Camera Pose Estimation using Regression Forest and RANSAC Optimization

Falk Ebert
4018276
Physics M.Sc.

Jason Pyanowski
3663907
Data and Computer Science M.Sc.

Nadine Theisen
3475402
Physics M.Sc.

Marven Hinze
3664283
Data and Computer Science M.Sc.

Abstract

This project aims to realize the approach of Shotton et al. [9] to infer the position and rotation of a camera in the 3D world space given a set of 2D images. A regression forest is used to predict the correspondences between pixels and world coordinates using RGB and depth information. An initial set of hypothesized camera poses are deduced by applying the regression forest and are then optimized with a preemptive RANSAC. The algorithm is evaluated on the 7-scenes dataset [5] which contains RGB-D images of 7 different 3D scenes. Subsequently, the ability of the forest to predict 2D-3D correspondences is evaluated. To verify the estimated camera poses of the RANSAC algorithm, the translational and angular error with respect to the known ground truths are measured. These findings are then compared to the results of [9].

1. Introduction

Inferring the pose of a camera in a known 3D environment is a crucial part of many up-to-date applications. Especially in the area of robotics, where Simultaneous Localization and Mapping (SLAM) algorithms are applied or in augmented reality a sufficiently precise estimate is required [9]. More recently, approaches using neural networks became present and provide a fast estimation of the position and orientation of a camera [1].

In this project, we provide an approach of interfering the pose of a camera in the world coordinate system based on the work of Shotton et al. [9]. Therefore, a regression forest is utilized to find correspondences between pixels in an image and world coordinates. Given a data set consisting of RGB images as well as depth information for each pixel, the 3D world coordinate can be estimated.

Based on the predicted world coordinates the camera location and orientation is obtained by applying RANSAC optimization. Initially, camera pose hypotheses are generated and subsequently refined using an energy function. The hypothesis with the lowest energy term is considered as the final camera pose.

The present approach does not require any deep understanding of the scene but completely relies on pixel-based features [9]. Additionally, only a small set of pixels is needed in order to estimate the camera pose, because the regression forest is trained on a sufficiently large data set. This allows to run the pose estimation on mobile devices or microcontrollers.

To evaluate the predictions of a forest, the 7-scenes dataset [5] set which consists of seven different scenes with each between $2,000$ and $12,000$ frames is utilized. To each frame the corresponding depth map and the ground truth camera pose is given which encodes position and orientation. The findings are compared to the results of [9] using the translational and angular error of the predicted to the ground truth camera poses for different scenes in the data set.

1.1. Related Work

Nowadays, many approaches to infer camera poses or poses of objects in general are based on either image-based techniques or sparse keypoint matching. The application of imaged-based pose estimation uses the whole image and matches different frames using high level features like Scale-Invariant Feature Transform (SIFT) or Speeded Up Robust Features (SURF) [8]. Thereby, an initial camera pose is refined using the obtained features and known camera poses. The result is then calculated as a weighted average over the images that describe the camera pose best.

More recently, approaches concerning sparse feature matching gained interest [6]. Again, high level features are detected but instead matched with a large set of feature descriptors. This approach requires a lot of engineering regarding the feature description and matching process. The

routine of inferring a camera pose presented in the reference paper of Shotton et al. [9] comes with the difference that only low-level features are utilized. Simple pixel-based features are applied in order to train a regression forest without the need of feature descriptors. The predictions of the camera pose are then made directly on the output of the forest which correlates 2D image pixels with 3D world coordinates. The underlying forest can be evaluated for arbitrary pixel positions and is therefore sparse.

2. Methods

This section gives an overview of the methods used in this project. We will discuss the concept of regression forests including feature extraction and the RANSAC algorithm used for estimating the camera pose from the image data. We will not discuss all aspects in full detail but only provide further explanations where we find it relevant for the presentation of our work and results.

2.1. Data Preparation

To train the regression forest the 7-scenes dataset [5] is utilized. Each scene consists of multiple image sequences that cover the area of interest. Using RGB-D Kinect camera, depth information for each pixel is extracted. The resolution of the 24 bit RGB-images is 640×480 pixels and the corresponding 16 bit depth map is given in millimeters. Invalid depths are assigned the value 65535. For each image a 4×4 ground truth camera pose in homogenous coordinates is provided, which allows to validate the estimated camera pose matrix for a given image. Each dataset is split randomly into train and test.

Based on that, our data set is composed of a number of samples $\{(\boldsymbol{p}_i, \boldsymbol{w}_i)\}$ with $\boldsymbol{p}_i = (x_l, y_l)_i$ and $\boldsymbol{w}_i = (x_s, y_s, z_s)_i$ where the subscripts l and s denote image pixel coordinates and scene coordinates respectively.

2.2. Regression Forest

In this project we use a regression forest to predict scene coordinates for a given sample image coordinate as suggested in [9]. In this section we will give some background on the concept of regression forests, the pixel coordinate labeling and the image-feature extraction on which the forests base their predictions.

2.2.1 Decision Trees

A regression forest consists of a number N of regression trees, which each consist of a root node and it's children [4]. We consider the special case of binary decision trees where each node (unless it is a leaf node) has a left and a right child. Each node n stores a set of parameters θ_n which are used in the calculation of a feature response function $f_\theta(\boldsymbol{p}) \in \{1, 0\}$ which determines if the input data point $\boldsymbol{p}$ is branched to the left or right child node. To every leaf node (i.e. a node without children) we assign a response value $\boldsymbol{w}_n$ representing the tree's prediction for a data point $\boldsymbol{p}$ that has reached this node. In our specific application the input data points $\boldsymbol{p}$ are 2D pixel coordinates and the responses $\boldsymbol{w}$ are 3D world coordinates.

The process of training the tree involves finding the set of parameters $\{\theta_n \,|\, \forall \text{ nodes } n\}$ which results in the best predictions for previously unseen input data points $\boldsymbol{p}$. However, this final goal cannot be optimized directly during training, as this would entail optimization in the very large space of all possible parameters for all tree-nodes simultaneously. Therefor it is much more efficient to optimize the parameters one node at a time using a proxy-objective $Q(S_{\text{tot}},\, S_{\text{left}}(\theta),\, S_{\text{right}}(\theta))$ which ideally leads to a similar result.

Let $S_{\text{tot},n} = \{(\boldsymbol{p}_i, \boldsymbol{w}_i) \,|\, i \in |S_n|\}$ denote the input data and target response for a given node n. According to its parameters the node splits this set into the subsets

$$\begin{aligned} S_{\text{left},n}(\theta_n) &= \{(\boldsymbol{p}_i, \boldsymbol{w}_i) \,|\, f_{\theta_n}(\boldsymbol{p}_i) = 0\} \\ S_{\text{right},n}(\theta_n) &= \{(\boldsymbol{p}_i, \boldsymbol{w}_i) \,|\, f_{\theta_n}(\boldsymbol{p}_i) = 1\} \end{aligned} \tag{1}$$

corresponding to left and right child node respectively. The objective function Q is then used to assign a score to the split resulting from the parameters θ_n at this node.

The objective function used here optimizes for a reduction in variance of the target responses (i.e. wants the tree to group together points with similar scene coordinates). Defining $W_n = \{\boldsymbol{w}_i \,|\, (\boldsymbol{p}_i, \boldsymbol{w}_i) \in S_n\}$ this objective can be mathematically expressed as:

$$\begin{aligned} Q(W_{\text{tot}}, \theta) &= \text{Var}(W_{\text{tot}}) - \sum_{d \in \{\text{left, right}\}} \frac{|W_d(\theta)|}{|W_{\text{tot}}|} \text{Var}(W_d(\theta)) \\ \text{with} \quad \text{Var}(W) &= |W|^{-1} \sum_{\boldsymbol{w} \in W} ||\boldsymbol{w} - \bar{\boldsymbol{w}}||_2^2 \end{aligned} \tag{2}$$

Training the complete tree then involves choosing a set of images from which a number of sample pixels is drawn. These samples are then evaluated (i.e. split into left and right set) by the tree's root node for a number of parameter-sets. For each set of parameters, the split's score is calculated using the objective function and the optimal parameters for this node are recorded. The process is then repeated for the node's children with the corresponding set (left or right; split according to the optimal parameters) until either the set contains only one element or the maximum depth is reached. When the maximum depth is reached and the set still contains more than one element, a response needs to be calculated for the node. This response represents the tree's prediction for all pixels, which reach this node and is calculated from the sample's ground truth scene coordinates.

In practice, a mean shift variance algorithm [3] is used to find the center of the largest cluster. Thereby, at most 500 random coordinates are utilized for calculation to limit the computational cost.

2.2.2 Image Features

In order to use a decision tree to infer scene coordinates from the pixel coordinates, it is necessary to calculate features associated with a given pixel coordinate from the image data. This is implemented in the feature response function mentioned earlier. We have chosen to use the *Depth-Adaptive RGB* image features from [9] which are defined as

$$f_\theta^{da-rgb}(\boldsymbol{p}) = I\left(\boldsymbol{p} + \frac{\boldsymbol{\delta}_1}{D(\boldsymbol{p})}, c_1\right) - I\left(\boldsymbol{p} + \frac{\boldsymbol{\delta}_2}{D(\boldsymbol{p})}, c_2\right) \tag{3}$$

where $I(\boldsymbol{p}, c)$ is the images value in the $c \in \{\text{r, g, b}\}$ component at the pixel location $\boldsymbol{p}$ and the *Depth* image feature specified as

$$f_\theta^{depth}(\boldsymbol{p}) = D\left(\boldsymbol{p} + \frac{\boldsymbol{\delta}_1}{D(\boldsymbol{p})}\right) - D\left(\boldsymbol{p} + \frac{\boldsymbol{\delta}_2}{D(\boldsymbol{p})}\right) \tag{4}$$

where $D(\boldsymbol{p})$ refers to the depth value corresponding to pixel $\boldsymbol{p}$. The parameters are given explicitly as $\theta = \{\boldsymbol{\delta}_{1,2}, c_{1,2}, \tau\}$. The offsets $\boldsymbol{\delta}_{1,2}$ allow the tree node to incorporate non-local information for each pixel. When the resulting lookup location is outside the image boundary, the original sample pixel $\boldsymbol{p}$ is disregarded in training and can also not be predicted by the forest. The parameter threshold τ is used to decide if the feature is split to the right or left child node according to $f_\theta(\boldsymbol{p}) \leq \tau$.

2.3. RANSAC Optimization

The predicted world scene coordinates of the forest are used to derive the corresponding camera pose matrix for a given image. The objective of the optimization is to find a camera pose matrix that minimizes an energy

$$E(H) = \sum_{i \in I} \rho(\min_{\boldsymbol{m} \in \boldsymbol{M_i}} \|\boldsymbol{m} - H\boldsymbol{x_i}\|_2) = \sum_{i \in I} e_i(H) \tag{5}$$

whereas $i \in I$ is a pixel index, ρ a top-hat error function with a width of 0.1m, M_i the set of predictions for pixel $\boldsymbol{p_i}$, H a camera pose hypothesis and $\boldsymbol{x_i}$ is the camera scene coordinate of $\boldsymbol{p_i}$ [9]. If ρ returns 0 for a given pixel, it is considered to be an inlier, otherwise it is an outlier.

The hypotheses optimization is based on an adapted version of preemptive RANSAC [9]. Algorithm 1 shows the pseudocode of the RANSAC optimization. The procedure starts with the initialization of K hypotheses for the camera pose matrix. The next step is to create a pixel batch B on which each hypothesis is evaluated. The resulting inliers per hypothesis are stored, and their energy is updated. Furthermore, to reduce the runtime, the mode $\boldsymbol{m}$ from Equation 5 is stored for the pixel inlier. The next step is to sort the hypotheses by their energy in ascending order and discard the top half. Next, the remaining hypotheses are refined.

Algorithm 1 RANSAC optimization

```
K ← K_init
initialize hypotheses {H_K}_{k=1}^K
initialize energies E_k
while K > 1 do
    sample random pixel batch B
    for all i ∈ B do
        M_i = forest(i)
        for all k ∈ {1, ..., K} do
            E_k ← E_k + e_i(H_k)
            if e_i(H_k) is 0 then
                add i to inliers_k
                add m to modes_k
            end if
        end for
    end for
    sort hypotheses by E_k
    K ← ⌊K/2⌋
    refine hypotheses H_k on inliers_k for k = 1, ..., K
end while
return H_k
```

2.3.1 Hypotheses Initialization

To initialize a hypothesis, three random pixels $\boldsymbol{p_i}$ with $i \in \{0, 1, 2\}$ are sampled. The pixel $\boldsymbol{p_i}$ must have a depth unequal to $2^{16} - 1$ and 0. Every $\boldsymbol{p_i}$ is evaluated on the forest, this results in three sets $\boldsymbol{M_i}$ containing the prediction $\boldsymbol{m}$ from every tree. For each p_i a random prediction is sampled from $\boldsymbol{M_i}$. Furthermore, for every $\boldsymbol{p_i}$ the camera scene coordinate $\boldsymbol{P}$ is calculated.

$$\boldsymbol{P} = (\boldsymbol{C}^{-1} \cdot \tilde{\boldsymbol{p}}_{\boldsymbol{i}}) \cdot d \tag{6}$$

where $\boldsymbol{C}$ is the intrinsic camera matrix, $\tilde{\boldsymbol{p}}_{\boldsymbol{i}}$ the pixel $\boldsymbol{p_i}$ in homogeneous coordinates and d is the depth value for $\boldsymbol{p_i}$ in meters.

For finding the best rotation between the camera scene and world coordinates, the Kabsch algorithm is used. The Kabsch algorithm allows to find the best rotation that transforms one set of vectors into a second set of vectors [7]. Let M be a matrix containing the camera scene coordinates row wise and W a matrix containing the world coordinates, whereas $\bar{\boldsymbol{m}}$ and $\bar{\boldsymbol{w}}$ are the mean coordinates. To find the best rotation one needs to compute the cross covariance matrix C of M and W. The next step is the application of a singular value decomposition on C given as [2]

$$C = U\Sigma V^T. \tag{7}$$

The optimal rotation R and the translation $\boldsymbol{t}$ are then obtained by [2]:

$$\begin{aligned} R &= V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(VU^T) \end{bmatrix} U^T \\ \boldsymbol{t} &= \bar{\boldsymbol{w}} - R\bar{\boldsymbol{m}} \end{aligned} \tag{8}$$

By combining R and $\boldsymbol{t}$ into an affine transformation the initial hypothesis H is obtained.

2.3.2 Pose refinement

For refinement of a hypothesis the respective pixel inliers are retrieved and their camera scene coordinates are calculated. These camera scene coordinates and the stored predictions for the pixel inliers are passed to the Kabsch algorithm, the returned affine transformation is then considered as the refined pose.

3. Experiment Evaluation

To verify the outcome of the regression forest and consecutive RANSAC optimization, the utilized metrics are described in the following section. As basic error measurement of the forests predicted world coordinates to the ground truth the ℓ^2-norm is used. Thereby, the error per scene is calculated by taking the average deviation in meters over all trees within a forest.

The camera poses predicted by the RANSAC algorithm are compared using the translational and angular error which are further described in the subsection below. All findings are then related to the results obtained by Shotton et al. [9]. They classify the camera poses as *correct* if the deviation is less than 5cm and 5°.

3.1. Metrics

To evaluate the estimated camera poses, the translational and angular error with respect to the ground truth camera poses is measured. Let the estimated pose matrix H_{est} and the ground truth H_{gt} be 4×4 matrices in homogeneous coordinates that encode the location and orientation of the camera in the world coordinate system. To measure the translational offset between them, the translation vectors $\boldsymbol{t_{est}}, \boldsymbol{t_{gt}} \in \mathbb{R}^3$ of the matrices are used. By applying the ℓ^2-norm to the translation vectors, the error is obtained by

$$\varepsilon_t = \sqrt{\sum_{x,y,z} |\boldsymbol{t_{est}} - \boldsymbol{t_{gt}}|^2}. \tag{9}$$

To compare two rotational poses, the angle of the difference rotation matrix R is utilized. Denote the whole 3×3 upper-left matrix of H_{est} and H_{gt} as R_{est} and R_{gt} respectively. They refer to the orientation in space and the difference rotation can be computed as $R = R_{est}^T R_{gt}$. The angle between those two rotation matrices is the desired error measure and defined as

$$\varepsilon_r = arccos\left(\frac{tr(R) - 1}{2}\right) \tag{10}$$

where $tr(R)$ is the trace of a matrix.

4. Results

In this section we present our findings and some details on the implementation as well as the parameters that have been chosen to train the forest and execute the RANSAC optimization. The findings are evaluated according to the measurements described in Section 3. Firstly, it is investigated how accurate the forest predicts the 3D world coordinates from 2D image coordinates. Subsequently, the trained forest is used to obtain the camera poses for a number of unseen test images using RANSAC. Five out of seven scenes are evaluated, and their results are compared among them and to the reference paper of Shotton et al. [9].

Hyperparameter	Value
Number of Trees	5
Max Tree Depth	16
Number of Train Images	500
Pixel Samples per Image	5000
Parameter Samples per Image	1024
Feature Type	DA-RGB

Table 1: Hyperparameters used for forests training.

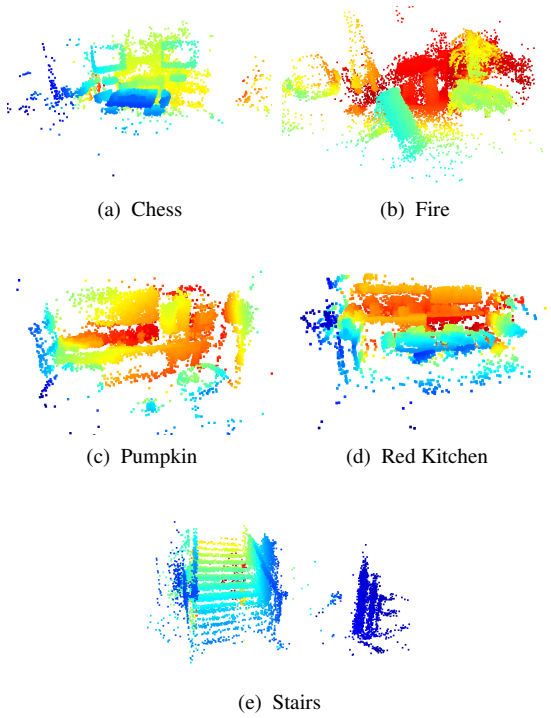
4.1. Implementation and Parameter Settings

For the scenes *Chess*, *Fire*, *Pumpkin*, *Red Kitchen* and *Stairs* a forest is trained. These scenes are chosen because they show the most significant results for the feature type *Depth Adaptive RGB* which is considered for the following evaluations.

The forests are trained with the hyperparameters given in Table 1 and are chosen according to the values stated in [9]. The number of parameter samples per image wasn't provided, and we found the value 1024 to work out well, because further increase wasn't improving the number of correct world coordinate predictions. For each scene 500 random images from the test data set are used to predict the 3D world coordinates on the trained forest. Every image is evaluated with a number of $5,000$ test pixels.

Hyperparameter	Value
Size of batch B	500
K_{init}	1024

Table 2: Hyperparameters used RANSAC optimization.



(a) Chess (b) Fire

(c) Pumpkin (d) Red Kitchen

(e) Stairs

Figure 1: For each scene a forest is trained with the hyperparameters stated in Table 1. The forest is then evaluated for 500 test images with each having $5,000$ sample pixels. For each pixel the corresponding 3D world coordinate is obtained. The resulting point clouds are displayed in subfigures (a)-(e).

The consecutive RANSAC optimization uses the parameters given in 2. Initially, a total of 1024 hypotheses are sampled. The evaluation of the hypotheses is then applied on 500 random pixels. From each scene 500 test images are sampled randomly and the corresponding camera poses are obtained.

Runtime: The full code is implemented in python using multiprocessing libraries and vectorization. On a standard desktop computer with AMD Ryzen 5-1600 CPU with 6 cores the training process of a single image takes approximately 5.4 seconds per tree. This is slower than the implementation of Shotton et al. [9]. We assume that this is due to the fact that python runs via an interpreter, while C++ is precompiled and therefore faster.

Scene	Average Error [m]
Chess	1.079 ± 0.388
Fire	0.893 ± 0.313
Pumpkin	1.413 ± 0.615
Red Kitchen	1.705 ± 0.854
Stairs	1.496 ± 0.753

Table 3: Average Error and standard deviation in meters between the predicted 3D world coordinates of the forest trained with *Depth Adaptive RGB* features and the respective ground truths using the ℓ^2-norm. A total of 500 images each with $5,000$ sample pixels is used for evaluation for each scene.

4.2. Image to World Point correspondences

The forests exhibit the basic ability to predict 3D world coordinates based on 2D image pixels and qualitative results are shown in Figure 1. The point clouds allow to recognize the scenes even by a sparse number of utilized sample pixels.

However, most scenes show areas where no 3D coordinates were predicted. This could be attributed to a false prediction of 3D points. Due to the fact that the forest is trained on known world coordinates, the assignment of a pixel to a world point that is actually present in the scene is possible.

Another explanation are invalid values during feature generation. Correspondences that could not be predicted are caused by the fact that sample pixels are potentially shifted out of the image frame or correspond to an invalid depth value. Predicting the 3D world coordinates for all scenes, the forests exhibit 91.2% valid predictions on average. This implies that a forest is capable of finding valid correspondences.

To further investigate these findings, the predicted coordinates of the forest are compared to the known ground truths using the ℓ^2-norm. In Table 3 the error in meters and corresponding standard deviation indicates rather large deviations. It varies among 1.705m for *Red Kitchen* to 0.893m for the scene *Fire*. This suggests that the mapping is not as precise as desired. By looking at the histogram of accumulative errors for each scene in Figure 2, the previous observations are confirmed. It can therefore be expected that the majority of poses that will be predicted by the RANSAC algorithm deviate more than 5cm and $5°$.

Comparing the average deviations for each scene to the findings of Shotton et al. [9, Table 1], similarities appear. The camera poses estimated by their implementation show the best results for the scenes *Chess* and *Fire*. This is also indicated by the findings of the forests in this section. Since no precise measurements were made in [9] a direct comparison of the predicted world coordinates is not possible.

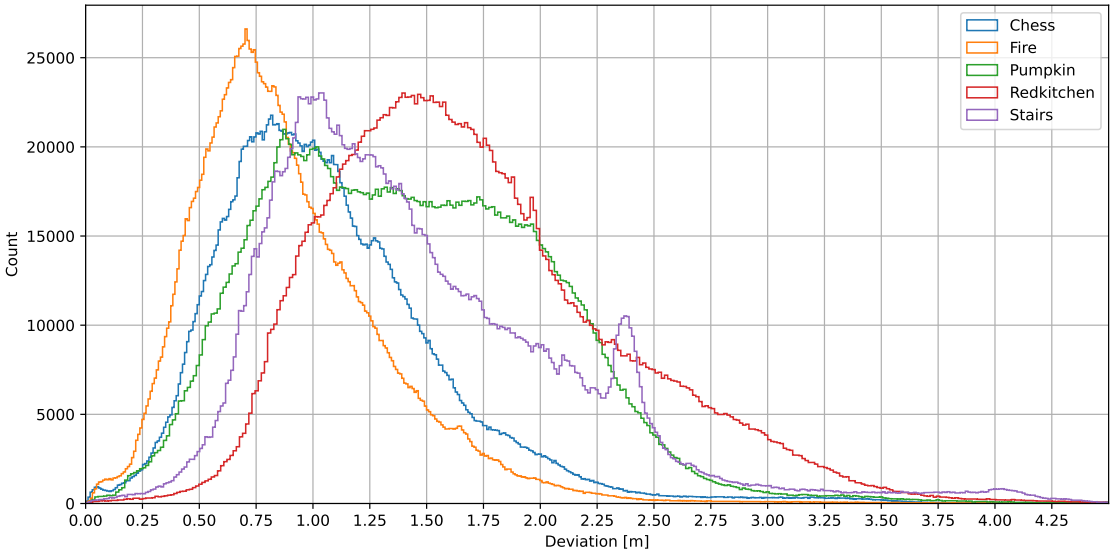


Figure 2: The histogram shows the average error of the predicted 3D world coordinates of each tree in a forest.

It is to be expected, that the obtained camera poses in the following section follow the same trend.

4.3. Camera Pose Predictions

The predicted camera poses show rather large deviations from the ground truths as shown in Table 4. All scenes exhibit deviations with more than 0.68m and $40.06°$ which is fairly inaccurate. However, this is to be expected regarding the results of the forests stated in Subsection 4.2. Having large deviations of the image to world correspondences already, the prediction of related camera poses based on these values can not lead to higher accuracies.

Concerning the scenes *Chess* and *Fire*, the translational and angular error of the camera poses are the smallest. This is coherent with the precision of the predicted point correspondences of the previous subsection. That indicates that more precise regression forests lead to smaller error of the estimated camera pose by the RANSAC optimization.

The scenes that exhibit the highest amount of correctly classified poses are *Chess* and *Fire*. The worst is provided by the scene *Red Kitchen* where a lot of specularities and reflections are present. In the scene *Stairs* many repeating patterns and ambiguities occur. This could explain the bad results in comparison to the other scenes.

With increasing tolerance of a pose to be classified as *correct* the percentage of correct poses increases. Given a maximum of 5cm and $5°$ the results can not compete with [9]. Nevertheless, the trends found in the data sets could be reproduced. The only exception is that the *Fire* scene performs better in our implementation which can be addressed to the performance of the respective forest shown in Subsection 4.2.

5. Conclusion

To estimate the pose of a camera, a regression forest with consecutive RANSAC optimization is utilized. Pixels are sampled from random images of a scene and corresponding 3D world coordinates are predicted by the regression forest. Subsequently, the RANSAC algorithm initializes a number of camera pose hypotheses and refines them. To verify the results, the 7-scenes data set is used which consists of seven different 3D scenes. The predicted camera poses are then evaluated using the translational and angular error. These findings are compared to the results of Shotton et al. [9].

The best camera pose predictions were obtained for *Fire* and *Chess* and the worst for *Red Kitchen* and *Stairs*. Our results show the same trend as in [9]. However, our approach is not able to achieve a similar amount of correctly classified poses. This can be addressed to the larger deviations of predicted 2D-3D correspondences by the trained forests.

There is certainly potential to optimize our implementation in order to increase the overall accuracy of predicted camera poses. For the future an optimization of the utilized hyperparameters could be beneficial. Additionally, other feature types may be of interest.

		Chess	Fire	Pumpkin	Red Kitchen	Stairs
Error	Translational [m]	0.92 ± 1.09	0.68 ± 0.78	1.41 ± 1.49	1.89 ± 1.31	1.48 ± 1.25
	Angular [°]	58.52 ± 61.85	40.06 ± 44.83	66.44 ± 65.79	91.01 ± 59.82	74.59 ± 65.92
Correct Poses [%]	$0.05m, 5°$	14.25	**18.25**	10.25	4.6	7.8
	$0.1m, 10°$	25.75	**29.75**	20.25	9.2	12.2
	$0.15m, 15°$	33.8	**38.8**	25.75	10.8	14.2

Table 4: For each scene the camera poses are predicted and compared to the ground truth. The average translational and angular error are shown as well as the related standard deviations. The amount of correctly classified poses is listed in the lower part of the table for different thresholds.

Abbreviations

SIFT Scale-Invariant Feature Transform

SLAM Simultaneous Localization and Mapping

SURF Speeded Up Robust Features

References

[1] Hunter Blanton, Scott Workman, and Nathan Jacobs. A structure-aware method for direct pose estimation. *CoRR*, abs/2012.12360, 2020. 1

[2] Eric Brachmann and Carsten Rother. Visual camera relocalization from rgb and rgb-d images using dsac, 2020. 3, 4

[3] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. 3

[4] A. Criminisi and J. Shotton. *Introduction: The Abstract Forest Model*, pages 7–23. Springer London, London, 2013. 2

[5] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time rgb-d camera relocalization. In *International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, October 2013. 1, 2

[6] Stefan Holzer, Jamie Shotton, and Pushmeet Kohli. Learning to efficiently detect repeatable interest points in depth data. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 200–213, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. 1

[7] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, Sep 1976. 3

[8] Georg Klein and David Murray. Improving the agility of keyframe-based slam. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, pages 802–815, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 1

[9] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2013. 1, 2, 3, 4, 5, 6