Matching Features Correctly through Semantic Understanding

Nikolay Kobyshev* and Hayko Riemenschneider* and Luc Van Gool*†
* Computer Vision Laboratory, ETH Zurich, Switzerland
†K.U. Leuven, Belgium
{nk,hayko,vangool}@vision.ee.ethz.ch

Abstract

Image-to-image feature matching is the single most restrictive time bottleneck in any matching pipeline. We propose two methods for improving the speed and quality by employing semantic scene segmentation. First, we introduce a way of capturing semantic scene context of a keypoint into a compact description. Second, we propose to learn correct matchability of descriptors from these semantic contexts. Finally, we further reduce the complexity of matching to only a pre-computed set of semantically close keypoints. All methods can be used independently and in the evaluation we show combinations for maximum speed benefits. Overall, our proposed methods outperform all baselines and provide significant improvements in accuracy and an order of magnitude faster keypoint matching.

Figure 1: Scene understanding for feature matching. The search area for a feature can be significantly reduced by its semantic context. For example, a search area for a feature located on the sidewalk takes only 10% of the image area.

1. Introduction

Keypoint matching plays a major role in many computer vision tasks. It is used for image registration, image retrieval, localization and so on. For example, in classical Structure-from-Motion (SfM) pipeline [12], keypoint matching is used as one of the initial steps. Once image features are matched correctly, it is possible to triangulate these correspondences in 3D space and then infer the camera positions of their respective images.

There has been tremendous progress in all parts of these tasks over the last years [17, 10]. However, matching remains one of the most time-consuming operations. Although there are fast methods for finding the features in an image (for example, SIFT [16] or SURF [3]), and the problem is well-parallelizable [1, 9], matching the features efficiently remains a major research focus.

The problem of feature matching is challenging on two levels. First, matching every image to every other one in a dataset is an operation of quadratic complexity. Without prior knowledge of the scenes or time constraints all images have to be matched to all other images, yet this complete matching is often not required. According to [25], 75%–98% of the images in large photo collections will not match because they are not even overlapping. To levitate some of this problem, extra knowledge (for example, the sequence in which the images were taken, coarse GPS location or viewing directions) can reduce the matching set. Alternatively, retrieval techniques can be used to reduce the number of feasible pairs to be computed. In all of these approaches, instead of being matched with all the other images, the image is only matched to a smaller subset of the entire dataset.

The second challenge lies on the image-to-image level. A naïve approach would match every feature of the first image to every feature in the second image, resulting again in quadratic complexity of the procedure. There are two principally different methods of tackling this problem. First, one may optimize the speed of the feature search, for example, by employing efficient data structures such as KD trees [4] or vocabulary trees [18]. Another approach is to try to eliminate the number of keypoints prior to matching, reducing the number of outliers (points that do not have a valid match) as well as the points that are not significant for the purpose of matching.

In this work [1], we focus on the image-to-image level and propose novel approaches to reduce the number of candidate features. Our approach relies on understanding of what we are trying to match. Using the results of a coarse semantic scene segmentation, we suggest several methods to significantly reduce and improve feature matching. In summary, we classify every part of an image into its semantic categories like road, building, sky, objects, vegetation, etc. and then limit the feasible sets of candidate to those which have a similar semantic context. The use of semantic context in feature matching provides several benefits w.r.t. standard matching as we can efficiently prune incorrect matches. An example of image segmentation is provided in Figure 1. It can be seen that knowing the semantic context of a feature can dramatically reduce the search space. Moreover, one can learn which semantic context makes a feature reliable for matching. This is demonstrated in Figure 2: the classifier omits the points that belong to the semantic areas that are usually matched incorrectly. The main contributions of our work are:

- novel feature matching based on scene understanding to capture semantic context
- reduction of search space by semantic indexing
- prediction of features likely to be correctly matched before actual matching

2. Related work

There is a significant amount of research for efficient feature matching in general. The approaches can be summarized into these main fields – interest point detection and description, search structures for matching, and filtering of features prior to matching.

For detection and description, the dominant methods are Scale-Invariant Feature Transform points (SIFT) [16] or Speeded Up Robust Features (SURF) [3]. The latter uses Haar wavelets to approximate the determinant of the Hessian and the Difference of Gaussians (DoG) scale space. SURF is optimized for faster computation and is therefore preferable for large-scale applications. Both methods define ways for locating and describing the keypoints.

One way of speeding the matching process up is to simplify the feature descriptors. For example, [7] propose binary descriptors that can be compared efficiently using the Hamming distance. Although this provides tremendous speedups for individual matching, the main challenge is the quadratic complexity as still many matching operations have to be performed.

[1]This work was supported by the European Research Council (ERC) under the project VarCity (#273940) at www.varcity.eu.

Figure 2: Search space reduction by correct matchability: The top image shows initial image features, while in the bottom only the features predicted to be matched correctly are shown. The majority of features on unreliable objects (pedestrians, car, vegetation) are omitted from the prediction, while features on buildings are preserved.

In terms of efficiently searching for matching candidates, the problem of feature matching is easily parallelizable [1], but is nevertheless very time-consuming because of its quadratic nature. The state of the art in this field employs hierarchical data structures for approximating the search. One of the most common approaches that is used by the majority of feature matching software is using KD-trees [4]. Despite the fact that bringing in efficient data structures significantly reduces the search time, due to the high dimensionality of the descriptors, search through the whole tree can still be relatively long. Therefore, the number of nodes visited during the search process is usually restricted to some fixed value, resulting in an ANN (approximate nearest neighbor) search [2]. To further reduce the computational complexity of the problem, it is common to cluster image features into so-called visual words [22]. This provides an approximation of the search space due to quantization of the possible visual descriptors.

A technique that is relevant to visual words is the vocabulary trees [18]. For building a vocabulary tree, K-means clustering is run on a set of feature descriptors, dividing them into k clusters of visual words. This procedure is repeated iteratively for every cluster which results in a hierarchical search tree capable of sublinear search time. A feature to be matched is traversed down the tree and matched only to the features found in the leaf where the feature ended up itself. Examples of such efficient searches are found in [1, 14, 20]. Recent work like [8] employs hashing to very efficiently index matching pairs.

Up to now, the described approaches were aiming to speedup or improve the matching of the full set of features. The classical heuristic for cutting out unreliable feature points is first matching all of the points and then finding the best and second best nearest neighboring feature. For these a ratio of the descriptor distances is computed. Empirical evaluations showed that keeping only points with a ratio 0.8 and lower leads to the most stable results [16, 22].

A different way of thinking of improving feature matching is to reduce the number of feature points to match – prior to matching. This is the field of research most relevant to our approach and little research has placed a focus here.

The most straightforward method to reduce feature points is to extract fewer points from the input images. The filtering here is based on the implicit feature response. However, this reduces the features based on how detectable they are not on how matchable they are – resulting in potentially non-overlapping sets of detected feature points.

There is a number of other approaches that truncate the number of features to be matched in the image. [25] proposed to constrain the matching process to features within the same scale. To determine which images are matchable, pre-matching of features of high scale is performed. If the number of matches of first h features of the highest scale is large enough, the images are considered to be matchable. This is mainly used for discarding entire images, but can be extended to image-to-image pre matching at a coarse level.

[23] compute usefulness of features which is determined by their frequency and presence in other images. A vocabulary tree approach used to connect similar images and rank its features based on geometric verification.

[5] provide a non-maximum suppression technique together with indexing of the features to quickly reduce the set and provide an outlier rejection procedure.

Recently, [13] proposed an idea to classify features points to be matchable. Similar to [23] they extract a large set of feature matches that were verified by a geometric verification process. Then they train a simple random forest classifier to determine the likelihood of a feature to be matchable. This is then seen as a prediction of matchability of every feature point and those which are not expected to have a high likelihood to match are discarded.

All of the above methods are effective for reducing the set of potential candidates. However, none of them aims to reduce the candidates to only the correctly matching ones.

Our method has two significant differences to related work. First, instead of predicting a feature point's matchability (i.e. whether it will pass the ratio and maximal distance test), we go one step further and predict if it will be correctly matched. Second, we reduce the number of point comparisons by matching only those that are semantically close they to each other. This leads to a further speedup and quality improvement.

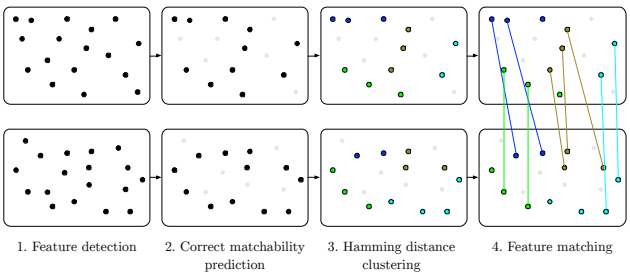


Figure 3: Suggested matching pipeline. Distances between binary histograms of the points of the same color are within a threshold. Note that the steps 2 and 3 are independent of each other and therefore can be used separately.

To the best of our knowledge, there is no other approach which employs semantic scene segmentation to feature matching. We are the first to propose a semantic context to filter out incorrect matches and predict which features will likely match correctly.

3. Method

In this section we describe the details of our proposed semantic context feature matching. It is a generic approach which can be applied to any sparse or even dense feature point extraction or matching method.

The overall outline of our method is as follows, see Figure 3. Given a set of M images, we perform interest point extraction and semantic segmentation on the images. This provides us with sets $\mathcal{F}_i, i = 1 \dots M$ of features for every image. For any pairwise matching of images i and i', the naïve method uses a quadratic matching $|\mathcal{F}_i| \times |\mathcal{F}_{i'}|$.

First, we suggest reducing this set by learning which features are likely to match correctly based on their semantic context and leaving the other features out. This reduced set of features is then used within an efficient search tree structure to provide final matching candidates.

Second, we propose a method for finding the semantic proximity of the features. Using it, one can match every feature only with the ones that are semantically close to it. Both methods can be used separately; however, they can be combined into a pipeline. In the following we will define our novel semantic contexts for feature points, efficient ways of preselecting candidates and the approach we use to learn which features will match correctly.

3.1. Semantic Context

For semantic segmentation each pixel of the input image is labeled into a semantic class, such as wall, window, door, sky, road, sidewalk, pedestrian, vegetation, etc. We follow the implementation of [19], where for each image we extract a basic 11-dimensional feature vector containing

the CIELAB Lab^* color components and 8 responses of the MR8 filter bank [24, 11]. This feature vector and the ground truth label from the training set is used to train a random forest classifier. For assigning the label based on the classifier probabilities, we use a simple maximum-a-posteriori decision without any regularization.

This gives a mid-level representation of the scene understanding of that image. Following this we construct a semantic histogram to capture the semantic context around each feature point $f_{ij}, j = 1 \dots |F_i|$ of the i-th image.

Feature scales are provided by the feature detectors. Given the feature scale s_{ij}, we define support region by a circle c_{ij} of radius r_{ij} that is proportional to s_{ij} and center coinciding with the feature coordinate. If the segmentation has B classes and $b(p)$ is a function that returns the index of a semantic class for pixel p, the semantic histogram h_{ij} for feature j and image i is defined as

$$h_{ij} = \begin{pmatrix} h_{ij}^1 & h_{ij}^2 & \dots & h_{ij}^{|B|} \end{pmatrix}, \tag{1}$$

where

$$h_{ij}^k = \frac{\#b(p) = k}{\#p \in c_{ij}}, p \in c_{ij}, k = 1 \dots |B|. \tag{2}$$

Here, each pixel p within the support region c_{ij} is sorted into a histogram bin reflecting its semantic class. Hence, for every semantic class, the corresponding bin of the histogram contains the number of pixels of this class that are inside the interest region of the pixel normalized by the total number of pixels in the circle.

For stable approximation of the histogram, we threshold it into a binary vector $\hat{h}_{ij}$ as follows:

$$\hat{h}_{ij} = \begin{pmatrix} \hat{h}_{ij}^1 & \hat{h}_{ij}^2 & \dots & \hat{h}_{ij}^{|B|} \end{pmatrix}, \tag{3}$$

where

$$\hat{h}_{ij}^k = \begin{cases} 1 & \text{if } h_{ij}^k \geq t_{\text{bin}} \\ 0 & \text{if } h_{ij}^k < t_{\text{bin}} \end{cases} \tag{4}$$

where the threshold t_{bin} is selected to reduce the effect of arbitrary noise in histogram due to sporadic pixels and their semantic classes.

3.2. Efficient Distance Indexing

Given semantic histograms for every feature in the image, we can efficiently compare them. Unlike standard feature descriptors which undergo a large variety in descriptive power, our semantic histograms capture the semantic context of a feature point. This semantic context has a limited set of semantic classes combinations, which in practice is a feasibly low number.

For two feature points, p_{ij} and $p_{i'j'}$ from images i and i' respectively, we can compute the semantic distance as:

$$d_s(p_{ij}, p_{i'j'}) = d_{\text{ham}}(\hat{h}_{ij}, \hat{h}_{i'j'}) \tag{5}$$

where $d_{\text{ham}}(a, b)$ is the Hamming distance between the two binary vectors a and b.

If two features are compared, small distance value of a semantic context between them is a sign of a good matchability. Although in the ideal case the semantic histograms of the matched features should be completely coinciding, some relaxation should be made to avoid mismatching due the noise in the semantic histograms. For that, we define features to have a potential to be matched if their semantic distance is smaller than a threshold t_{ham}.

Instead of matching every feature point of the first image to every feature in the second image (as a quadratic matching would require), we propose an indexing of the semantic contexts. For this we can create a list of precomputed feature sets that are closer than t_{ham} by their binarized semantic histograms. For an image pair i and i', we use the following algorithm to reduce the search space:

1. Compute all features and their binary histograms following Equation 3.

2. Determine the subsets of unique histograms $\hat{H}_i^*$ and $\hat{H}_{i'}^*$ for images i and i'. Merge them to obtain $\hat{H}^* = \hat{H}_i^* \cup \hat{H}_{i'}^*$.

3. For every feature f_{ij} we define a mapping $\bar{h} = \hat{h}^*(f_{ij}), \bar{h} \in \hat{H}^*$ that returns a unique binary histogram number for every feature. The number of unique histograms is typically less than 10% of the number of features in one image.

4. We further define an inverse mapping $\hat{h}^*(\bar{h}, i)^{-1} \rightarrow \{j\}, j \in 1 \dots |\mathcal{F}_i|$ that for a given binary semantic histogram $\bar{h}$ and image index i returns indexes of all features with the same unique histogram in the image.

5. For every unique histogram $\bar{h} \in \hat{H}^*$ we find a set of indexes of unique histograms that are closer than the threshold to the j^*-th histogram:

$$\hat{H}^*_{\text{match}}(\bar{h}) = \{j\} \in \hat{H}^* : d_s(\bar{h}, h) \leq t_{\text{bin}}. \tag{6}$$

6. At the matching process, for every point, we consider its unique histogram $\bar{h}$ and match it only to the points with the unique histogram indexes from $\hat{H}^*_{\text{match}}(\bar{h})$.

Such a reduction of a search space facilitates both the speed and the quality of the matching process. It takes much less time to perform the search of the semantic histograms that are close to a given one than to perform full search in the feature descriptor space. Moreover, using semantic histograms guarantees that the matched points describe the objects of the same classes, reducing the number of potential outliers. Note that instead of classifying each feature

point directly into a single semantic class, our semantic histogram description leverages the power of the support regions which provides a much richer indexing into similar semantic areas.

3.3. Learning to Predict Correct Matches

The final part of our method tries to predict correct matches. Unlike related work [5, 13, 23], which purely filters or predicts if standard matching criteria like distance ratio will be passed, we take the concept to the next level.

In contrast to the related work there are two important differences. First, instead of trying to predict if a feature will be matchable or not, we are aiming for leaving only the features that are predicted to be matched *correctly*. There are often too many false matches remaining after the distance ratio, as it only considers matches that have passed the ratio of the distance to the first to the second match test, and not correct matches. Second, instead of using the original feature descriptions as an input to the classifier, we train on the semantic histograms which provide a mid-level abstraction of the scene understanding.

Our method hence is formulated as follows. We aim to classify each feature point as being likely to be correctly matched or not. This is a binary decision for which we need ground truth training examples. Obtaining such ground truth matches is extremely time-consuming and is not guaranteed to be reliable if done by manual work.

Hence, we propose to use a very strict automatic method to collect ground truth. It is based on a series of well-tested automatic parts of an entire Structure-from-Motion (SfM) and Multi-View-Stereo (MVS) pipeline. The SfM includes reliable feature detection, robust description, nearest neighbor full brute force matching, geometric verification and triangulation for outlier rejection and final bundle adjustment of exact camera and point locations. The MVS includes dense depth estimation, robust surface reconstruction and photometric fine tuning of the surface. The resulting surface ranks among the most accurate methods as evaluated on the MVS Middlebury benchmarks [21, 15].

The accurate surface allows to re-project points on the surface to artificially create matching pairs across images. If a point is seen in more than one image, its projections to the images are considered to be matching features. For scale and orientation invariance we project spheres of fixed radius around the mesh surface and its 3D points to the 2D images. From the size of the resulting ellipse we infer point scale by taking a fixed number of equidistant points on its border, finding the largest distance for every point to another point of the subset, and averaging this value. This gives a radius of a circle containing the feature interest region.

To determine positive and negative labels, we perform a standard feature matching speeded up by using KD-trees. This is done for every point and the results are filtered using the standard first to second match distance ratio of 0.8 to determine final matches. If the original 3D surface point could be matched and the computed match for is coincides with the ground truth match from the surface projections, then it is considered a positive sample. If the ground truth for a point exists but is different from what was found, it is taken as a negative sample. If the point does not contain a ground truth match, it is not considered at all. Due to the construction of the ground truth, the fact that a point does not have a match, does not necessarily indicate that it is not matchable. However, for training and testing classifiers it can be ignored and is part of future research.

To learn our proposed correct matchability of the points, we then train a random forest classifier on the semantic histograms (as defined in Equation 1). This classifier hence learns the semantic context of a feature point and its likelihood to be matched. The intuition is that certain semantic class combinations are more likely to be matched and homogeneous semantic context is less discriminative for correct feature matching.

In the next section we analyze the accuracy and efficiency of our proposed approach. For this we also train a classifier directly on feature descriptions. Further, we evaluate the entropy over the binary semantic histograms from Equation (3) as this is a similar measure of complexity of the semantic context.

4. Experiments

The evaluation is performed in three parts with various baseline methods. First, we analyze our proposed classifier for predicting correct matchability. Second, we investigate the performance during actual matching scenarios. Finally, we report the runtime benefits of all methods.

4.1. Datasets and Implementation Details

In this work we evaluate on two multi-view reconstruction and semantic segmentation datasets in urban scenes. Both are annotated with ground truth labels, such as road, wall, window, door, street sign, balcony, door, sky, sidewalk, etc. First is CamVid [6], that is a public dataset. We use its sequence 0016E5, which contains the most buildings and frames. Note that SfM/MVS was only stable for a subset sequence of 102 of its 300 frames. We use the larger RueMonge dataset [19] showing 60 buildings in 428 images covering 700 meters along Rue Monge in Paris. The CamVid dataset is taken from a car driving forward on a road (with an average viewing angle of $40°$) while in the other in dataset the human camera man points more or less towards the buildings (avg. angle $\approx 10°$).

For completeness, we verify the methods while running on automatic semantic segmentation as described in Section 3.1 and also on ground truth semantic labels. The information about the datasets is provided in Table 1.

Dataset	#classes	#train	#test
Camvid (GT)	32	880k	1806k
Camvid	11	880k	1806k
RueMonge (GT)	8	312k	1425k
RueMonge	8	103k	490k

Table 1: Parameters of the datasets: number of semantic classes and features in disjoint training and testing. GT refers to the datasets with ground truth segmentation.

For SfM/MVS reconstruction we use the publicly available tools from [25, 15]. As feature descriptors, we use SURF descriptors [3] of length 64 (as they are proven to be reliable and fast). It is worth mentioning, however, that our method is generic and works with any kind of feature descriptions. For search structure we use KD trees [4].

For the parameters, we use $t_{\text{bin}} = 0.1$ for filtering semantic segmentation noise and small overlaps into other classes. The maximal Hamming distance is set to $t_{\text{ham}} = 1$, although we also study the case of $t_{\text{ham}} = 0$ (that means that only the features with exactly equal binary histograms are matched). The random forest is trained on 10 trees with a depth of 20.

The runtime for our preprocessing takes relatively little time and only has to be done once per image. Semantic segmentation is done in less than 2 seconds. Extraction of semantic histograms takes from 1-2 seconds per image in an unoptimized Matlab implementation.

4.2. Evaluation and Discussion

The goal is to evaluate how many correct matches can be predicted prior to performing the matching. We evaluate the proposed classifier using our semantic histograms as feature vectors. As first baseline, we use random selection that simply picks a random subset of keypoints and marks them as matchable. Second is the random forest classifier trained on feature descriptors. Third is the subset selection based on the value of semantic entropy. Every method has a confidence that determines how many points it will predict to be correctly matchable.

Measurements Suppose the i-th image has $N = |\mathcal{F}_i|$ feature points that have ground truth matches (as mentioned above, we discard the points that do not have ground truth match information). For a fixed confidence, the evaluated method returns N^+ matches that coincide with ground truth and N^- that are incorrectly matched, as illustrated in Figure 4. Hence precision of prediction is defined as

$$\text{precision} = \frac{N^+}{N^+ + N^-}, \tag{7}$$

where the sum $N = N^+ + N^- + N^\dagger$ is the total number of features. Since there are no TN matches (correctly classified false matches), the total number of points is different than

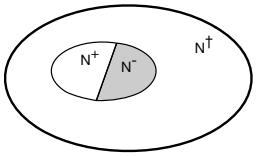


Figure 4: The points from the inner circle (N^+, N^-) were matched by the algorithm, while for the other points $(N^\dagger)$ no match was returned.

in the ground truth, and we use a slightly different definition w.r.t. standard recall. We instead define the fraction points for which the classifier has predicts matchable features as

$$\text{returned fraction} = \frac{N^+ + N^-}{N} \tag{8}$$

which behaves similar to recall, as it measures how many features are classified as matchable w.r.t. all features N.

Prediction Evaluation Figure 5a demonstrates the precision of the classifiers for the two datasets. The precision obtained by both the semantic classifier and the semantic entropy are best for all operating points. Both the classifier on feature vectors and random subsampling cannot reach this performance. For random subsampling it is clear that, considering that both images are subsampled, the chance of finding a correct match is very low, especially if there are not many points selected.

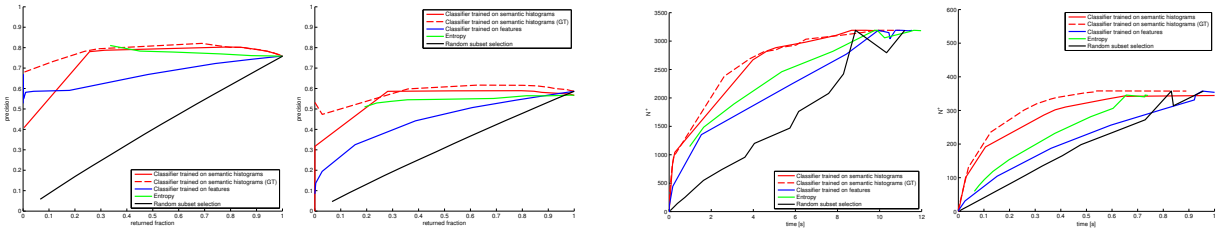
In contrast, in our approach the semantic classifier already achieves the top precision at only 30% of the initial point set. Hence, the high scoring matchability features also have a high accuracy and repeatability.

Semantic entropy is equally accurate, however it reaches its limit when the semantic histogram contains only one class. Here the minimal number of points the method can return is the number of points (around 20% to 40%) that lie entirely in one class with their area of interest.

An interesting and useful property of our robust semantic classifier prediction is the fact that it is robust to the quality of semantic segmentation. The average IOU classwise accuracy is only 42%, see Figure 8 for a qualitative example of the segmentation noise. Yet the precision when using automatic semantic segmentation is highly competitive to an oracle ground truth segmentation. This allows using very fast but imprecise methods to obtain semantic segmentation and still achieve high precision.

Figure 5b demonstrates how many features will a method return in a given time. Again our semantic histogram method is more effective: for every unit of time, it returns more correct points compared to all other methods.

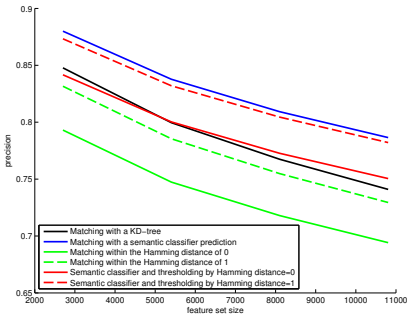
Overall the gain in speed is significant for both datasets. For example, for RueMonge our method is 5 times faster to return 200 correctly matched features (0.1 vs 0.5 seconds).



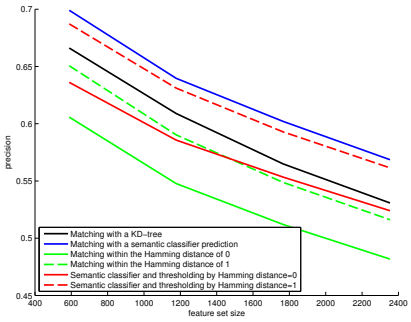
(a) CamVid / RueMonge – precision vs returned fraction

(b) Camvid / RueMonge – correct return vs time

Figure 5: Prediction Evaluation– the prediction of a correct match based on the semantic context is much higher than on the pure feature description or semantic entropy, and more correct matches can be returned in less time.



(a) Precision vs feature set size (Camvid)



(b) Precision vs feature set size (RueMonge)

Figure 6: Matching evaluation – precision of correct matches w.r.t. the baseline of KD-trees (black).

It is also interesting to note that the matching speed for the same number of points within the KD-trees are different. This effect results from the fact that well-matchable points are also distinctive which makes them unique and easier to find within the search tree.

Matching Evaluation In the second experiment, we evaluate the performance of the actual feature matching. For quantitative evaluation, we downsample the number of image features in each candidate image. To preserve the same probability that a feature will be matched, we use the ground truth matches and randomly select a fraction to keep as initial feature candidates.

Hence, the values corresponding to the rightmost value of the x-axis reflect the results of the experiment undertaken on the full feature sets without downsampling. The other values show the case when the number of detected features is lower (e.g. due to a higher keypoint quality tolerance threshold).

First, matching time and its dynamics are studied. The results are presented in Figure 7a. All the proposed methods are faster than the baseline KD-tree matching. It can be seen that the methods that involve Hamming distance thresholding by 0 are the fastest (because the search space is drastically limited).

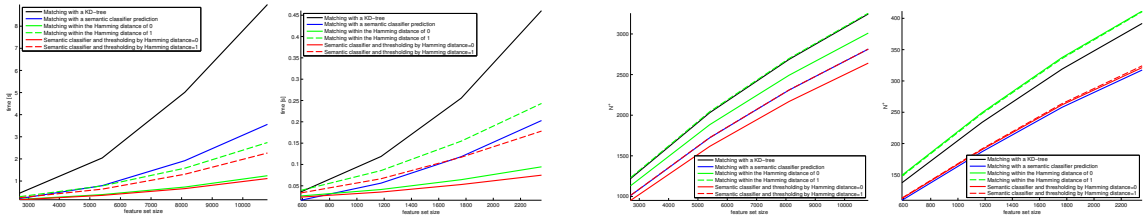
Figure 7b shows the absolute number of correct matches. For the majority of methods the overall number of returned matches that are correct is lower (about 75%) compared to the baseline methods. This is explained by the significantly lower number of points that is used for matching. However, and most importantly the precision of the results (as shown in Figure 6) is higher (+3-5%) for our method that uses semantic classifier prediction and its combination with the Hamming distance thresholding by $t_{\text{ham}} = 1$.

In summary, the best method is the combination of classifier predictions and Hamming distance thresholding by $t_{\text{ham}} = 0$. Although it has the same precision as the baseline, it provides a significant speed gain (matching time up 8x faster). To further increase precision, the combination of a predictor with the Hamming distance thresholding $t_{\text{ham}} = 0$ with still a speed gain of 6x over the KD-tree baseline.

5. Conclusions

Image-to-image feature matching is the main time bottleneck in any matching scenario. In this work we provides a novel insight into understanding what is being matched.

We proposed semantic scene understanding to improve the quality and speed of the image feature matching process. The three novel methods are describing semantic context of a feature, eliminating the search space so that only the features that are close semantically are matched and last but not least, predicting if a feature will be matched correctly prior to the actual matching.



(a) Return time vs feature set size (Camvid / RueMonge) (b) Correct return vs feature set size (Camvid / RueMonge)

Figure 7: Matching evaluation – absolute size of correct matches w.r.t. matching time provides significant speed gains.

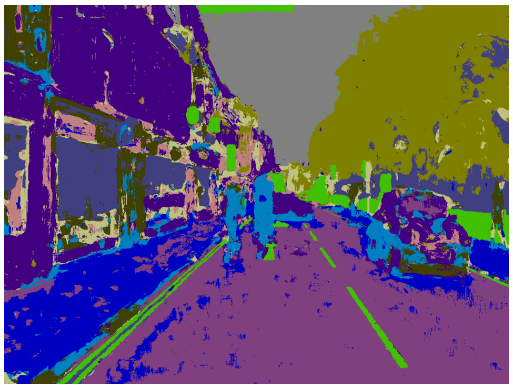


Figure 8: An example of noisy semantic segmentation

Experimental results confirm that exploiting semantic information significantly reduces the search space and results in a considerable speed gains as well as increase of the precision of the matching.

This provides an exciting benefit for any general feature matching. Future work in this field will be directed at using additional semantic information for low-level vision procedures like feature matching or video compression where also human vision has benefits from scene understanding.

References

[1] S. Agarwal and N. Snavely. Building rome in a day. *ICCV*, 2009. 1, 2

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998. 2

[3] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. *ECCV*, 2006. 1, 2, 6

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. 1, 2, 6

[5] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. *CVPR*, 2005. 3, 5

[6] G. Browstow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, 2008. 5

[7] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. *ECCV*, 2010. 2

[8] J. Cheng, C. Leng, J. Wu, H. Cui, and H. Lu. Fast and Accurate Image Matching with Cascade Hashing for 3D Reconstruction. In *CVPR*, 2014. 2

[9] J. Frahm, P. Fite-Georgel, and D. Gallup. Building Rome on a Cloudless Day. *ECCV*, 2010. 1

[10] S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *IJCV*, 94(3):335–360, 2011. 1

[11] J. Geusebroek, A. Smeulders, and J. van de Weijer. Fast Anisotropic Gauss Filtering. *TIP*, 12(8):938–943, 2003. 4

[12] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. 1

[13] W. Hartmann, M. Havlena, and K. Schindler. Predicting Matchability. *CVPR*, 2014. 3, 5

[14] M. Havlena, A. Torii, J. Knopp, and T. Pajdla. Randomized structure from motion based on atomic 3D models from camera triplets. *CVPR*, June 2009. 2

[15] M. Jancosek and T. Pajdla. Multi-View Reconstruction Preserving Weakly-Supported Surfaces. In *CVPR*, 2011. 5, 6

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1, 2, 3

[17] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 27(10):1615–1630, 2005. 1

[18] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. *CVPR*, 2006. 1, 2

[19] H. Riemenschneider, A. Bodis-Szomoru, J. Weissenberg, and L. Van Gool. Learning Where To Classify In Multi-View Semantic Segmentation. In *ECCV*, 2014. 3, 5

[20] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, 2007. 2

[21] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *CVPR*, 2006. 5

[22] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. *ICCV*, 2003. 2, 3

[23] P. Turcot and D. G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. *ICCV Workshops*, Sept. 2009. 3, 5

[24] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *IJCV*, 62(1–2):61–81, 2005. 4

[25] C. Wu. Towards linear-time incremental structure from motion. In *3DPVT*, 2013. 1, 3, 6