Visual words for 3D reconstruction and pose computation

Srikrishna Bhat K K, Marie-Odile Berger, Frédéric Sur
INRIA/LORIA/Nancy-Université, France
{bhatsrik, berger, sur}@loria.fr

Abstract

Visual vocabularies are standard tools in the object/image classification literature, and are emerging as a new tool for building point correspondences for pose estimation. This paper proposes several visual word based methods for point matching, with structure from motion and pose estimation applications in view. The three dimensional geometry of a scene is first extracted with bundle adjustment techniques based on the keypoint correspondences. These correspondences are obtained by grouping the set of all SIFT descriptors from the training images into visual words. We obtain a more accurate 3D geometry than with classical image-to-image point matching. In the second step, these visual words serve as 3D point descriptors robust to viewpoint change, and are then used for building 2D-3D correspondences for a test image, yielding the pose of the camera by solving the PnP problem. We compare several visual word formation techniques w.r.t robustness to viewpoint change between the learning and test images and discuss the required computational time.

1. Introduction

Camera Pose Estimation is a challenging problem with numerous applications[12]. In this paper, we consider an approach in which the pose of a moving camera is estimated using the location information of the 3D points previously extracted from a training video sequence. Point matching is involved in each step of the process and still presents challenging difficulties. Building the 3D model (here, an unstructured 3D point cloud, also called a 3D map) from the training sequence calls for keypoint correspondences across the views. Then, with the model at hand, estimating the pose via the Perspective-n-Points (PnP) camera pose estimation needs correspondences between the 3D points and the keypoints from the training image.

In this paper, we investigate visual word techniques for point matching, both 2D-2D and 3D-2D. While visual vocabularies are now standard tools for object or image classification and recognition [18, 20], they are still emerging techniques for Structure from Motion (SfM) and pose estimation problems. The standpoint is not the same: in the first situation visual words are used to build image signatures, whereas in the SfM problems visual words are used as local photometric descriptors to ease point-to-point matching. The standard tool for building point correspondences in SfM problem with Bundle Adjustment (BA) techniques is undoubtedly keypoint matching with e.g. SIFT [15]. This approach is used for example in Snavely's Bundler [21], or in [8], where a relatively limited set of reference images is used to build the 3D model from BA. Now, attaining a good precision in Bundle Adjustment estimation needs tracks of 2D correspondences between images spanning a large viewpoint change. This is endangered by the lack of invariance of SIFT descriptors to these changes. That is why we use visual word clustering from the set of all images of the learning sequence: the idea is that this sequence should contain smoothly varying aspects of the same objects, thus clustering should group the descriptors associated to the same 3D point into the same visual word.

1.1. Related Work

Usual image matching techniques with visual vocabularies have been recently successfully used in 3D reconstruction for multimedia application [1], location recognition and loop closure detection [2], or pose computation [6, 8, 10, 23].

Our paper shares idea with these four latter papers. In [23], visual word matching is used to define a 3D model from several views of an object through SfM techniques and manual annotation. Each point of the 3D model is then associated with the visual words from the associated 2D points, permitting to build 2D/3D correspondences from a new image (and estimate its pose), after some ad-hoc segmentation and robustification. As mentioned earlier, in [6] the SIFT correspondences are used to build a 3D model with BA. The 3D points are associated with the SIFT descriptors attached to their projection in the learning views. Then, the camera pose is estimated in a new video sequence by matching the SIFT keypoints from the test images to these sets of SIFT descriptors, which can be seen as visual words describing the 3D points. In [10], visual words are also used as descriptors of 3D points, and a clever algorithm is used to draw 2D-3D correspondences with wide viewpoint changes

(thanks to viewpoint simulation [16]) and repeated patterns. The approaches from [8] and [23] are somewhat mixed together in [6]. In this latter paper, visual word matching is indeed used in addition to local feature tracking. Local feature tracking enables to estimate pose from frame to frame, while visual words permits longer range matching and stabilizes the estimation. Submaps are built on-line and when the reprojection error becomes too high, a new submap is initiated, and linked to old submaps via visual word matching. Remark that Lepetit et al.'s paper [14] is also relevant to this work, since it considers point matching as a classification problem: points of interest are associated with artificially generated descriptors under affine or homography hypothesis. To some extent these sets of descriptors can be seen as visual words. As in our framework, the aim is to ease wide baseline matching.

Unlike existing techniques for obtaining 2D/3D correspondences, where the training images are either captured from constrained views of the environment [8] or deal only with specific objects [23, 10], we aim at achieving a similar objective from a training video sequence in a large environment. Hence, scalability and robustness of the visual words to pose variations between training and testing images are all the more relevant in our case.

1.2. Organization of the paper

Our work involves building a 3D model of the target environment from training images and using the model to compute pose in a test image. In section 2, we present our framework for using the visual word techniques to obtain 2D matches across images during training stage and to represent 3D points for testing stage. In section 2.2 we describe different visual word methods in the literature which will be used in our experiments. Section 2.3 contains the main contribution of this paper, where we present adaptive transitive closure method for computing visual words. In section 2.4 we describe the way in which we match test features with the visual words from the training stage and perform pose computation with the 2D/3D correspondences thus obtained. We separately evaluate the performance of different visual words for 3D model building and pose estimation. In section 3 we describe our data and performance measure for this purpose. The experimental results presented and discussed in section 4 show that our proposed method gives many advantages in terms of speed and quality of the 3D map. Finally, we draw conclusions based on the results and mention our plans for future work in section 5.

2. Visual word based framework for 3D model building and pose estimation

A *Visual Word* v is defined by a region R_v in *Feature Vector Space* F, whose elements are identified as representing the same visual image pattern. A feature vector f extracted from a 2D location of an image is said to belong to v if $f \in R_v$. We use visual word based framework to represent 3D points in an environment.

Let $S = \{f_1 ..... f_N\}$ be the set of SIFT descriptors extracted from the training images. A visual word formation technique partitions S based on some *similarity* measure on the features, where each partition forms a visual word. In section 2.2, we present different visual word formation techniques used in our experiments. Some of the visual words obtained from S (for e.g. visual words containing features corresponding to repeated patterns in the environment) may not be suitable to uniquely identify 3D points. By employing a pruning strategy (section 2.1) we select suitable elements from the set of visual words computed from S. After this step, a word can be considered as a set of features in physical correspondences and form a track of 2D point matches across respective training images. We then obtain Structure from Motion (SfM) by applying Bundle Adjustment algorithm (Snavely's Bundler, see [21]), which computes 3D points corresponding to some of these tracks by minimizing the reprojection error. The 2D tracks with high reprojection error will be discarded by SfM and will not produce any 3D points.

At the end of SfM, we obtain camera pose for the training images and a set of 3D points in the environment. Each such 3D point is uniquely associated with the visual word which was used as input to SfM[1]. Hence, we can detect these 3D points in a test image by matching the SIFT features to these visual words, thus providing 2D/3D correspondences for pose estimation. The process of matching a test feature vector with the visual words is explained in section 2.4. Finally, in section 2.5, we mention some of the inherent properties of each type of visual word which are relevant to the pose computation problem.

2.1. Pruning visual words

As we are interested in 3D reconstruction and pose computation, our goal is to remove, to a large extent, words which contain features of different 3D physical points. After computing the set of visual words from training video of the target environment, we prune the set to remove false correspondences as much as possible. Figure 1 shows one of the training images and the 2D locations in the image (marked by green '+') corresponding to the SIFT features belonging to the visual words containing the highest number of feature vectors. In [19], patterns occurring more than 5 times in the same image are a priori discarded, in order to handle burstiness [11] of visual elements. We also apply a similar strategy in a stricter sense. We discard the visual words which appear multiple times in a single training image. We also discard visual words with less than 4 feature

[1] SfM algorithm may create multiple 3D points for some 2D tracks. We discard those 3D points. In our experiments, the number of such 2D tracks is negligible, typically around 1% of the total number of tracks).

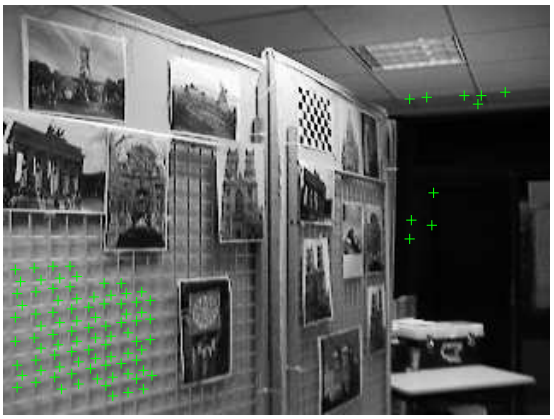


Figure 1: Sift features belonging to the three largest visual words (marked by green '+'). Clusters are computed by range based transitive closure method on TD1.

vectors, since they are likely to be unstable for reliable detection.

2.2. Types of visual words

Currently, the different ways of forming the visual words can be broadly classified into four categories, which are described in the following subsections. In our experiments, we have applied visual words from each category on SIFT features with Euclidean distance $d(x, y)$ based similarity measure.

2.2.1 Type1:k-means based clustering

In the training step, the set of feature vectors extracted from the training images is divided into a pre-determined k number of groups by clustering [20, 18]. A Voronoi region corresponding to the cluster center of a group forms a visual word. In our experiments, we apply integer k-means algorithm [22] to cluster the SIFT features extracted from training images.

2.2.2 Type2:Proximity based matching

The training features are grouped together based on the proximity to each other as in [2]. In our experiments, we apply range based transitive closure (TC) technique [3]. Two SIFT features x_1, x_2 are said to be *similar* if $d(x_1, x_2) < r_1$. This similarity relation is *reflexive* and *symmetric*. Transitive closure operation is performed on this similarity relation. Each equivalence class in S obtained in this way, represents a visual word. Hence a *TC visual word* v is represented by a set of feature vectors $\{f_{v_1}, .., f_{v_n}\}$ in the equivalence class.

2.2.3 Type3:Mean-shift (MS) based clustering

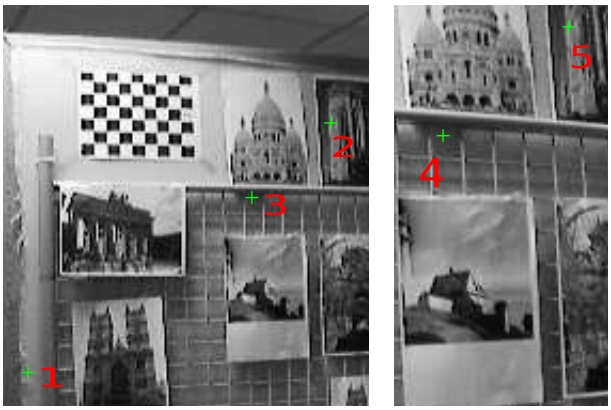
Training features are grouped based on the modes of the estimated probability density function. Iterative gradient-ascent is performed on the estimated probability function starting at each training vector [4]. The vectors which converge to the same mode are grouped into a single visual word. In our implementation, the gradient ascent iteration is stopped when the updated vector lies within a distance ϵ_1 from the previous position. After performing mean-shift iterations on all the training vectors, the modes within a distance ϵ_2 are merged and represented by their mean (as in the implementation of [7]). We use values $\epsilon_1 = .1$ and $\epsilon_2 = 1$. We experiment with two types of kernels: (1)Uniform, i.e., $K(x) = 1$ for $x \leq w$ and $K(x) = 0$ for $x > w$ and (2)Gaussian, i.e., $K(x) = e^{-(x^2/\sigma^2)}$. Uniform kernel has *compact support*, i.e., we have to explicitly provide a bandwidth parameter w, beyond which the kernel has zero value. Gaussian kernel has infinite support, but for practical reasons, we derive w, beyond which the value of the kernel is negligible[5]. For variance σ of the Gaussian kernel, our w is such that, if $x > w$, then $e^{-(x^2/\sigma^2)} < \frac{\epsilon_1}{128}$, where 128 is the dimension of SIFT features. We use kd-tree based *exact range search* (i.e. without any approximation for range search) algorithm [17], for speeding up mean-shift iterations.

2.2.4 Type4:Visual words from SfM

Instead of clustering the whole set of training features, in this method, initial matches between each pair of training images is established based on standard SIFT keypoint matching [15], that is to say nearest neighbor matching, provided the ratio between the distances to the nearest and to the second nearest neighbor is below 0.6. The matches which satisfy geometric constraints for the respective pair of images are used to compute 2D tracks. SfM is applied on these 2D tracks to obtain 3D points. The features in the 2D track of a 3D point form the visual word. We use the matching process integrated with the Bundler software [21] to establish initial matches between each pair of images. There is no pruning step involved for this type of visual word.

2.3. Adaptive TC visual words

For SfM, it is preferable to have longer 2D tracks with multiple views of the same 3D point with *significantly* varying viewpoint, in order to obtain better quality 3D model. We may have to increase the range value r_1 in range based TC method (section 2.2.2), in order to obtain matches between the features with significant pose variation. As we increase r_1 from a value a to a significantly higher value b, it will definitely retain the existing matches and may even establish new matches between elements of two or more different visual words. Consequently, those visual words will be merged together through transitive closure operation, producing visual words which are supersets of the original *sub-words* with $r_1 = a$. Depending on the characteristics of the visual pattern around 3D points or equivalently the region of the pattern in the feature vector space (different visual patterns may vary differently w.r.t. varying pose), this increment in r_1 may lead to some wrong matches.



(a) (b)

Figure 2: Range selection for matching in TC.

This problem is illustrated in figure 2. It shows portions of two training images and positions of 5 training SIFT features (marked by green '+' and numbered from 1 to 5). By closely observing the local image pattern around these points, we can see that locations 3 and 2 in image (a) match with 4 and 5 in image (b) respectively. Apparently there is no other match among these points. When we compute TC visual words with $r_1 = 100$, it combines 2 and 5 correctly, but 3 and 4 belong to different visual words. If it is increased to $r_1 = 150$, then we obtain both the matches, but the feature vector at location 1 gets wrongly merged with the visual word of the feature vector at location 2. The pruning step (section 2.1) will reject the visual word containing 1 and 2 due to its repetition in the same image. Hence, TC visual word with matching range fixed to one of the above values will definitely loose one of the matches. This can be prevented if we can adaptively compute the visual words for locations 3 and 4 with $r_1 = 150$ and for 2 and 5 with $r_1 = 100$. Essentially, what we do is to compute visual words for both the values of r_1 and perform pruning on visual words with higher range value $r_1 = 150$. If a visual word is rejected, then we look for its sub-words in the set of visual words with $r_1 = 100$ and perform pruning again. This adaptive strategy can retain both the matches shown in the image.

We extend the above strategy for multiple values of r_1 and design a recursive top down approach for obtaining visual words. We obtain TC visual words for multiple values of $r_1 = \{a_1 > a_2 ... > a_k\}$. This does not add much computational overhead because the range search for the highest value $r_1 = a_1$ retrieves all the required candidates for other lesser range values. Let $V_1, V_2 ... V_k$ be the corresponding set of visual words. The set of visual words V which finally will be used for building 2D tracks for SfM, is initialized empty. Starting with the set of visual words having highest range value i.e., V_1, we recursively select candidate visual words which are accepted by the pruning process and add them to V. If a visual word $v^i \in V_i$ is rejected while pruning due to repeated occurrence in some images, then we look for its sub-words in V_{i+1}. Pruning is applied on each sub-word. Successful sub-words are added to V and will be used for building 2D tracks. Unsuccessful ones will in turn lead to pruning of their sub-words in the next level, till we reach sub-words in V_k.

Our strategy relies on the assumption that if a matching range value r_1 is significantly high for a particular kind of visual pattern p in the training set of features, in a sense that it will match p with regions which are dissimilar to p, then the visual word containing p along with wrongly matched patterns will repeat in at least one of the training images and will be rejected by pruning process. If this assumption is valid up to the level of accuracy required for the RANSAC based 3D model building process, then we obtain the visual words whose matching range is adapted to the characteristics of the visual patterns represented by them. Even if we do not have any conclusive evidence at present, it will be interesting to investigate the practical effectiveness of this strategy in contrast to the standard SIFT keypoint matching where it is assumed that in each pair of training images, a wrong nearest neighbor match to a keypoint will have, in most of the cases, the second nearest neighbor within a ratio of the distance to the first nearest neigbhor.

2.4. Matching and pose computation on test image

During testing, we mainly perform range based matching with the elements in each visual word. A feature vector f from a test image is said to *match* with visual word v if the nearest neighbor $f_{NN} \in S$ to f belongs to v and $d(f, f_{NN}) < r_2$. We run experiments with different values for r_2. For MS visual words, we run additional experiments where matching is performed by executing the mean-shift iterations. Starting at each test vector, the same mean shift iterations as in the training step, are executed with ϵ_1 convergence criteria. A test vector f is said to match a visual word v if corresponding modes lie within a distance of ϵ_2 (the values of ϵ_1, ϵ_2 are the same as mentioned in section 2.2.3).

We obtain 2D/3D correspondences for a test image through the test feature vectors which match to visual words associated with 3D points. We use a camera of known focal length f. Our intrinsic parameter matrix is K = [f 0 0; 0 f 0; 0 0 1]. We assume that the distortion parameters are zero. We apply EPnP algorithm [13] to compute the camera pose $[R\ T]$ such that the reprojection error of the projection matrix $P = K[RT]$ is minimal on the set of 2D/3D correspondences. Let $Q = \{(U_i, u_i)\}$ be the set of pairs of associated 3D and 2D points in the test image. Due to the presence of possible erroneous matching, we use a RANSAC based pose computation algorithm. Given any set of four 2D/3D correspondences, we first compute an ini-

tial guess of the pose using 500 random 4-element subsets of Q to obtain 500 projection matrices P_j. We select the projection P_{best}, which produces highest number of inliers (points having reprojection error less than 2 pixels) in Q. Finally, we run EPnP algorithm again using only those elements of Q which are inliers with respect to P_{best}.

2.5. Visual word parameters and representation

It is worth noting, in the context of the topic of this paper, the fundamental differences between the types of visual words described above due to the inherent characteristics like computation parameters and representation. Unlike the rest, k-means based method produces pre-determined number of visual words, each with a shape of a Voronoi cell. These are disadvantages for computing 3D points because we may not be able to predict the number of 3D points in a training data a priori and the different feature descriptors of 3D points form complex shapes. But the representation is computationally efficient, since we need to store and match only one cluster center per visual word, unlike the rest where we need to store and match all the feature vectors belonging to a visual word in order to detect the corresponding 3D point in a test image. That is why k-means is so popular in the object/image classification literature (see for example [20]). Range based matching with every element in a visual word, as explained in the beginning of section 2.4, can represent visual words with complex shapes at the cost of additional space and time complexity for matching.

3. Experimental Framework

3.1. Data Description

For our experiments we capture data inside a room using a camera whose intrinsic parameters are known. We have 7 different short video sequences, out of which 6 training sequences (TD1 to TD6) are captured while manually moving the camera and the last one is a *Test* sequence captured by fixing the camera at a particular orientation on a robotic table which is instructed to move in a circular path. The radius of the circular path is approximately 42.5 centimeters. Using the robotic measurements we obtain relative positions of the camera in each test image. A rough sketch of the tracks are shown in figure 3. The camera tracks of the sequences are marked with curves in various colors. The track corresponding to each training sequence *TDi* is tagged with *TDi- -N* where $i \in \{1, 2, 3, 4, 5, 6\}$ and N is the number of images in the sequence. The arrows indicate camera orientation at subsampled positions on the track. $TD1$ and $TD4$ have maximum pose variation w.r.t the test sequence. $TD5$ encircles the test sequence, but the orientations at closest camera positions are such that there is less than 50% view overlap with the test sequence. There are three 2D surfaces in the room on which we have pasted pictures which contain texture information. Some of the pictures are repeated in order to introduce ambiguity. The distance between the center of the circular robotic trajectory and the angular meeting point of the planar surfaces is approximately 275 centimeters

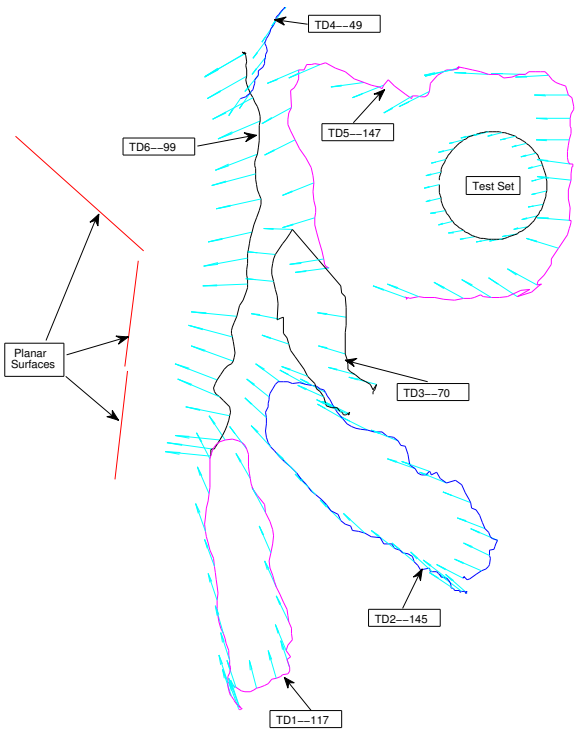


Figure 3: Camera tracks in the sequences captured for experiments.

3.2. Evaluation

We have designed an experimental framework for evaluating the *goodness* of visual words from the 3D model we obtain by applying SfM on the 2D matches established by the visual words. For $TD2$, we manually mask the two planar regions in all the images. Using these masks we identify the SIFT features extracted from the 2D locations of each surface. For each flat surface in $TD2$, we identify those 3D points which are associated with at least one SIFT feature lying on the surface. Ideally, all such 3D points should lie on a 2D surface. We compute a plane from these 3D points using RANSAC based least square fit. The goodness measure can be the root mean square distance of the 3D points from the plane. But we cannot apply this measure across different 3D models, unless they are represented in the same scale. To normalize the scale, we manually mark few 2D correspondences in the training images. We compute the 3D positions of these locations in each 3D model by triangulation. Then we manually measure the actual distances between those manually marked points in the environment. Using the ratio of the ground truth distance to the corresponding distance in the 3D model for these points, we can normalize the scale.

For evaluating the accuracy of pose estimation, we use

the ground truth 3D camera positions in the test sequence. Let $C_{gt} = \{p_1, p_2 ... p_n\}$ and $C_{est} = \{q_1, q_2 ... q_n\}$ are ground truth and estimated 3D camera positions respectively. Using the closed form solution in [9], we find the optimal transformation (translation, rotation and scale) of points in C_{est} which minimizes the sum of distances between the respective elements of C_{gt} and C_{est}. We measure the error of C_{est} as $\frac{\sum_i^n ||p_i - q_i||}{nR}$, where R is the radius of the circular trajectory of the camera positions in the test sequence.

4. Experimental Results

We extract SIFT features from the images. For training images we obtain around 200 to 250 features per image. For the test sequence we enable the option to double the size of the image during feature extraction which produces nearly 800 key points per image. We compute the different visual words for range of parameters on training feature vectors. We run k-means with $k = \{1000, 2000, ..., 9000\}$, Gaussian mean-shift with $\sigma = \{50, 60, ..., 120\}$, uniform mean-shift with $w = \{125, 150, ..., 250\}$ and range based TC with $r_1 = \{125, 150, ..., 250\}$. For our adaptive TC technique proposed in section 2.3, we use the TC visual words with range values of $\{200, 175, ..., 75\}$ in all experiments. For obtaining visual words directly through standard keypoint match based SfM, we did not need to give any parameter to Bundler software.

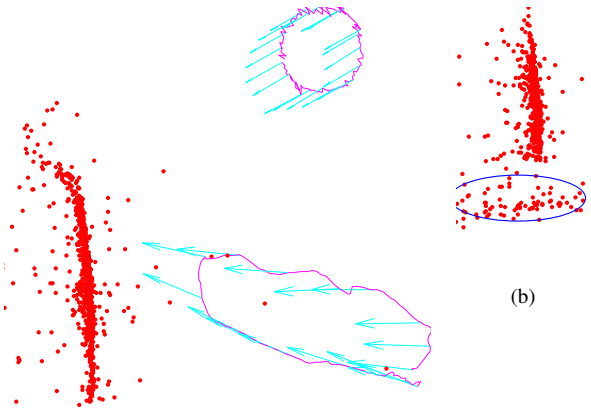
Due to lack of space, while presenting the values in tabular form in sections 4.1 and 4.2, we represent the type of visual word and the corresponding parameters as follows. We use letters G, U, T, A, B, K to represent Gaussian mean-shift, Uniform mean-shift, range based TC, adaptive TC, visual words obtained directly from Bundler and k-means based visual words respectively. '$X\ x_1/x_2$' represents visual word of type 'X' with x_1 as training parameter and x_2 as testing parameter. If there is no parameter required for training, like for B, or the parameters do not vary, like for A, then we leave it blank, for e.g. '$A\ /x_2$ stands for adaptive TC based visual word with testing parameter x_2. For MS visual words x_2 is left blank when the matching is performed thorugh mean-shift iterations with the same x_1 used for training. This will avoid confusion with the cases where x_2 is used as range search parameter.

4.1. SfM

Table 1, shows the result of evaluation of SfM process on $TD2$. The first column shows the type of visual word and the parameter. The second columns shows the % of training features which belong to visual word associated with a 3D point. Since the 3D points, during test stage, are detected through the training features, increase in this portion of the training features associated with 3D points will increase the chances of detection of 3D points. The third column shows the % of feature vectors which correspond to the 3D points lying within 10 centimeters from the best fit plane (as explained in section 3.2). They are the inliers for RANSAC based plane fit. The fourth column shows the root-mean-square (RMS) distance of these inliers from the plane in centimeters. The fifth column shows the average size of the visual words associated with the 3D points. The last column shows the total number of 3D points computed from $TD2$. From the values in the first 3 columns, it is clear that the adaptive TC method performs best closely contended by SfM based visual words from Bundler. The entries in the fifth column for these two methods indicate that the adaptive TC method produces 3D points from longer visual words which are split into multiple 3D points by the other, resulting in more number of 3D points in the last column. In a similar way, Gaussian MS performs better than uniform MS and TC visual words which are relatively more sensitive to the parameter values. Visual words based on k-means are highly unstable with respect to the k parameter, and produces least number of inliers for planar 3D points in most of the cases. Hence we ignore it in the pose estimation

Param	% Train Features in 3D	% of inliers in plane	RMS Distance of inliers	Mean Word Size	Number Of 3D Points
T 125	33.0	90.9	3.11	16.43	759
T 150	26.6	69.8	4.84	15.89	634
T 175	28.8	87.3	2.84	16.87	646
T 200	27.0	86.0	3.60	18.52	551
T 225	24.2	85.7	3.21	21.37	429
G 50	38.0	98.3	2.36	14.58	985
G 60	39.8	98.2	2.27	15.18	992
G 70	32.6	96.0	2.34	14.75	835
G 80	29.8	98.0	2.23	14.83	760
G 90	32.1	98.4	2.01	15.61	777
G 100	25.9	85.7	4.33	15.62	627
G 110	21.3	82.4	4.39	15.11	534
G 120	20.8	97.5	2.32	15.89	495
U 125	35.0	71.9	4.94	13.23	1001
U 150	38.0	89.3	3.26	13.55	1061
U 175	27.9	87.7	3.41	12.55	841
U 200	24.9	67.8	4.44	12.86	733
U 225	27.3	84.1	3.68	14.42	716
U 250	25.6	88.8	3.02	16.63	583
A	62.5	99.2	1.96	20.56	1150
B	61.6	98.6	2.35	14.81	1571
K 1000	15.9	78.7	4.68	27.67	217
K 2000	39.7	96.7	2.72	17.68	848
K 3000	48.1	71.5	5.13	12.97	1402
K 4000	46.9	50.9	5.28	10.25	1730
K 5000	56.5	86.4	3.95	8.92	2396
K 6000	51.2	74.4	4.92	7.77	2490
K 7000	50.4	76.2	5.23	7.13	2670
K 8000	51.2	82.1	4.39	6.64	2914
K 9000	41.9	54.4	5.25	6.21	2555

Table 1: Result for SfM.



(a) (b)

Figure 4: (a)Estimated circular trajectory in test and train sequence for A /200 on TD2. The arrows show the orientation at subsampled camera positions. The red dots are the 3D points computed through SfM. (b)Points in the blue ellipse belong to the ceiling.

stage.

In the above mentioned methods, excluding the SfM based visual words, mean-shift is the slowest and the TC visual word is the fastest. To mention some cases, T 200 and T 250 took around 1 minute and 1.5 minutes respectively, where as U 200 and U 250 took 62 and 117 minutes. Gaussian mean-shift is even slower and took 196 and 223 minutes for G 70 and G 90. Integer k-means took 9.3 and 16 minutes for K 2000 and K 4000 respectively. It has no meaning here to evaluate the speed of SfM based visual words because visual words are obtained at the end of Bundler.

4.2. Pose Estimation

The top view of the 3D map and the estimated test trajectory on TD2 for 'A /200' is shown in figure 4(a). There is significant view-point change between the train and test sequences. We can see that the deviation in the estimated circular trajectory is small compared to the distance between camera positions and the planar surface. The camera orientation (shown by the arrows at subsampled camera positions) in the test sequence is as expected, since the camera is moving in a circular path while keeping a fixed orientation. We have good reconstruction of the 3D points (red dots in the figure) on the planar surface. There are some scattered 3D points, most of which are from the ceiling (and hence do not belong to the planar region) as marked by blue ellipse in figure 4(b), in a different view obtained by rotating the 3D point cloud by 90 degrees in clock-wise direction around an axis in the direction from the planar region to the test camera positions.

The deviation of the estimated camera positions of the test sequence from the best fit ground truth positions (as explained in section 3.2) is given in table 2. The first column shows the type of visual word and the parameters. The various combination of parameters for training and testing, leads to hundreds of entries for performance values. Due to space limitations, we display the values only for those parameters which provide the best results for their category of visual word on at least one dataset. For some parameters we were not able to perform pose computation on the test sequence due to extremely poor quality of the 3D model built during the training stage. Red bold faced values correspond to the best result on a training data. The best result within a type of visual word for a training data is displayed in red.

The values corresponding to $TD1$ and $TD4$ which have large pose difference with the test sequence are given in the last two columns. The quality of the estimated test trajectory is very poor in these cases and consequently, the optimal transform [9] to the ground truth is not reliable. The only case of slight interest is the entry corresponding to adaptive TC method with matching range 225 for $TD4$, which is 0.32. None of the other entries are close to it.

By looking at the results corresponding to the rest, i.e., $TD2$, $TD3$, $TD5$ and $TD6$, we can say that no method is absolutely best in all the cases, but the performance of adaptive TC visual word method is the most stable across various matching parameters. It provides a result very close to the best result (with a maximum gap of .05 times the radius of the ground truth track). Stability of the performance across different training data for a particular value of the parameters is an important factor in our case because we do not have any mean to guess the best parameter value for an environment without ground truth information.

The results indicate that there is no improvement in performance of MS visual words by executing mean-shift iterations during test stage when compared to range based matching.

The poor performance of pose estimation in some of the cases on $TD2$ is due to the bad quality of the 3D model built during SfM. Even if we ignore the values on $TD2$, 'A 225' provides the second best average performance (with average deviation 0.15) in the other three cases with a gap of .01 from U 225 (with average deviation 0.14). Apart from stability of performance, adaptive TC visual words are much faster to compute than MS visual words, which makes it more suitable for training in large environments.

5. Conclusion and Future Work

In this work we have presented a framework for using visual words to represent 3D points in an environment and evaluated the performance of different visual word techniques for representing 3D points in an environment. Our experiments show that the adaptive TC visual words are better in many ways when compared to other techniques. We obtain the best or close to the best performance with an additional advantage of stability of performance across differ-

Params	TD2	TD3	TD5	TD6	TD1	TD4
T 125/225	0.13	0.10	0.13	-	1.00	1.07
T 150/225	0.85	0.13	0.11	0.24	1.04	0.93
T 175/200	0.19	0.15	0.13	0.23	0.93	1.02
T 175/225	0.12	0.12	0.13	0.23	0.96	1.08
T 225/225	0.71	0.12	0.15	0.21	1.05	0.44
T 250/225	-	-	0.17	**0.18**	-	0.50
G 50/150	0.56	0.10	0.19	-	-	-
G 80/200	0.11	0.14	0.19	0.24	0.99	0.58
G 90/200	**0.08**	0.16	0.11	0.29	0.95	1.07
G 100/200	0.79	0.14	**0.10**	0.23	0.95	1.06
G 100/225	0.79	0.13	0.10	0.25	0.86	1.09
G 110/225	0.98	0.14	0.16	0.22	0.95	0.73
U 150/225	0.10	0.10	0.18	-	1.02	0.90
U 200/	0.70	0.10	0.13	0.23	0.85	1.08
U 200/225	0.75	0.10	0.12	0.25	**0.77**	0.92
U 225/	0.21	0.12	0.12	0.19	0.86	1.06
U 250/225	0.17	0.20	0.15	0.32	0.84	0.43
A /175	0.14	0.09	0.14	0.29	0.89	0.73
A /200	0.12	0.09	0.15	0.28	1.15	0.68
A /225	0.13	**0.09**	0.14	0.23	1.04	**0.32**
B /150	0.67	0.10	0.12	0.33	1.00	1.00
B /175	0.50	0.09	0.12	0.31	1.05	1.10
B /200	0.42	0.10	0.12	0.29	0.86	1.07
B /225	0.37	0.11	0.12	0.37	0.91	1.00

Table 2: Result of pose estimation for each training set.

ent parameters and relatively lesser time required during the training stage. One of the drawbacks of our 3D point representation is the space complexity. Multiple feature vectors need to be stored for representing a single 3D point. We need to come up with techniques for approximate matching during training and compact representation for the 3D points during testing, in order for our approach to be efficient in large environments. We were not able to obtain reasonably good results for huge view-point variation between train and test sequences (i.e., for TD1 and TD4). We want to investigate whether ASIFT based view-point simulation [16] can help to obtain better results for such cases. Besides, at present, our pose estimation algorithm treats the reprojection error on each detected 3D points with equal weight. We want to investigate how we can improve pose accuracy. Incorporating reconstruction uncertainty on the 3D maps could certainly help us to only consider relevant 3D points in pose computation.

References

[1] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building rome in a day. *ICCV - 2009*. 1

[2] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. Fast and incremental method for loop-closure detection using bags of visual words. In *IEEE Transactions on Robotics 2008*. 1, 3

[3] S. Bhat, M. Berger, G. Simon, and F. Sur. Transitive closure based visual words for point matching in video sequence. *ICPR 2010*. 3

[4] Y. Cheng. Mean shift, mode seeking, and clustering. *PAMI 1995*. 3

[5] D. Comaniciu, P. Meer, and S. Member. Mean shift: A robust approach toward feature space analysis. *PAMI 2002*. 3

[6] C. Engels, F. Fraundorfer, and D. Nistér. Integration of tracked and recognized features for locally and globally robust structure from motion. In *VISAPP International Workshop on Robotic Perception*. Springer, 2008. 1, 2

[7] B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: a texture classification example. *ICCV 2003*. 3

[8] I. Gordon and D. G. Lowe. Scene modeling, recognition and tracking with invariant image features. In *ISMAR 2004*. 1, 2

[9] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A 1987*. Matlab code available at www.ti.inf.ethz.ch/ew/courses/GCMB07/index.html#CourseMaterial. 6, 7

[10] E. Hsiao, A. Collet, and M. Hebert. Making specific features less discriminative to improve point-based 3d object recognition. *CVPR 2010*. 1, 2

[11] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. *CVPR 2009*. 2

[12] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, pages 1–89, 2005. 1

[13] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An Accurate $O(n)$ Solution to the PnP Problem. *IJCV 2009*. Code available at http://cvlab.epfl.ch/software/EPnP/. 4

[14] V. Lepetit, J. Pilet, and P. Fua. Point matching as a classification problem for fast and robust object pose estimation. In *CVPR 2004*. 2

[15] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision*, pages 1150–1157, 1999. 1, 3

[16] J.-M. Morel and G. Yu. ASIFT: A new framework for fully affine invariant image comparison. *SIAM J. Img. Sci. 2009*. 1, 8

[17] D. M. Mount and S. Arya. ANN: A library for approximate nearest neighbor searching. http://www.cs.umd.edu/~mount/ANN. 3

[18] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. *CVPR 2006*. 1, 3

[19] F. Schaffalitzky and A. Zisserman. Automated location matching in movies. *CVIU 2003*. 2

[20] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV 2003*. 1, 3, 5

[21] N. Snavely. http://phototour.cs.washington.edu/bundler/. 1, 2, 3

[22] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/. 3

[23] J. Xiao, J. Chen, D.-Y. Yeung, and L. Quan. Structuring visual words in 3d for arbitrary-view object localization. In *ECCV 2008*. 1, 2