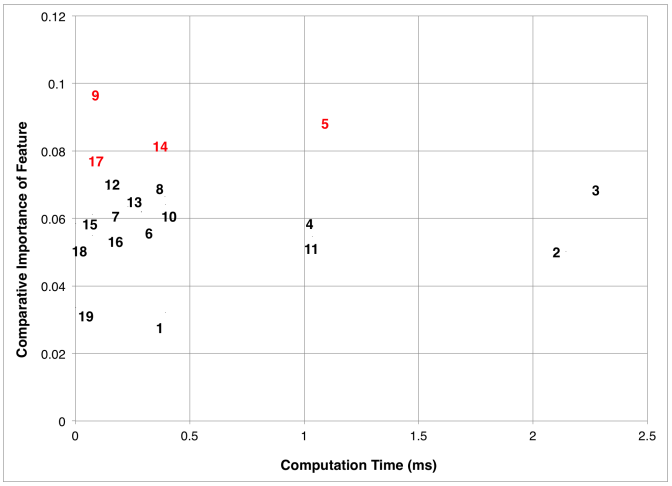


Fig. 3: **Feature Importance –** The importance of each feature is an estimation of how much it affects the accuracy of the random forest regression. The numbers refer to the features as numbered in Table 1. The features in red highlight some of the more important features.

4.2 Evaluation of Random Forest and Switching Performance

After training, we compared the trajectory error predicted by the random forest regression to the trajectory error previously extracted for training. We measure the effectiveness of our method by evaluating the RPE of the resultant path of switching, particularly in comparison with the lowest error odometry method.

Our results are detailed in Tables 2a and 2b. Here the RMS, maximum, and failure rate (FR) of the translational component of the RPE of each of the trajectories are compared. The same statistics on the angular component of the RPE is shown in Tables 3a and 3b. Here, the ideal path is generated by making the correct selection (according to Eqn. 2) at each point, using the true, extracted RPE. The switching path is generated using the architecture we have described thus far. In these tables (2 and 3), the RMS RPE is shown as a percentage of the RMS of the ideal path's RPE. The maximum of each path is the largest percentage increase in RPE over the ideal path's RPE over all time points. The FR of each path is the fraction of data points for which RPE exceeds a given threshold. For translational error, this threshold is 1 m; for angular error it is 0.5 rad.

Our method is able to improve robustness to large faults in VO. This is demonstrated by both the maximum and failure rate metrics. The switching method almost always has the lowest rate of failure, and often avoids the largest maximums in error that correspond to large failures in the VO estimation.

Our data also shows that overall our method is able to improve accuracy by improving the RMS relative pose error. However, our method does not always outper-

Table 2: **Translational Error –** We compare translational relative pose error (RPE) of our method against raw odometry output to determine if the random forest regression can predict RPE in a useful way. The RMS error is represented as a *fraction* of the RMS error of the *ideal* path. The Max. error is chosen as the largest % increase over the ideal path's error and is also represented as a fraction of the ideal. The failure rate (FR) is the fraction of data points for which the RPE exceeds a certain threshold. For translational error this threshold is 1 m.

(a) Training Data

Dataset	**Data**	**Switching**	**FastICP**	**Fovis**	**Opt. Flow**
Basement	RMS	**1.09**	1.40	1.14	1.97
(136 s)	Max.	**5.43**	27.38	**5.43**	18.95
	FR	**0.04**	0.16	0.05	0.44
Hallways	RMS	1.49	4.40	2.32	**0.93**
(235 s)	Max.	25.58	172.34	90.30	**8.77**
	FR	**0.00**	0.12	**0.00**	**0.00**
Spacious	RMS	**1.04**	1.84	**1.04**	1.51
(305 s)	Max.	**11.82**	34.67	**11.82**	18.75
	FR	**0.03**	0.30	**0.03**	0.12
Cluttered	RMS	**1.08**	1.63	**1.08**	2.75
(106 s)	Max.	**10.95**	11.45	**10.95**	168.62
	FR	**0.01**	0.05	**0.01**	0.05
Overall	RMS	**1.07**	1.86	1.14	1.75
	Max.	**25.58**	172.34	90.30	168.62
	FR	**0.02**	0.20	0.03	0.15

(b) Testing Data

Dataset	**Data**	**Switching**	**FastICP**	**Fovis**	**Opt. Flow**
Basement	RMS	1.15	1.84	**1.13**	2.49
(130 s)	Max.	24.06	24.06	**15.04**	15.46
	FR	**0.04**	0.19	**0.04**	0.59
Hallways	RMS	1.51	3.31	1.85	**1.10**
(244 s)	Max.	15.32	34.21	15.74	**3.88**
	FR	0.02	0.10	0.03	**0.00**
Spacious	RMS	**1.04**	1.80	**1.04**	1.41
(257 s)	Max.	9.01	39.19	9.01	**6.11**
	FR	**0.02**	0.56	**0.02**	0.27
Cluttered	RMS	1.14	**1.06**	1.14	1.10
(93 s)	Max.	12.08	**11.93**	12.08	67.45
	FR	0.41	**0.36**	0.41	**0.36**
Overall	RMS	**1.11**	1.76	1.13	1.55
	Max.	24.06	39.19	**15.74**	67.45
	FR	**0.07**	0.34	**0.07**	0.28

form the best individual odometry method. Notably, our method fails to outperform optical flow odometry in translational or angular RPE for either dataset, training or testing, of the *hallways* environment. One possible explanation is that this environment was not sufficiently distinguished from others in the feature space. Future work will include analyzing the difference between these environments in the fea-

Table 3: **Angular Error –** Angular error is represented here as translational error is in Table 2. The threshold used to calculate the failure rate (FR) was 0.5 rad.

(a) Training Data

Dataset	Data	Switching	FastICP	Fovis	Opt. Flow
Basement	RMS	1.53	1.51	1.77	**1.23**
(136 s)	Max.	**18.35**	22.23	**18.35**	21.46
	FR	0.06	0.04	0.07	**0.01**
Hallways	RMS	**1.08**	1.39	1.32	**1.08**
(235 s)	Max.	77.87	82.88	77.87	**10.36**
	FR	**0.00**	0.04	0.04	0.01
Spacious	RMS	**1.01**	1.54	**1.01**	1.62
(305 s)	Max.	28.96	**20.94**	28.96	32.41
	FR	**0.01**	0.06	**0.01**	0.07
Cluttered	RMS	**1.09**	1.21	**1.09**	3.89
(106 s)	Max.	5.89	**5.20**	5.89	60.40
	FR	**0.02**	0.03	**0.02**	0.22
Overall	RMS	**1.13**	1.46	1.25	1.68
	Max.	77.87	82.88	77.87	**60.40**
	FR	**0.02**	0.04	0.03	0.06

(b) Testing Data

Dataset	Data	Switching	FastICP	Fovis	Opt. Flow
Basement	RMS	1.22	**1.06**	1.30	1.13
(130 s)	Max.	11.87	9.50	11.87	**9.07**
	FR	0.05	**0.01**	0.08	0.03
Hallways	RMS	1.15	1.23	1.19	**1.08**
(244 s)	Max.	13.73	14.48	13.73	**3.26**
	FR	0.03	0.04	**0.02**	0.04
Spacious	RMS	1.06	1.26	**1.05**	1.33
(257 s)	Max.	**11.46**	157.67	**11.46**	79.82
	FR	0.04	0.10	**0.03**	0.13
Cluttered	RMS	**1.01**	1.07	**1.01**	1.16
(93 s)	Max.	**22.74**	29.48	**22.74**	70.82
	FR	**0.15**	0.18	**0.15**	0.18
Overall	RMS	**1.10**	1.19	1.12	1.22
	Max.	**22.74**	157.67	**22.74**	79.82
	FR	**0.05**	0.07	**0.05**	0.09

ture space and exploring potential new features that may aid distinguishing different environments.

5 Conclusions

In this paper we presented a method to robustify visual odometry by switching between algorithms based on the environment. By learning the error associated with sensory information through regression, this method aims to reduce visual odome-

try errors. The current results are promising in improving state estimation in various indoor environments, and particularly in avoiding large failures.

In future work, we would like to explore different methods in each component of the framework, and evaluate how they affect performance; for example, by adding more sensors and odometry algorithms, or jointly learning the features and the error prediction model.

References

1. Besl, P., McKay, N.D.: A method for registration of 3-D shapes. IEEE TPAMI **14**(2), 239–256 (1992)
2. Breiman, L.: Random Forests. Machine learning **45**(1), 5–32 (2001)
3. Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: ICML, pp. 161–168 (2006)
4. Censi, A.: An accurate closed-form estimate of ICP's covariance. In: ICRA, pp. 3167–3172 (2007)
5. Fang, Z., Scherer, S.: Experimental study of odometry estimation methods using RGB-D cameras. In: IROS, pp. 680–687 (2014)
6. Haralick, R.M., Shanmugam, K., Dinstein, I.H.: Textural features for image classification. IEEE Transactions on Systems, Man and Cybernetics (6), 610–621 (1973)
7. Honegger, D., Meier, L., Tanskanen, P., Pollefeys, M.: An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In: ICRA, pp. 1736–1741 (2013)
8. Huang, A.S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., Roy, N.: Visual odometry and mapping for autonomous flight using an RGB-D camera. In: International Symposium on Robotics Research (ISRR), pp. 1–16 (2011)
9. Kaess, M., Ranganathan, A., Dellaert, F.: iSAM: Incremental smoothing and mapping. IEEE Transactions on Robotics **24**(6), 1365–1378 (2008)
10. Kalman, R.E.: A new approach to linear filtering and prediction problems. ASME Journal of Basic Engineering (1960)
11. Lalonde, J.F., Vandapel, N., Huber, D.F., Hebert, M.: Natural terrain classification using three-dimensional ladar data for ground robot mobility. Journal of Field Robotics **23**(10), 839–861 (2006)
12. Leishman, R.C., Koch, D.P., McLain, T.W., Beard, R.W.: Robust visual motion estimation using RGB-D cameras. In: AIAA Infotech Aerospace Conference, pp. 1–13 (2013)
13. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: IROS, pp. 573–580. IEEE (2012)
14. Tomic, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., Grixa, I.L., Ruess, F., Suppa, M., Burschka, D.: Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. IEEE Robotics & Automation Magazine **19**(3), 46–56 (2012)
15. Vega-Brown, W., Bachrach, A., Bry, A., Kelly, J., Roy, N.: CELLO: A fast algorithm for covariance estimation. In: ICRA, pp. 3160–3167 (2013)
16. Zhang, Y., Chamseddine, A., Rabbath, C., Gordon, B., Su, C.Y., Rakheja, S., Fulford, C., Apkarian, J., Gosselin, P.: Development of advanced FDD and FTC techniques with application to an unmanned quadrotor helicopter testbed. Journal of the Franklin Institute **350**(9), 2396–2422 (2013)