

INFORMATION AND COMMUNICATION TECHNOLOGIES
COLLABORATIVE PROJECT

“V-Charge”

Autonomous Valet Parking and Charging for e-Mobility

Grant Agreement Number 269916

Deliverable Number	2.2
Deliverable Name	Report on novel concepts and algorithms for localization with visual feature and global semantic map
WP Number	2
Date of Preparation	28.11.2012
Date of Submission	29.11.2012
Editor/Beneficiary	R. Triebel/Univ. Oxford

Contents

1	Introduction	2
2	Semantic Mapping	2
2.1	Laser-based Semantic Mapping	3
2.2	Vision-based Semantic Mapping	3
3	Features Used	4
3.1	Features for Laser Data	4
3.2	Features for Vision Data	5
4	Classification for Semantic Mapping	5
4.1	Support Vector Machines	6
4.2	Gaussian Process Classification	6
5	Experiments	7
5.1	Experiments with Laser Data	8
5.2	Experiments with Vision Data	8
6	Conclusions	11

1 Introduction

This report covers two aspects, namely visual/geometric and semantic aspects of the maps. For the former, more low-level, a major problem in localization (in particular for loop closing and recognition of earlier-visited places) is that perspective effects strongly distort the appearance of surfaces useful for visual navigation. Several authors have tried to (approximately) factor out the nuisance of perspective by designing invariant feature descriptors or by detecting local affine frames in order to normalize image structure at least with respect to the linearization of the distortion (cf. to [1] for an overview). In case that one does also have access to the underlying geometry (e.g. in a stereo system, or when a 3D model is constructed from a video), this geometry can be very useful for obtaining a normalized (e.g. frontal) representation of the features that does not suffer from perspective effects. Consequently, the goal in task 2.2 is to improve and extend existing stable visual features for wide-baseline matching by exploiting the underlying local 3D shape information. Towards this end the consortium has generalized a state-of-the-art technique called viewpoint invariant patches [2] that takes advantage of large dominant planes in the scene in order to remove perspective effects from local image features. However, rather than achieving viewpoint invariance only when planes are seen (e.g. a facade), we can now also normalize features that lie on any developable surface, which is much more general than before. The intuition is that we unroll the “wall paper” of the scene and rather than trying to find correspondences in perspective images we find correspondences in the wall papers constructed from each view. This work has been published[3] at the depth camera workshop in conjunction with the European Conference on Computer Vision and a detailed derivation of the concepts can be found in that publication (attached to this report).

The other aspect of D2.2 is the automated semantic annotation of the environment map, which is also an important prerequisite for the further navigation and planning process of the vehicle. In particular, the semantic information will be used to improve the localisation of the vehicle within its environment by disambiguating situations where semantics are needed for localization. Examples of this include cases where the vehicle is located close to landmarks such as a parking spot or a curb, which are very hard to identify and to match to a map when no semantic information is given. Therefore, we investigate in this work ways of obtaining a semantic annotation automatically. We note however, that, despite the somewhat misleading title of this report, the main focus is not the localisation process itself but rather the mechanism of obtaining useful information required for localisation. The reason for that is the fact that the localisation module is mostly handled in WP3 with the corresponding reports given there. Of course, the integration of the modules is an important aspect addressed in this project, and this explains the comparably comprehensive denomination of this work. The material presented in the following sections is mostly taken from two publications, [4] and [5], where the latter one is still under review and can therefore not be provided in full length at this point in time. However, the results in there that are most relevant for this project are presented here.

2 Semantic Mapping

Adding semantic information to a given environment map can be performed in various different ways, and they strongly depend on the specific requirements and also on the particular kind of semantic information requested. For example, in Deliverable D2.1 we already presented techniques to automatically annotate overhead images from car parks based on template matching. There, we showed that the method works well in general, but that it also has shortcomings in the presence of occlusions or shadows. Also, for more complex objects such as traffic lights and road signs, template matching can not be applied due to the high variability of the appearance of the objects. Therefore, we are developing a semantic annotation module based on classification with a supervised learning method, and we are reporting results of this algorithm here.

It is important to note that, apart from the goal of providing a reliable classification in terms of precision and recall of the algorithm, we are also and particularly interested in a method that provides useful information about the *uncertainty* of an assigned class label. There are at least two important reasons for this: First, a given uncertainty estimate about a class label, which resembles the “intuitive” error margin of the classifier well enough, can be used to improve



Figure 1: Semantic labeling using laser data. In this example, a car par was scanned by a rotating laser scanner and a classifier was trained on 6 different classes. The left figure shows the ground truth labeling, and the right one shows the classification result.

the classification result by simply reporting only class labels with low uncertainties. An algorithm, where these highly certain labels often correspond to wrong classifications is less useful here, and we show an example later in the experiments. The second reason to aim for good uncertainty estimates is that they can be directly used by other modules such as the localisation or the navigation. These will then have a notion of reliability of the provided information, which can be used to improve, e.g. the position estimate of the vehicle when looking for matches of observed landmarks to the given map representation. The capability of providing useful uncertainty information along with the class labels will be denoted as *introspection*.

2.1 Laser-based Semantic Mapping

Although the sensor setup used on the vehicle in this project does not include laser range scanners, we nevertheless investigate this sensor modality here as a potential data source for our semantic labeling algorithm. The reason for this is two-fold: First, we want to show the usefulness of the approach for other applications beyond the one investigated in this project, and second, the data provided by the dense map, which is developed in task 2.1 is very similar to the point cloud data obtained from a rotating laser scanner. Therefore, the algorithm will not change much when used on the dense map data later on the fully integrated system.

Fig. 1 shows an example of a 3D mesh obtained from a laser scanning device, that was automatically annotated with our method. In particular, we use a Gaussian Process Classifier (GPC) as described in Section 4.2, which was trained on a hand-labeled set of point clouds. As we can see, the method yields a very good classification result. But we will also show below that it provides very reasonable uncertainty estimates, especially on instances of unknown classes, i.e. which are not represented in the training data.

2.2 Vision-based Semantic Mapping

In addition to the laser data shown in the last section, we also investigated introspective classification on camera data, as this is the sensor modality used on the vehicle used in V-Charge. However, instead of first computing a dense 3D map from the camera images, we apply the algorithm directly on objects to be classified in the raw images, where we focus on traffic relevant objects such as traffic lights and road signs. Thus, the classification here is based on

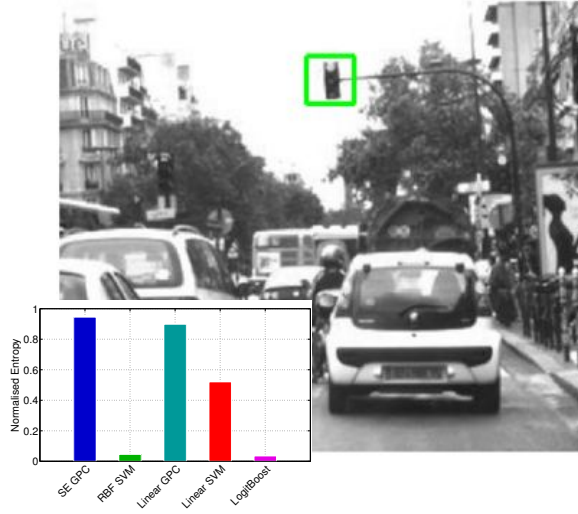


Figure 2: Uncertainty estimates obtained from five different classifiers trained on traffic lights. Note that *all classifiers falsely reject* this window as *not containing* a traffic light (class decisions are not shown). However, the GPC variants do so with a significant amount of uncertainty while the others are inappropriately overconfident.

appearance rather than on the geometric properties of the environment. The idea here is to combine appearance-based and geometry-based classification in a later stage of the project to obtain more accurate results, but for now both modalities are analysed separately.

Fig. 2 shows an example scene, for which the use of an introspective classifier is a safety-relevant requirement. We used five different classifiers to detect the traffic light inside the green box¹, but no classifier could detect the traffic light. However, the classifiers using the GP framework were able to give high uncertainty values for their classification, which is a very useful source of information. For example, it can be used to either not rely at all on the classification result and instead treat this as a “dangerous” situation due to lack of knowledge, or to take other measures such as a request for human intervention in these cases. All other classifiers are too confident in this situation, which can lead to safety-critical decisions, e.g. by not reducing the speed.

3 Features Used

Before describing the classification algorithms investigated in this work, we explain the features extracted from the raw data. These features are different between the input sensor modalities, and we give more details in the following.

3.1 Features for Laser Data

The laser scanner used in our experiments provides 3D point clouds acquired with a rotating device. To obtain feature vectors that are useful for classification, the point clouds are converted into triangular meshes by connecting neighboring data points if they are closer than a given threshold. We then compute normal vectors for all triangles and apply a

¹In the course of this report, only four of them are analysed in more detail, namely the two SVM-based ones and the two GPC variants.



Figure 3: Mesh segmentation. Every segment has a different color assigned. In this example, the mesh has self-overlapping parts, which is why e.g. the building is split into several segments. Note that “rough” surfaces such as those on the trees are segmented as well as smoother regions such as the ground.

segmentation algorithm based on the work of Felzenszwalb and Huttenlocher [6], where the similarity of two adjacent triangles is defined by the angles of their normal vectors. Each resulting mesh segment consists of a single connected component and is consistent with respect to the orientation of the triangles it contains. Thus, segments are consistently shaped, e.g. all triangles are all mostly co-planar or they are all similarly distributed in orientation. An example result of our segmentation algorithm can be seen in Figure 3.

In the next step, we compute feature vectors for all mesh segments. Here, we use *shape factors*, *shape distributions* based on Euclidean distance, on angles between normal vectors and on the elevation of the normal vectors, and finally *spin images*, where the latter are computed per data point and then an average is computed per mesh segment. As a result, we obtain a 113 dimensional feature vector for each mesh segment, where 50 account for the 5×10 spin image, 20 for each shape distribution (i.e. the number of histogram bins) and 3 for the shape factors. These feature vectors, together with a set of ground truth class labels are then fed into the training algorithm as described below.

3.2 Features for Vision Data

A rich body of work on the detection and classification of road signs and traffic lights has established a successful track record of template-based features for this purpose. Specifically, we leverage the approach proposed by Torralba *et al.* [7] in which a dictionary of partial templates is constructed, against which test instances are matched. A single feature consists of an image patch (ranging in size from 8×8 to 14×14 pixels) and its location within the object as indicated by a binary mask. For any given test instance, the normalised cross-correlation is computed for each feature in the dictionary. Therefore, per instance (or window, in the detection case) a feature vector of length d is obtained, where d is the size of the dictionary. We found empirically that $d > 200$ leads to negligible performance increase in classification. Throughout our experiments we therefore set $d = 200$.

4 Classification for Semantic Mapping

The semantic mapping problem has been investigated already by many other researchers in the robotics community, and a number of different classification algorithms has been used to perform semantic mapping. From all these, the

one that is probably used most frequently is the Support Vector Machine (SVM), which is why we include it here in our analysis. In addition, we also use the Gaussian Process Classifier (GPC), which is still comparably recent and has not been investigated in depth for concrete robotics applications. As we will see later, the GPC yields classification results that are competitive in quality compared to the SVM, but it also provides uncertainty estimates about the labels, that are more intuitive than the ones provided by the SVM. But before, we give a brief summary about SVMs and the GPC.

4.1 Support Vector Machines

SVM classification is well established in robotics so that we provide here only an overview². SVMs are based on a linear discriminant framework which aims to maximise the margin between two classes. The model parameters are found by solving a convex optimisation problem, thereby guaranteeing the final classifier to be the best feasible discriminant given the training data. Once training is complete, predictions on future observations are made based on the signed distance of the observed feature vector from the optimal hyperplane, such that

$$f(\mathbf{x}_*) = \sum_{i=1}^N \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_*) + b, \quad (1)$$

where N is the size of the training set, α_i refers to a Lagrange multiplier associated with datum i , b denotes a bias parameter and $k(\mathbf{x}_i, \mathbf{x}_j)$ denotes the kernel function. Both α_i and b are obtained by training, and α_i is then non-zero only for *support vectors* $\mathbf{x}_i$. The kernel function amounts to a scalar product between two data, which have been transformed from d -dimensional feature space into some higher dimensional space. The nature of this mapping between spaces is inherent in the choice of kernel and need not be specified explicitly (the kernel trick). The regularization and kernel parameters are learnt using cross-validation.

In its original form, the SVM classifier output is an uncalibrated real value. A common means of obtaining a probabilistic interpretation is Platt's method [9]. Here, using a hold-out set not used for classifier training, a parametric sigmoid model is fit directly to the class posterior $p(y_* = C_1 | f(\mathbf{x}_*))$, such that

$$p(y_* = C_1 | f(\mathbf{x}_*)) = \frac{1}{1 + \exp(a f(\mathbf{x}_*) + b)}. \quad (2)$$

The sigmoid parameters a and b are chosen via cross-validation using a model-trust optimisation procedure. Note that class likelihoods are derived here using only a *single* estimate of the discriminative boundary obtained from the classifier training procedure. No other feasible solutions are considered. Hence, the predictive variance of the discriminant $f(\mathbf{x})$ is not taken into account while determining probabilistic output [10]. Further, no guarantees exist that the optimisation itself is well-behaved³.

4.2 Gaussian Process Classification

Let $\mathbf{x} = \mathbf{x}_1, \dots, \mathbf{x}_n$ be a given set of n feature vectors with dimensionality d and $\mathbf{y} = y_1, \dots, y_n$ corresponding class labels where $y_i \in \{1, \dots, k\}$ and k is the number of classes. Here, we first provide a formulation for the general case where $k \in \mathbb{N}$, later we will show experiments with $k = 2$, which simplifies the problem without weakening the significant benefits of the Gaussian Process Classifier (GPC) over other existing methods.

To formulate the Gaussian Process classification problem, a *latent* function $f_j(\mathbf{x})$ is introduced for each class along with the *probit regression* model. The probability of a class label y_i for a given feature vector $\mathbf{x}_i$ is defined as:

$$p(y_i = j | \mathbf{x}_i) = \Phi(f_j(\mathbf{x}_i)) \quad i = 1, \dots, n, \quad j = 1, \dots, k, \quad (3)$$

²For a detailed account the reader is referred to, for example, [8].

³Throughout this work we use LIBSVM [11] for SVM training, calibration and testing.

where Φ denotes the standard normal cumulative distribution function, i.e. $\Phi(z) = \int_{-\infty}^z \mathcal{N}(x | 0, 1)dx$. The latent function f is represented by a Gaussian Process, determined by a mean function – which in our case is the zero function – and a covariance function $k(\mathbf{x}_p, \mathbf{x}_q)$, usually denoted as the *kernel function*. Several different kinds of kernel functions are used in the literature, where the most common ones are the squared exponential, which is also denoted as the Gaussian kernel function. It is defined as

$$k(\mathbf{x}_p, \mathbf{x}_q) = \exp\left((\mathbf{x}_p - \mathbf{x}_q)^T D (\mathbf{x}_p - \mathbf{x}_q)\right) \quad p, q = 1, \dots, n, \quad (4)$$

where D is a $d \times d$ diagonal matrix. The diagonal entries of D are known as the *hyper-parameters* of the model.

In contrast to Support Vector Machines (SVMs), Gaussian Processes are non-parametric, which means that there is no explicit computation of model parameters in the training step. However, in GP classification there still needs to be done some training to obtain the hyperparameters of the covariance function and the posterior distribution of the latent function. More specifically, the aim of the classifier is to find the distribution over values of the latent function given the training data and some test input $\mathbf{x}^*$, i.e.

$$p(f^* | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{y}, \mathbf{x}^*) = \int p(f^* | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{x}^*, \mathbf{f}) p(\mathbf{f} | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{y}) d\mathbf{f}, \quad (5)$$

where we use the notation $\mathbf{x}^*, f^*$ to refer to the test input and its function value. This distribution is then used to compute the class probabilities:

$$p(y^* = j | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{y}, \mathbf{x}^*) = \int p(y^* | f_j^*) p(f_j^* | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{y}) df_j^*. \quad (6)$$

The main problem here is that the latent posterior $p(\mathbf{f} | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{y})$ is not Gaussian and hence Equation (5) can not be computed in closed form. Therefore, approximations need to be done, and the main approaches to do this are the Laplace approximation and Expectation Propagation (EP), as described in [?]. In our work, we employed a variational Bayes formulation as shown by Girolami and Rogers [12] for the multi-class case, and the EP approach for the binary case. In the former, the hyperparameters are learned by gradient ascent on the estimated marginal likelihood. In the latter, the latent posterior is approximated by a Gaussian distribution, where the likelihood means and variances are estimated iteratively using moment matching. Then, in both cases, once the hyperparameters and the latent posterior are obtained from training data, inference is performed on new test input by applying Equation (6).

Compared to an SVM, the GP classification framework offers certain benefits making it particularly suitable for our application. GPs possess a probabilistic formulation and express belief over the latent function via *marginalization* as opposed to *minimization* in SVMs and hence provide uncertainty estimates for the distribution over classification labels [13]. This is more principled than a heuristic approach of using the distance from the classification boundary (margin). A high uncertainty in the GP classification output distribution can give evidence for a category not modeled during training and hence can be used to actively seek examples for incremental training. Additionally, the GP kernel parameters and noise models are interpretable and can be learned without cross validation, which is significant if less data is available for a rare category.

5 Experiments

In this section, we present results obtained from experiments that were carried out for the two different sensor modalities as mentioned earlier. For the laser data, we use a multi-class version of the GP classifier, while a binary GPC was used on a traffic sign data set acquired with a monocular camera. This difference, however, has no influence on the results, and we will see that for both kinds of input data, the GPC performs better in terms of introspection than the SVM.

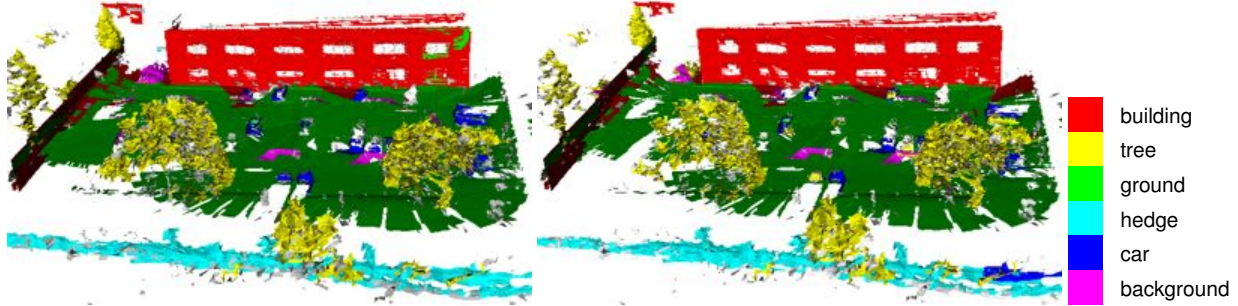


Figure 4: Classification result on laser data after 9 point clouds (time steps). **Left:** Ground truth. Note that even in the ground truth some areas are not labeled correctly, e.g. on the ground close to the *building*. This is due to the fact that the mesh segmentation is not perfect and a correct manual labeling of segments that actually correspond to more than one class is not possible. In our evaluation we abstract from such segmentation errors. **Right:** Classification result. Only minor errors are visible. Note again the *hedge* in the front, but also on some *cars*.

5.1 Experiments with Laser Data

Figure 4 shows the classification result of 9 meshes corresponding to 3D point clouds that were acquired consecutively. We note that there are slight labeling errors even in the ground truth (left image). This is caused by imperfections in the segmentation process, which lead to under-segmentation. For example, some few segments contain sensor readings from the building and the ground. As it is impossible to determine a single *true* label for these segments, we decided to assign the ground truth label based on a majority voting during annotation. From the figure we can see that in general, the classification is very good, only the under-represented classes such as *car* and *hedge* are classified slightly worse.

To evaluate how useful the obtained label uncertainty is, we compared the uncertainty estimates of the probabilistic classification output of the GP and the SVM classifier to new object classes not used in training. We trained both classifiers only on segments from two classes (randomly picked): *building* and *ground*. Data from the remaining *un-modeled* classes was presented to both classifiers for inference, resulting in a classification distribution over binary labels. The normalized entropy values measuring uncertainty in the classification decision were computed for each label distribution.

Figure 5 presents the normalized entropy histograms for the inference set. The SVM classifier commits a large majority of the *un-modeled* points to one of the *modeled* classes with high certainty, resulting in a peaked distribution over one of the two labels. As a result, for a majority of the data points, the label distribution has lower normalized entropy. In contrast, the GP classifier assigns higher normalized entropy for a majority of the test points. The same pattern was found consistent for other choices of training and testing classes. The classifier uncertainty for the test points from new classes is expressed as a more uncertain (uniform) distribution over labels, indicating the presence of one or more potentially *un-modeled* classes.

5.2 Experiments with Vision Data

As examples of classes typically encountered in autonomous driving applications we use a subset of the German Traffic Sign Benchmark (GTSRB) dataset [14], which comprises over 50,000 loosely-cropped images of 42 classes of road signs, with associated bounding boxes and class labels. From this dataset we specifically focus on the seven classes shown in Table 1. First we analyse the classification performance of four different classifiers, namely the GPC with a linear and a squared-exponential (SE) kernel, and the SVM, also using a linear and a squared-exponential kernel, while the latter is, in the context of SVMs, more often referred to as the radial-basis function kernel (RBF).

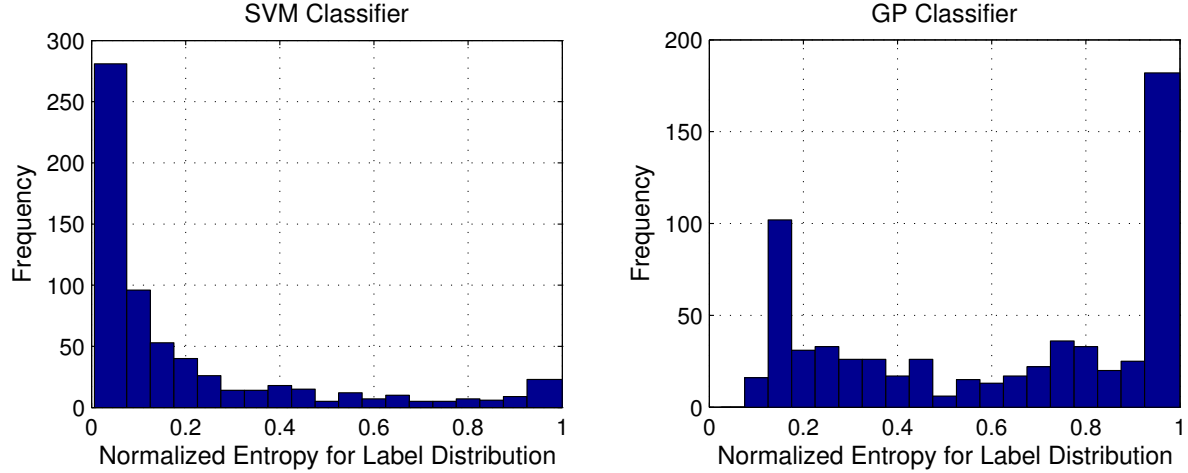


Figure 5: Histogram of normalized entropy values of the label distribution for SVM and GP classifiers, evaluated on laser data. Both classifiers were trained explicitly on two classes. The data from the remaining classes was presented for inference. **Left:** SVM classifier assigns a majority of the points to a particular class with high certainty. **Right:** As a contrast, GP classifier assigns greater classification uncertainty to a majority of the points, providing evidence for a potential new class. Note the scale on y-axis.



Table 1: The seven classes of the German Traffic Sign Recognition Benchmark dataset considered in our work. The numbers in brackets indicate the number of data available per class.

Classifier	Precision	Recall	F_1
SE GPC	1.000	0.990	0.995
RBF SVM	1.000	0.995	0.997
Linear GPC	1.000	0.990	0.995
Linear SVM	1.000	0.990	0.995

Table 2: Classification performance when separating *stop* sign from the *lorries prohibited* signs. Note that different class combinations were found to yield classifiers of similar quality.

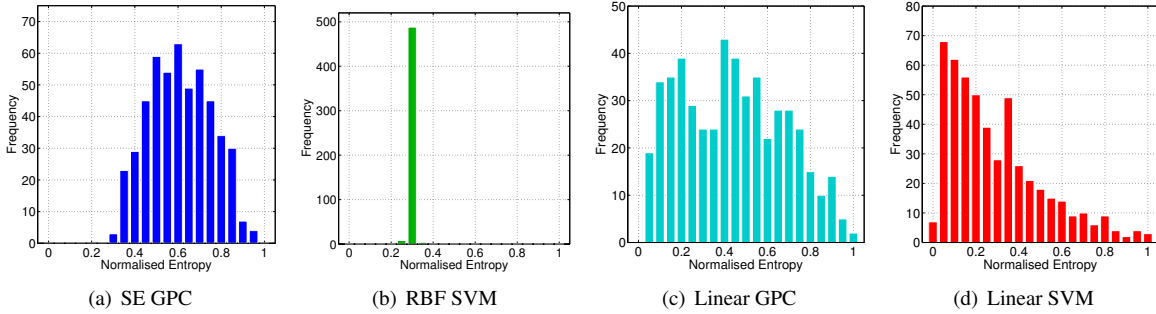


Figure 6: Normalised entropy histograms of the marginal probabilities for four classifiers trained on the road sign classes *stop* and *lorries prohibited* and tested on 500 instances of the unseen class *roadworks ahead*. Higher normalised entropy implies more uncertainty in classifier output. Note that the mean normalised entropy for the SE GPC is higher than that of the others.

We arbitrarily selected two classes for training: *stop* and *lorries prohibited*. To investigate the efficacy of the features used and training procedures employed, classifiers were trained separating these two classes using a balanced training set of 400 data (200 per class) and applying a canonical training procedure for each classifier type, including five-fold cross-validation where appropriate. Classifier performance was evaluated using standard metrics on a hold-out set of another 400 class instances (200 of each class) of the same two classes. The results are shown in Table 2. Classification performance is commensurate across all classifiers. The corresponding precision-recall curve confirms the near-perfect separation of the classes and has been omitted here as it is otherwise uninformative. The classifiers are next retrained using the full 800 training data (400 per class) and the same canonical training procedures. They are then applied to 500 instances of the previously unseen class *roadworks ahead*. The resulting normalised entropy histograms are shown in Figure 6. The mean normalised entropies for the GPC-based classifiers are significantly higher than those of the other classification frameworks, indicating that the GPC-based classifiers exhibit greater uncertainty in their judgement. Conversely, the RBF SVM classifier is extremely confident in its classifications with a very narrow distribution around a relatively low value of normalised entropy. This was an effect consistently observed throughout our experiments, which we attribute to the relatively gradual decay of the estimated class posterior probabilities through feature space often encountered far away from the decision boundary. Features from an unseen class which are located in feature space at a distance from the decision boundary therefore only span a very narrow range of estimated class posterior probabilities.

In order to mitigate any influences of the specific feature set used and the specific training and test data selected we repeated the above experiment across a number of random dictionaries, data samples and unseen classes. Specifically, for each of five different unseen classes, we perform ten iterations of classifier training and testing with a random dictionary and training and test datasets resampled for each run. The results, presented in Table 3, are consistent with those in Figure 6 in that the GPCs tend to be more uncertain while SVM and LogitBoost are more confident with an often significantly narrower distribution of normalised entropy values. The results in Table 3 indicate that the gap in uncertainty between the different frameworks is more pronounced for some unseen classes than for others. We attribute this to the varying degree of similarity in feature space between the unseen class and the classes in the training set. A

Test Class	Classifier	Normalised Entropy	
		$\mu \pm \text{std. err.}$	$\sigma \pm \text{std. err.}$
	SE GPC	0.504 $\pm$ 1.92E-03	0.110 $\pm$ 9.35E-05
	RBF SVM	0.313 $\pm$ 1.33E-04	0.012 $\pm$ 2.12E-06
	Lin GPC	0.245 $\pm$ 9.34E-04	0.173 $\pm$ 9.19E-05
	Lin SVM	0.106 $\pm$ 4.77E-04	0.107 $\pm$ 2.51E-04
	SE GPC	0.487 $\pm$ 1.70E-03	0.139 $\pm$ 7.67E-05
	RBF SVM	0.310 $\pm$ 1.13E-04	0.017 $\pm$ 3.72E-06
	Lin GPC	0.286 $\pm$ 8.09E-04	0.179 $\pm$ 6.24E-05
	Lin SVM	0.076 $\pm$ 3.72E-04	0.097 $\pm$ 2.43E-04
	SE GPC	0.723 $\pm$ 4.91E-04	0.186 $\pm$ 1.59E-04
	RBF SVM	0.306 $\pm$ 1.03E-04	0.095 $\pm$ 7.96E-05
	Lin GPC	0.680 $\pm$ 4.75E-04	0.235 $\pm$ 1.11E-04
	Lin SVM	0.634 $\pm$ 7.29E-04	0.267 $\pm$ 4.28E-05
	SE GPC	0.804 $\pm$ 6.08E-04	0.163 $\pm$ 1.72E-04
	RBF SVM	0.335 $\pm$ 1.43E-04	0.050 $\pm$ 1.26E-05
	Lin GPC	0.811 $\pm$ 4.39E-04	0.184 $\pm$ 1.66E-04
	Lin SVM	0.642 $\pm$ 3.24E-04	0.294 $\pm$ 9.19E-05
	SE GPC	0.259 $\pm$ 2.36E-03	0.116 $\pm$ 1.37E-04
	RBF SVM	0.255 $\pm$ 1.28E-04	0.027 $\pm$ 5.26E-06
	Lin GPC	0.155 $\pm$ 9.27E-04	0.140 $\pm$ 2.61E-04
	Lin SVM	0.043 $\pm$ 7.82E-05	0.059 $\pm$ 1.26E-04

Table 3: Mean and standard deviation normalised entropies (including standard errors) from ten iterations of classifier training and testing, each with a randomly created dictionary and both training and test datasets resampled. Results are presented for classifiers trained on the road sign classes *stop* and *lorries prohibited* and tested on five different unseen classes as shown.

more in-depth analysis of this phenomenon remains future work.

6 Conclusions

The semantic mapping module developed in the V-Charge project has gained further functionalities and capabilities compared to the template-matching approach described in the earlier Deliverable D2.1. In particular, we have added the potential to semantically annotate geometric data given by 3D point clouds, as well as images taken from a monocular camera similar to the ones used on the V-Charge vehicles. Furthermore, we have investigated introspective classification as a useful additional source of information, and as a powerful method to potentially improve further processing such as navigation and localisation, especially in safety-critical application scenarios such as the one investigated in this project.

Further work will be focussed on a tighter integration into the overall framework of the project, but also on improvements and extensions of the proposed method. In particular, a multi-class version of the GPC for image data, a more time- and memory-efficient version of the classifier, and a combination of geometric and appearance based information for classification are currently under investigation.

References

- [1] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors - a survey. *Foundations and Trends in Computer Graphics and Vision*, 2008.
- [2] Changchang Wu, B. Clipp, Xiaowei Li, J.-M. Frahm, and M. Pollefeys. 3d model matching with viewpoint-invariant patches (vip). In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.
- [3] Bernhard Zeisl, Kevin Koeser, and Marc Pollefeys. Viewpoint Invariant Matching via Developable Surfaces. In *Europ. Conf. on Computer Vision (ECCV), WS on Consumer Depth Cameras in Computer Vision (CDC4CV)*, 2012.
- [4] Rohan Paul, Rudolph Triebel, Daniela Rus, and Paul Newman. Semantic categorization of outdoor scenes with uncertainty estimates using multi-class Gaussian process classification. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*, 2012.
- [5] Hugo Grimmett, Rohan Paul, Rudolph Triebel, and Ingmar Posner. Knowing when we don’t know: Introspective classification for mission-critical decision making. In *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2013. under review.
- [6] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004.
- [7] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing visual features for multiclass and multi-view object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(5):854–869, May 2007.
- [8] C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [9] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances In Large Margin Classifiers*, pages 61–74. MIT Press, 1999.
- [10] C.E. Rasmussen and CKI Williams. Gaussian processes for machine learning, 2006. *The MIT Press, Cambridge, MA, USA*.
- [11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [12] M. Girolami and S. Rogers. Variational bayesian multinomial probit regression with gaussian process priors. *Neural Computation*, 18(8):1790–1817, 2006.
- [13] CKI Williams and CE Rasmussen. Gaussian processes for machine learning, 2006.
- [14] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012.

Viewpoint Invariant Matching via Developable Surfaces

Bernhard Zeisl, Kevin Köser, Marc Pollefeys

Computer Vision and Geometry Group
ETH Zurich, Switzerland
{zeislb,kkoeser,pomarc}@inf.ethz.ch

Abstract. Stereo systems, time-of-flight cameras, laser range sensors and consumer depth cameras nowadays produce a wealth of image data with depth information (RGBD), yet the number of approaches that can take advantage of color and geometry data at the same time is quite limited. We address the topic of wide baseline matching between two RGBD images, i.e. finding correspondences from largely different viewpoints for recognition, model fusion or loop detection. Here we normalize local image features with respect to the underlying geometry and show a significantly increased number of correspondences. Rather than moving a virtual camera to some position in front of a dominant scene plane, we propose to unroll developable scene surfaces and detect features directly in the “wall paper” of the scene. This allows viewpoint invariant matching also in scenes with curved architectural elements or with objects like bottles, cans or (partial) cones and others. We prove the usefulness of our approach using several real world scenes with different objects.

1 Introduction and previous work

Utilizing image based features to compactly describe image content or to identify corresponding points in images has become a de facto standard in computer vision over recent years. Applications where a feature based approach proved particularly successful are for example structure-from-motion, image registration, visual SLAM systems or object detection and recognition. However, a major problem when trying to find correspondences between widely separated views is that the appearance of objects can change drastically with viewpoint. To remedy this problem techniques have been developed which normalize images or image regions such that they become (at least approximately) invariant to viewpoint changes. In case one matches two images (without depth information) against each other the most popular method is to use local image features that compensate for the first order effects of viewpoint change by normalization, i.e. affine transformations (cf. to [1, 2]) or slightly weaker models (e.g. [3]). Since scale, orientation and (anisotropic) stretch are all effects that could have been caused by a viewpoint change they need to be factorized out and therefore it is not possible to distinguish, e.g. real-world circles from ellipses or a small round dot from a huge sphere any more. This is a general dilemma of discriminative power



Fig. 1. Exemplary objects and their developed surfaces. To normalize wrt. viewpoint changes we propose to detect and describe features in the unrolled surface textures rather than in the original images.

vs. invariance. For an in depth discussion of invariant feature constructions we refer the reader to [4].

When depth information is available one can normalize wrt. the given 3D structure. In [5, 6] the authors have shown that it is possible to virtually move a camera to a frontal view and then render a canonical representation of a local image feature. Similarly, normalization can also be obtained by extracting vanishing points in Manhattan scenarios [7, 8] and virtually rotating the camera. However, all these approaches have strong limitations: While [5] still requires an affine detector and thus the number of features obtained is limited, other approaches [6–8] rely on the existence of dominant scene planes. In contrast to this assumption we observe that many structures in our environment are also curved, e.g. like cylinders, cones or consist of free-form shapes. Many man-made objects are made by bending sheets or plates and thus - by construction - form *developable surfaces* that can virtually be “unrolled” when their geometric structure (depth) is known (see Fig. 1). For the particular application of pose-robust face recognition, Liu and Chen [9] coarsely approximate a human head via a 3D ellipsoid and back-project images onto its surface. Recognition is then conducted in the flat, but stretched and distorted texture maps. In comparison, we are interested in objects possessing developable surfaces and by this allow to create an undistorted texture map.

Further, in this work we follow the idea to develop such observed scene surfaces and to extract image features in the flat 2D wall-paper version of that very same surface, allowing for less invariant (and more discriminative) detectors/descriptors. In case scale is known (e.g. from a Kinect camera) the affine or perspective invariance requirement of the original problem is reduced to rotation in the image plane and can be reduced even further for surfaces such as cones or cylinders. We strongly believe that this technique is useful in several applications such as robotic scenarios, where a robot has to identify and manipulate an object, for automatically registering overlapping 2.5D or 3D models or loop detection in large structure-from-motion or SLAM systems.

The paper is organized as follows: Developable surfaces are discussed in Sec. 2, while Sec. 3 presents our algorithmic approach and Sec. 4 states implementation details. In Sec. 5 we illustrate experiments and results on both, data from active and passive consumer depth sensors, followed by concluding remarks.



Fig. 2. Developable surfaces present in our environment. Note, that in the left image only cones are highlighted although planes and cylinders exist as well.

2 Developable Surfaces

As our approach builds on the notion of developable surfaces, we start by briefly introducing the underlying concept. In general a surface with *zero* Gaussian curvature at every surface point is developable [10] and can be flattened onto a plane without distortion (such as stretching or shortening).

To determine the Gaussian curvature of a surface, suppose we are given a smooth function s that maps 2D parameters u, v to points in 3D space, i.e. $s : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ such that $s(u, v) = (x, y, z)^T$. The graph S of this function is a two-dimensional manifold and our surface of interest in 3D space. The derivatives $s_u = \frac{\partial s}{\partial u}$ and $s_v = \frac{\partial s}{\partial v}$ of s with respect to the parameters u and v define tangent vectors to the surface at each point. Their cross product yields the normal vector $n = s_u \times s_v$ to the surface. The second partial derivatives of s with respect to u, v are now used for constructing the shape operator

$$II = \begin{pmatrix} L & M \\ M & N \end{pmatrix} = \begin{pmatrix} n^T s_{uu} & n^T s_{uv} \\ n^T s_{uv} & n^T s_{vv} \end{pmatrix}, \quad (1)$$

which is also called the second fundamental form of s . The principal curvatures κ_1, κ_2 of the surface at a given position are defined as the eigenvalues of II . They measure how the surface bends by different amounts in different directions at a particular point. Finally, the determinant $\det(II) = \kappa_1 \kappa_2$ denotes the Gaussian curvature; in case it vanishes everywhere on the surface (at least one of the eigenvalues is zero) the surface is developable. The intuition is that in direction of zero curvature the surface can be described as a line. Hence, the surface development is just an unrolling of all corresponding lines into one plane. We refer the interested reader to [10] for more details.

For example a cylinder is developable, meaning that at every point the curvature in one direction vanishes. Its mean curvature is not zero, though; hence it is different from a plane. Contrary, a sphere is not developable, since its surface has constant positive Gaussian curvature at every point. Other basic developable shapes are planes, cylinders, cones and oloids¹ and variants thereof, such as cylindroids, or oblique cones. Intuitively, they are flattened by rolling the object on

¹ An oloid is defined as the convex hull of two equal disks placed at right angles to each other, so that the distance between their centers is equal to their radius.

a flat surface, where it will develop its entire surface. In fact, all surfaces which are composed of the aforementioned objects are developable as well. In practice, many objects in our environment are made by bending sheets or plates and thus form developable surfaces. Fig. 2 illustrates several real-world developable surfaces; note that even such complex structures as the church roof top (Fig. 2 very right) are (piece-wise) developable.

3 Exploiting Developable Surfaces for Viewpoint Invariant Matching

In the following we present our approach of matching two views of a rigid scene, separated by a wide-baseline, by means of developable surfaces. However, we point out that the same techniques are applicable for identifying and recognizing a single object in a database, for loop detection or for automatically registering multiple overlapping textured 3D models. As input to our algorithm we assume two RGBD images with sufficient overlap. Given pixel-wise depth measurements $d_{u,v}$ and camera intrinsics K a 2.5D point cloud is obtained per view via

$$(x, y, z)_{u,v}^T = K^{-1}(u, v, 1)^T d_{u,v} \quad \forall u, v \in I, \quad (2)$$

with image coordinates u, v . Then our method progresses in four steps, which are (a) detection and parameter estimation of certain developable surfaces in the depth data, (b) generating flat object textures by means of developing the detected surfaces, (c) detecting/describing features in the unrolled images (i.e. in the surface) and matching against the other views, and (d) verification of found correspondences. We will explain them in more detail in following Sec. 3.1 to 3.4.

3.1 Multi-Model Estimation

As described in the previous Sec. 2, many different developable surfaces exist. In this paper we focus on three basic shapes, the plane, the cylinder and the cone, because these shapes possess a low parametric representation and thus are detected reliably in depth data. Identifying these surfaces falls into the category of multi-model estimation and several techniques have been suggested to cope with it, including randomized hough transform, sequential RANSAC or more recently J-Linkage [11], multi-structure segmentation [12] or more problem-specific machine-learning inspired geometric classification approaches in the spirit of [13].

3.2 Developing Surfaces

Subsequent to the initial model estimation and parameterization is the generation of a flat texture per detected model. We describe obtained mappings for principal geometric shapes in the following.

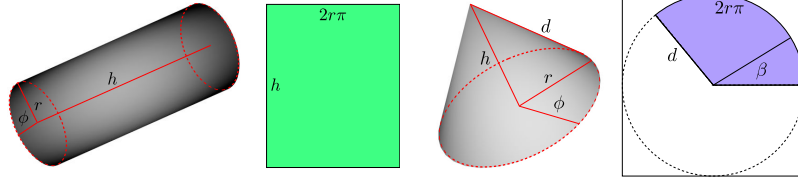


Fig. 3. (left) Cylinder with height h and radius r and its developed texture of dimensions $h \times 2\pi r$. (right) Cone with height h and radius r and its developed texture with radius d . Note that angles ϕ around cone axis and β in developed texture differ by $\frac{r}{d}$ and thus a 360 turn of the cone does not describe a full circle.

Planes From the previous model estimation (Sec. 3.1) we know a four parameter description $\pi_S = (\mathbf{n}^T, d)$ with normal vector $\mathbf{n}$ and distance to the origin d , as well as a bounding box (or mask) for the region of interest on the 3D plane. Two orthogonal vectors $\mathbf{u}, \mathbf{v}$ in the plane are chosen as a basis $\mathcal{B}_S$ and we sample the plane in equidistant steps, i.e. we define a grid in the plane. Original image plane π_I and surface plane π_S together with the origin of $\mathcal{B}_S$ define a unique mapping $\mathbf{P}$. It is used to project each of the grid vertices (u_i, v_j) into the original image to obtain the appropriate color. The resolution is chosen such that we do not lose any image details; this means we project the four bounding box corners into the original image and evaluate the Jacobian matrix of the texture warp for some arbitrary grid resolution. Afterwards we increase or decrease the grid resolution such that the smallest minification between the developed surface and the original image is 1 (for details on texture mapping see [14]). In practice the transformation $\mathbf{P}$ will be a homography and the result is equivalent to [6]. Here, the frontal view of the plane coincides with the developed plane, however we will now generalize this to other developable surfaces.

Cylinders After model detection a cylinder is parameterized as $(\mathbf{c}, \mathbf{a}, r)$ with cylinder base center $\mathbf{c}$, axis vector $\mathbf{a}$ of length h and radius r . In order to unroll the cylinder we represent 3D points on the surface in their cylinder coordinates (r, ϕ, z) (with $\mathbf{c}$ as the origin). By removing the radius coordinate we obtain a 2D parameterization (ϕ, z) of the surface. The projection of surface points into the image plane π_I is thus defined by a unique mapping $\mathbf{P}$. The angular resolution in ϕ is determined to match the resolution along the cylinder axis and to obtain an image of aspect ratio $h \times 2\pi r$ for a full 3D development of the cylinder (see Fig. 3). In case scale is known (e.g. when a Kinect camera is used) and when it is desirable not to normalize over scale (e.g. because of similar features at different scales) we choose a metric surface resolution. Otherwise, we evaluate the local magnification/minification between the original image and the surface texture and ensure that no resolution is lost during unrolling. Given 2D coordinates (ϕ_i, z_j) in the unrolled surface texture, each corresponding 3D surface point (r, ϕ_i, z_j) is identified and projected into the original image via $\mathbf{P}$.

to obtain the color. This mapping is very efficiently implemented on the GPU or using standard backward mapping on the CPU.

Cones A cone is parameterized and developed very similarly to the cylinder, taking into account that the surface tapers smoothly towards the apex. To obtain a flat surface texture, it is positioned with a line from the apex to the base circle of length $d = \sqrt{r^2 + h^2}$ (see Fig. 3) in the plane for development (imagine laying it on a piece of paper). Afterwards the apex is fixed and the cone rolled around it, resulting in a circle segment. The created circular texture contains the apex in the center, where the radial lines connect the apex with the cone’s base circle (consequently of radius d , up to resolution). Thus 2D texture coordinates (β_i, d_j) are directly related to points on the cone surface. Similar to the cylinder, we backward map texture coordinates across the 3D surface into the original image to obtain the colors for the surface texture, maximizing its resolution.

3.3 Feature Detection and Matching

Feature detection is performed directly in the unrolled textures. This is conceptually different to [1, 5] which first detect features and then try to normalize these wrt. to viewpoint variations. It is related to [6–8], however these approaches only consider planes for normalization. The unrolled textures allow to reach perspective invariance with only normalizing in-plane rotation in the image (or similarity normalization in case absolute scale is unknown). Even better, since cylinder² and cone define an inherent reference direction with their axis, all features can be expressed with respect to this orientation rather than computing an orientation from the local region as for example in SIFT [3]. Consequently it is possible to extract very basic features on the surface, which is very fast on the one hand and on the other hand allows to distinguish local regions that differ only by scale, orientation or linear shape. All detected features on the different developed textures are combined to form the set of features for a RGBD image and are subsequently used for wide-baseline matching.

3.4 Correspondence Verification

Naturally the set of estimated matches contains numerous outliers, which do not satisfy the underlying camera pose change. Therefore, correspondences are checked using geometric verification (e.g. using RANSAC). As observed in [6], each feature does not only include information about the 3D position, but also the local normal. Additionally, when orientation is known from the cylinder or cone geometry or when local orientations are estimated from gradients [3], three characteristic directions are known at each 3D feature point. This allows for a stratified verification as in [6] or for a minimal solution in RANSAC that requires only a single correspondence.

² For the cylinder there is still a 180° ambiguity for the direction of the axis.

4 Implementation Details

We employ RANSAC to obtain model parameters for planes, cylinders and cones in the captured RGBD data. Since surfaces are mostly local and continuous we utilize a local sampling strategy, where consecutive samples are drawn within a 0.5m radius. Surface models are searched for in order of increasing complexity, i.e. initially planes are detected, followed by cylinders and cones. We limit the size of models to physically plausible extents for the expected outdoor or indoor environments. In addition, found models need to guarantee that they show sufficient support over their surface to avoid algorithmic plausible, but incorrect estimations. Consequently, we reject models whose support is only defined at isolated points or clusters. Once detected, we robustly determine the model size in the image and estimate the spatial extent (e.g. height of cylinder). Subsequently, initial model parameters are updated via a non-linear optimization on the evaluated inlier set. After each iteration 3D points supporting the estimated model are removed from the search space, which prohibits assignment to multiple models. This iterative procedure terminates as soon as no model with sufficiently large support is found any more.

As mentioned in Sec. 3.3, image feature estimation in the developed surfaces can be accomplished with a basic detector such as a Harris corner detector. However, since we aim to compare obtained matches from developed surfaces with matches in the original RGBD data, we chose standard SIFT as our detector and descriptor to guarantee comparability. (Note, that employing upright-SIFT would also treat our approach with favor due to its greater discriminative power.) Detected image features are matched against each other in their descriptor space. To eliminate ambiguous matches (e.g. between repetitive structures) all best matches are kept for which the distance ratio to the second best match falls below 0.6 (known as the ratio test in [3]). Then, each feature is additionally augmented by its position in 3D space, which is determined by the corresponding 3D surface model (according to the derived mappings in Sec. 3.2). For features in the original RGBD images we consider present pixel-wise depth measurements and neglect them in case no depth is available, e.g. due to occlusions.

To obtain a robust estimate of correct matches, we employ a correspondence rejection method via RANSAC, which samples from the feature correspondence set and estimates a 6 DoF transformation between the two views. Since correspondences between 3D points are explicitly defined, we utilize Procrustes analysis [15] (and do not need an iterative non-linear optimization scheme) for the estimation of rotation and translation. Finally, the estimated transformation is validated on all potential correspondences, which gives a final set of correct and consistent matches. These consistent matches are visualized in the different experiments presented in the following.

5 Results for Active and Passive Stereo Devices

In this section we demonstrate our novel technique for different scenes and cameras. Fig. 4, Fig. 5 and Fig. 6 illustrate obtained results for a synthetic setup,



Fig. 4. Wide baseline matching for a synthetic setup. (top row) Outer left and right image illustrate the found models in the 2.5D point cloud; images in-between show developed textures. (bottom row) SIFT matches consistent wrt. the underlying 6DoF transformation between the initial RGBD images and between developed surfaces.

Scene type	Synthetic setup	Floor (Kinect)	Table (Kinect)	Trees (stereo)	Pylon (stereo)
Matches orig. RGBD images	37	9	27	3	56
Matches developed surfaces	255	79	195	22	227
Enhancement ratio	6.89	8.78	7.22	7.33	4.05

Table 1. Quantitative comparison of SIFT descriptor matches between original RGBD images and developed surfaces for different scenes.

indoors scenes captured with a Kinect camera and outdoor scenes taken with a Fuji3D stereo camera, respectively. Rectified textures of detected planes are not illustrated due to space limitations; though, they are included in the evaluation.

Comparing feature detection and matching in the original images and in the images of developed surfaces (see Tab. 1) we can record the following: While approximately the same number of features are detected and an equal amount of potential matches is obtained, evaluation shows that for the latter the amount of finally remaining *correct* matches is significantly larger. Between the original RGBD images many potential matches are wrong due to viewpoint distortions in the descriptor space and thus need to be rejected. This validates that our approach of viewpoint invariant description of developable surfaces is able to extract features, which are stable over a variety of largely different viewpoints and improves wide-baseline matching considerable. In addition, rather than interpreting our approach as a competitor to standard feature matching, one should see it as an additional cue for obtaining more stable features.

6 Conclusion and Future Work

We have presented a novel technique to exploit depth information for viewpoint invariant matching between RGBD images. It develops the surface and detects

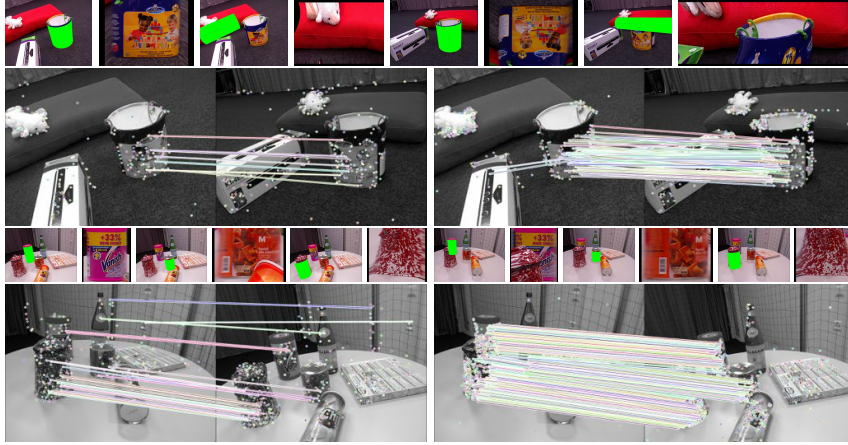


Fig. 5. Wide baseline matching for RGBD data captured with a Kinect sensor (Floor and Table scene). (1st and 3d row) Detected objects (green) and their respective developed surface for the two views. (2nd and 4th row) Consistent SIFT matches between original images and developed surfaces, respectively.

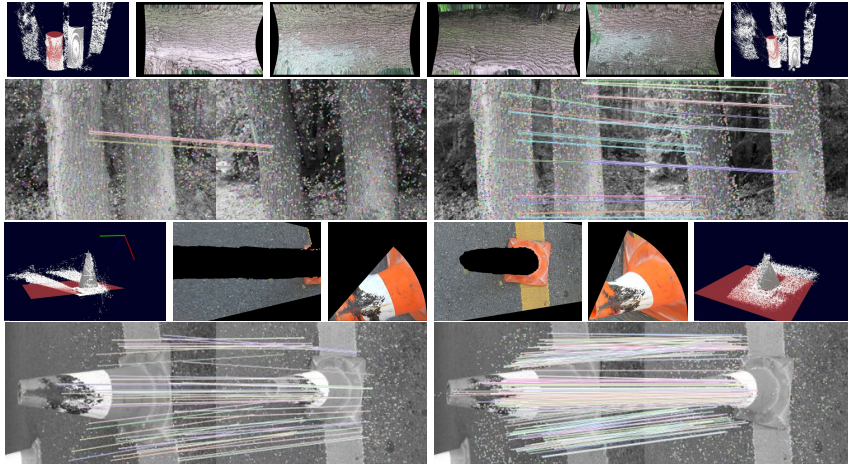


Fig. 6. Wide baseline matching between images taken by a Fuji3D consumer stereo camera (Trees and Pylon scene). (1st and 3rd row) Detected models in the 2.5 point cloud and their respective developed surfaces. (2nd and 4th row) SIFT feature matches between original scenes and developed surfaces, respectively. Note, that for the bottom experiment depth estimates are noisy and contain a considerable amount of errors, leading to degraded parameter estimation for the detected cone.

features in the surface's wall paper which removes perspective effects. While we have shown drastically increased number of matches as compared to classical SIFT, the feature detector can actually be chosen freely for the given application. Scale can easily be integrated (when known from the range sensor) to allow for using simple and more discriminative features such as Harris corners and in the special case of cylinders and cones even in-plane rotation is known. In any case, while we have demonstrated the approach as a competitor that outperforms standard SIFT by far, it should be clear that both can easily be combined for a real system. Compared to earlier viewpoint normalization approaches that relied on global scene planes we have shown that many other geometric shapes are feasible and demonstrated this using cones and cylinders. Possible further work will include the extension from detectable parametric surfaces to flattening of general (approximately developable) surfaces.

References

1. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Van Gool, L.: A Comparison of Affine Region Detectors. *IJCV* **65**(1-2) (2005) 43–72
2. Mikolajczyk, K., Schmid, C.: A Performance Evaluation of local Descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(10) (2005) 1615–1630
3. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. *IJCV* **60**(2) (2004) 91–110
4. Gool, L.V., Moons, T., Pauwels, E., Oosterlinck, A.: Vision and Lie's Approach to Invariance. *Image and Vision Computing* **13**(4) (1995) 259 – 277
5. Köser, K., Koch, R.: Perspectively Invariant Normal Features. In: *ICCV, Workshop on 3D Representation and Recognition*. (2007)
6. Wu, C., Clipp, B., Li, X., Frahm, J.M., Pollefeys, M.: 3D Model Matching with Viewpoint-Invariant Patches (VIP). In: *Proc. CVPR*. (2008) 1–8
7. Robertson, D., Cipolla, R.: An Image-Based System for Urban Navigation. In: *Proc. BMVC*. (2004) 819–828
8. Cao, Y., McDonald, J.: Viewpoint Invariant Features from Single Images using 3D Geometry. In: *Workshop on App. of Comp. Vision*. (2009) 1–6
9. Liu, X., Chen, T.: Pose-robust Face Recognition using Geometry Assisted Probabilistic Modeling. In: *Proc. CVPR*. Volume 1., IEEE (2005) 502–509
10. Kuehnel, W.: *Differential Geometry : Curves - Surfaces - Manifolds*. American Mathematical Society, Providence, R.I (2006)
11. Toldo, R., Fusiello, A.: Robust Multiple Structures Estimation with J-Linkage. In: *Proc. ECCV*. (2008) 537–547
12. Wang, H., Chin, T.J., Suter, D.: Simultaneously Fitting and Segmenting Multiple-Structure Data with Outliers. *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(6) (June 2012) 1177 –1192
13. Rusu, R., Holzbach, A., Blodow, N., Beetz, M.: Fast Geometric Point Labeling using Conditional Random Fields. In: *Proc. IROS, IEEE* (2009) 7–12
14. Heckbert, P.S.: *Fundamentals of Texture Mapping and Image Warping*. Technical report, U.C Berkeley (1989)
15. Eggert, D., Lorusso, A., Fisher, R.: Estimating 3-D Rigid Body Transformations: A Comparison of Four Major Algorithms. *Machine Vision and Applications* **9**(5) (1997) 272–290