VISUAL WORDS FOR 3D RECONSTRUCTION AND POSE COMPUTATION

Bhat S., Berger M.O., Simon G., Sur F., - INRIA/LORIA/Nancy-Université, France
{bhatsrik, berger, gsimon, sur}@loria.fr

ICVSS 2010 International Computer Vision Summer School

Abstract

We present Transitive Closure based visual word formation technique for

1. Extracting the three dimensional geometry from a training video sequence
2. Using it to estimate the pose of a camera in a test video sequence

Our framework permits to match points in spite of a large baseline, resulting in a denser 3D map in (1) and in a more accurate estimation in (2), compared to the standard keypoint matching. This also gives a handy way of getting rid of the perceptual aliasing problem and of non-informative image points.

Overview of the Pose Estimation Process

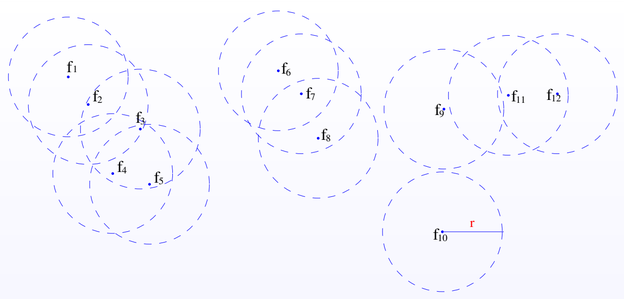
Training Phase

1. Set of Visual Words V is computed from the set of SIFT[7] features $S = \{f_1.....f_N\}$ extracted from the training image sequence and pruning V to retain only reliable words. We will use *Range-Reducing tree* structure[1] for speeding up the visual word computation process.
2. 3D map of the scene is built through Structure from Motion (SfM)[2] using the point-to-point matches established through visual words in V. We will obtain a set P of 3D points of the scene, each element of which will be represented by the set of SIFT features in the corresponding visual word in V.

Test Phase

1. Detecting the 3D points in each test image using the visual words.
2. Computing the pose using the 2D/3D correspondence.

(1) Visual word formation using transitive closure



Definition: Two SIFT features x_1, x_2 are said to be *similar* if $d(x_1, x_2) < r$. This similarity relation is *reflexive* and *symmetric*. We perform transitive closure operation on this similarity relation on S. Each equivalence class in S obtained in this way, represents a visual word. Hence our visual word v is represented by a set of feature vectors $\{f_{v_1}, .., f_{v_n}\}$ instead of a single feature vector. A vector f is said to *match* with v if there exists at least one $f_{v_k} \in v$ such that $d(f, f_{v_k}) < r$.

For the feature vectors in the figure, the final set of visual words V will be $\{ \{f_1, f_2, f_3, f_4, f_5\}, \{f_6, f_7, f_8\}, \{f_9, f_{11}, f_{12}\}, \{f_{10}\} \}$. Each visual word $w \in V$ will be the union of the spherical regions or radius r centered at the elements of w.

(2) Computation

Computation: V is initially empty. Eventually it will contain mutually exclusive subsets of S, each one of which represents one visual word. The algorithm will loop over each element of S. In each iteration $i = 1..N$ it will execute following 3 steps :

1. Find the words in V which have at least one element f such that $d(f, f_i) < r$
2. If any such words found in V, then merge all those sets together by union operation to form a single visual word and add vector f_i to it.
3. If no such set is found in V then a new element $\{f_i\}$ is added to V.

In i^{th} loop, the above algorithm needs to compare f_i with $i-1$ vectors in step(1). Hence, for N vectors in S it will need $\frac{N(N-1)}{2}$ number of comparisons. To reduce the number of comparisons, we incrementally build *range reducing tree* structure.

(3) Data for Experiments

For our experiments we have captured two videos inside a room, *Video1* for training and *Video2* for testing. Video1 contains 400 images captured through hand held camera. Video2 is captured by fixing the camera on a robotic base and moving the base through control instructions in a circular path for one full round. The orientation of the camera is fixed while capturing Video2. Video2 contains 125 images. All the images are of size 320 (width) by 240 (height). We have used the value $r = 125$ for performing transitive closure operation on Video1. While matching the features in Video2 with the visual words obtained from training images, we have used distance threshold 150.

(4)Pruning Visual Words

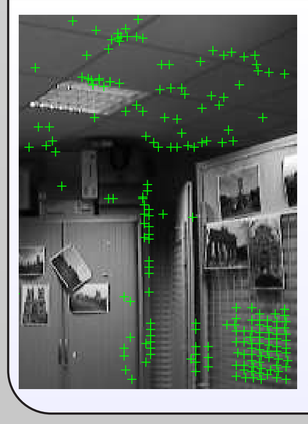


Figure shows one of the training images and the 2D locations in the image (marked by green '+') corresponding to the SIFT features belonging to the visual word containing the highest number of feature vectors. We can see that they belong to the patterns repeatedly occurring in the environment.

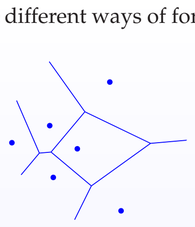
In order to prevent such ambiguous matches, we discard the visual words which appear in more than 70% of the training images. We remove the visual words with less than 6 feature vectors since they are likely to be unstable for reliable detection. In order to ensure one-to-one correspondence between V and P, we discard the visual words containing multiple feature vectors from a single image and remove the words which lead to multiple 3D points after SfM.

References

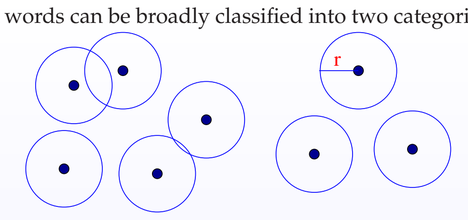
[1] Bhat S., Berger M.O., Simon G., Sur F., Transitive Closure based visual words for point matching in video sequence, in *ICPR*, 2010
[2] Snavely N., `http://phototour.cs.washington.edu/bundler/`
[3] Sivic J., Zisserman A., Video Google: A Text Retrieval Approach to Object Matching in Videos, in *ICCV*, 2003
[4] Nister D., Stewenius H., Scalable Recognition with a Vocabulary Tree, in *CVPR*, 2006
[5] Angeli A., Filliat D., Doncieux S., Meyer A.J., Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words, in *IEEE Transactions on Robotics*, 2008
[6] Pavel Z., Giuseppe A., Vlastislav D., Michal B., Similarity Search: The Metric Space Approach, *Advances in Database Systems*, 2005
[7] Lowe D.G., Object Recognition from Local Scale-Invariant Features, in *ICCV*, 1999
[8] DeMenthon D.F., Davis L.S., Model-Based Object Pose in 25 Lines of Code, in *IJCV*, 1995

Visual Words Type1 and Type2

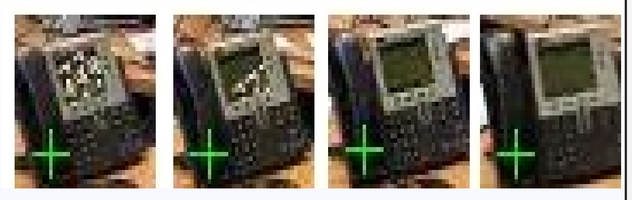
Currently, the different ways of forming the visual words can be broadly classified into two categories:



Type1 [3, 4]: The set of feature vectors extracted from the training images is divided into a pre-determined k number of groups by clustering. Each cluster forms a visual word represented by the cluster center. While testing, a feature vector is categorized to the word corresponding to the closest cluster center.
Problems:
(i) Determining suitable value for k
(ii) Cannot reject a feature from unseen object.

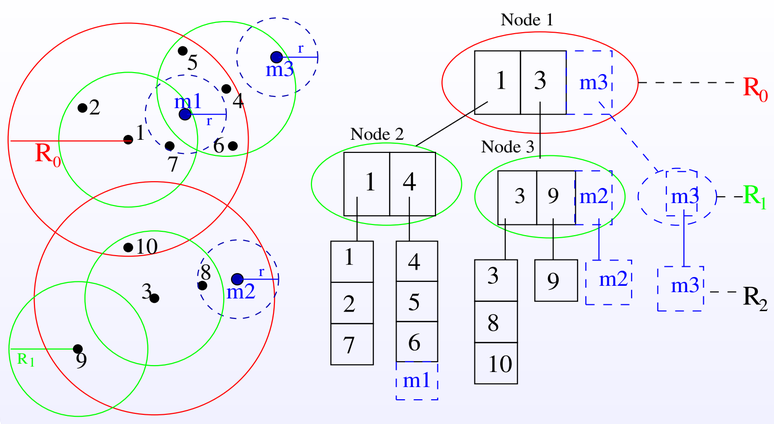


Type2 [5]: A visual word is defined as a spherical region of radius r around a feature vector. For each SIFT feature f, the dictionary is searched for a visual word centered within a distance of r from f. If f does not match any of the existing words, then a new visual word centered at f is added to the dictionary.
Problem:
Here, we can have only spherical shaped visual words. This creates a problem when feature descriptors are extracted from smoothly varying multiple views of an object as shown at the right.



The '+' mark in green color in each image shows the location at which SIFT feature vectors x_1, x_2, x_3, x_4 were detected. We have observed that the Euclidean distance $d(x_i, x_{i+1}) < 125$ for $i \in \{1, 2, 3\}$, but the distance $d(x_1, x_4) > 250$. Hence for any value $r < 250$, Type2 methods will categorize x_1 and x_4 into two separate visual words. But, for $r > 200$ we have obtained many false matches between SIFT vectors. To address this problem we propose a different way for visual word formation based on transitive closure operation [1].

Range-Reducing tree structure for Step(1) of the Visual Word computation algorithm



Range reducing tree structure is similar to *M-tree*[6]. But, in addition we have structural constraints that the tree should have a fixed number of levels L and fixed *covering radius* or *range* R_l at each level l. Figure shows how new vectors $m1, m2$ and $m3$ (blue dots with a dotted blue circle of radius r around them) will be incrementally added to a tree structure which already contains 10 vectors (black dots numbered 1 to 10). The tree has 3 levels with range R_0 (red circle), R_1 (green circle) and $R_2 = 0$. Similar to M-Trees, each node at level $l = 1, .., L-1$, contains a list of centers in which each center c has a link to a sub-tree T_c such that all the centers $c' \in T_c$ satisfy the condition $d(c, c') < R_l$. The radius at the leaf level R_L is 0.

During the search f_i will be added to the node T_{opt} corresponding to the closest center c_{opt} at level l_{opt} beyond which the *range condition* $d(c_j, f_i) \leq R_l$ fails. In figure, for $m2$, the condition fails in $Node3$ corresponding to center 3 in the rood node at level 0. Hence it is added to $Node3$. While adding a new feature vector to a node at level l, a subtree with a single child at each level till the leaf is added so that the tree structure remains consistent. Since the range condition $d(c, f_i) \leq R_l$ is stricter than the match condition $d(c, f_i) \leq R_l + r$, we do not need to search any additional nodes for adding f_i.

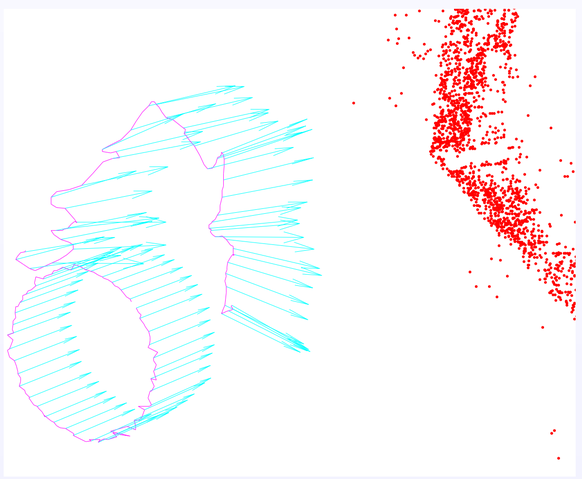
(5) Results

(i) Matches between two training Images

Each '+' mark in green color in the figure on the left shows the location of the SIFT feature in each image. Locations identified by same number correspond to the SIFT features belonging to the same visual word. We can see that our visual words can establish correspondences between two images with huge variation in respective camera poses. We can also observe that the two 2D locations numbered 6 geometrically belong to two different 3D locations. The local image pattern around those 2D points belong to the same visual word because they are extracted from two duplicate photographs of the same building. Hence we still need a RANSAC step in the 3D map building process in order to discard such ambiguous matches which occur due to similar image patterns in non-overlapping views of the scene.

(ii) 3D point detection in a test image

In the training phase, after the SfM step of our algorithm, we obtain a set of 3D points of the scene which is in one-to-one correspondence with the visual words in V. The Figure on the left shows the 2D locations of the common visual words (or equivalently 3D points) in a test image (left portion of the figure) and one of the training images (right portion of the figure). We can see that the matches we obtain for a test image are reliable. On average we obtain 80 2D/3D correspondences for a test image.



(iii) SfM and Pose Estimation

In the figure on the left, the red dots show the 3D points obtained (total 2811 points) from the training images. The pink colored contours show the camera trajectory. The circular shaped trajectory corresponds to the test video (Video2) and the other corresponds to the train video (Video1). The blue arrows at sub sampled camera positions show the direction of the camera orientation at those positions. The camera orientation for Video2, as shown in the figure is expected, since the camera is moving in a circular path while keeping a fixed orientation. Even though there are deviations in the circular path, they are very small when compared to the distance of the 3D points to the camera positions.

(iv) Augmentation

Using the pose information computed from the training and test image sequences, we have augmented two virtual objects on the two physical planes of the environment. The pose information for training sequence is obtained from Bundler while building the 3D map. For test sequence, we run a RANSAC based algorithm (POSIT[8] followed by an iterative refinement of the reprojection error) to compute pose information. Figure below shows the results of augmenting a virtual flower vase and a virtual cupboard at the same 3D coordinates in training and test sequences respectively. The left most figure belongs to the train video and the other three belong to test video. We can see that our pose estimation technique on test sequence provides good results for realistic augmentation.