J. Vis. Commun. Image R. 48 (2017) 30–42

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

The use of IMUs for video object retrieval in lightweight devices ☆

László Czúni *, Metwally Rashad

University of Pannonia, Egyetem u. 10, Veszprém, Hungary

ARTICLE INFO

Article history:
Received 1 September 2016
Revised 23 May 2017
Accepted 5 June 2017
Available online 7 June 2017

Keywords:
Video object retrieval
View-centered retrieval
IMU
Camera sensor
Image recognition
Tracking
KD-Tree indexing

ABSTRACT

We introduce a new object retrieval approach where besides cameras, Inertial Measurement Unit (IMU) sensors are used for the retrieval of 3D objects. Contrary to computationally intensive deep learning recognition and retrieval solutions we focus on lightweight methods which could be utilized in handheld devices and autonomous systems equipped with moderate computing power and memory. We use fast and robust compact image descriptors and the relative orientation of the camera to build multi-view-centered retrieval object models. As for retrieval the Hough transformation paradigm is used to evaluate the results of queries applied on several frames of a video. We analyze the performance of our lightweight approach on several test datasets and with different comparisons, including automatic tracking for the generation of queries. These experiments show the advantages of our proposed techniques since retrieval rate could be significantly increased.

1. Introduction

While optical information is crucial for the recognition of objects in the real world, other information, such as depth and audio data (e.g. [30,29]) are also often utilized. Besides the improving image quality of low-cost cameras, new sensors appeared in the last few years in handheld devices: IMUs are capable to estimate the orientation, position, and motion of cameras or other mobile devices. In our paper we show our attempts to utilize these sensors to help the retrieval and recognition of 3D objects. We build up multi-view object models composed of 2D images with additional orientation information. Thus instead of real 3D point and texture data we store, in the object models, only image information and orientation data from several view points (see Fig. 1) and during the retrieval process we fit the measurement data to the possible candidate models to answer the queries.

There are several psychophysical supports for two-dimensional view interpolation theory for object recognition in the human visual system. In [18] it is suggested that the human visual system can be described by recognizing 3D objects by 2D view interpolation. In [19] viewpoint aftereffects also prove that object-selective neurons can be tuned to specific viewing angles in the human visual system. The spatial properties of objects were always considered as significant information in the retrieval and representation of images [27]. View-centered recognition methods can be considered as early machine vision attempts for the recognition of 3D objects. The idea of storing only a limited number of views of 3D objects and then applying transformations to find correspondence with other views already appear for example in [20] where novel views are generated by the linear combination of stored ones. Rigid objects with smooth surfaces and articulated objects could also be represented this way.

Handheld/mobile 3D object recognition is a difficult task due to changing viewpoints, varying 3D to 2D projections, possible different noises (e.g. motion blur, color distortion, and thermal noise under weak illumination), and the limited computational performance and memory capacities. Local feature descriptors (like SIFT, FAST, etc.) are often used for view-centered recognition. In [6] the underlying topological structure of an image dataset was generated as a neighborhood graph of features. Motion continuity in the query video was exploited to demonstrate that the results obtained using a video sequence are more robust than using a single image.

It is obvious that video gives much more visual information about 3D objects than simply 2D projections. Not only the different views of the objects can be recorded but the 3D structure can be reconstructed by direct [31] or indirect [32] structure from motion techniques. However, these later approaches generally require high quality images and camera calibration with relatively large computational power still far from most of the mobile computing platforms and intelligent sensor motes.

☆ This paper has been recommended for acceptance by Zicheng Liu.

* Corresponding author.
E-mail address: czuni@almos.vein.hu (L. Czúni).
URL: http://virt.uni-pannon.hu/czuni (L. Czúni).

http://dx.doi.org/10.1016/j.jvcir.2017.06.001
1047-3203/

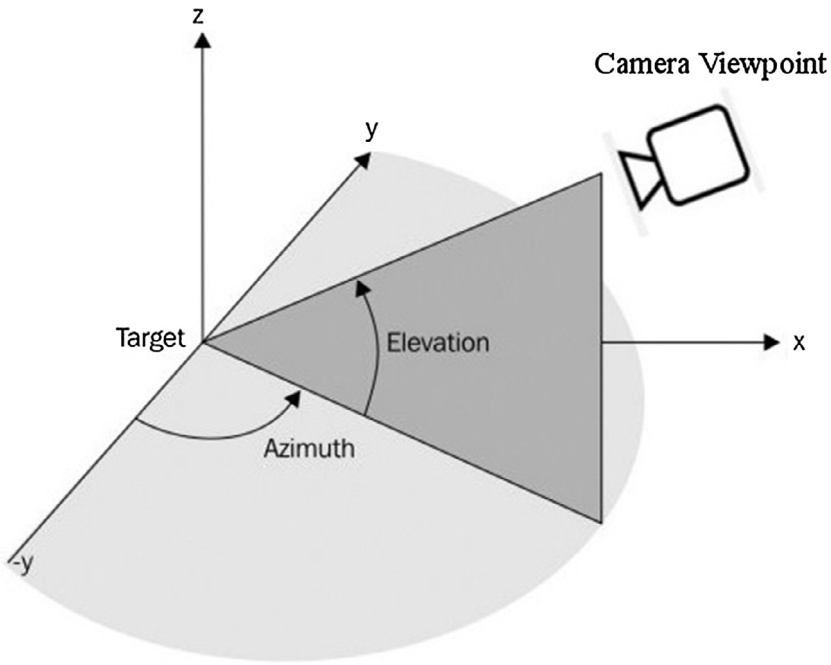

Fig. 1. Model generation setup with target object in the centre.

Luckily mobile computing devices often contain inertial measurements units (IMUs) and the calibration of cameras can be combined with IMUs [23]. However, it is still an open question how to exploit the IMUs in video retrieval and recognition omitting camera calibration and without going through the structure from motion reconstruction methodology. Our research is focused on a view-centered model where information about the relative position of the target object and the camera is exploited. Preliminary experiments already showed (see [3,4]) that IMUs can help in the retrieval process with low computational demands. However, the problem caused by low quality query images, fast object tracking and/or segmentation still can be a problem in this framework being also a subject for research.

The main contributions of our paper are showing how efficiently IMUs can help the retrieval and how efficient lightweight methods can be in case of limited number of 3D objects. It is also an advantage of our approach that it does not require any knowledge of camera calibration or viewpoint while generating the model. Moreover, the proposed method is efficient if the quality of the queries is low, it has a built in mechanism to amend, based on IMU data, possible missing visual information by the insertion of candidates to the evaluation set.

Contrary, it is not the purpose of our paper to find the most appropriate visual feature extractor and descriptor. While we apply the Color and Edge Directivity Descriptor (CEDD) [1,2] as an efficient, fast and low dimensional descriptor our proposed model also could use other popular descriptors (such as SIFT, FAST, GHOST, etc.). To prove the efficiency of our proposal several testbeds were used and in order to obtain realistic test scenarios; significant motion blur and heavy Gaussian noise was applied on the query images. Besides these simulations we carried out real-life tests where the queries were generated with the help of a semi-automatic process: first the target object was marked by the user then, in the consecutive frames, a mean-shift based tracking algorithm was tracing the object and generating the further query images.

2. Previous works

Due to the limitation of the article we can't give a comprehensive review of this ever improving area but focus on papers with similar aims and solutions (recognition of 3D objects by moving cameras).

In an early paper of [14] recognition was achieved from video sequences by employing a multiple hypothesis approach. Appearance similarity, and pose transition smoothness constraints were used to estimate the probability of the measurement being generated from a certain model hypothesis at each time instant. A smooth gradient direction feature was used to represent the appearance of objects while the pose was modeled as a von Mises-Fisher distribution. Recognition was achieved by choosing the hypothesis set that has accumulated the maximum evidence at the end of the sequence. Unfortunately, the testing of the method was carried out on four objects only.

In [15] in addition to the camera they used the accelerometer and the magnetic sensor to recognize the landscape. Clustered SURF (Speeded Up Robust Features) features were quantized using a vocabulary of visual words, learnt by k-means clustering. For tracking objects the FAST corner detector was combined with sensor tracking. Because of the small storage capacity of the mobile device a server-side service was needed to store the large number of images.

In [5] authors created object models, for video object recognition, with the help of SIFT points gathered from images taken by rotating around the object. Feature points were tracked from frame to frame and video matching was achieved by the comparison of every view of the query with all components of the optimized models of candidates. While the accuracy was about 80% in case of 25 objects, the complexity was too high to be implemented on mobile platforms. Also no explicit technique was utilized to discover the intrinsic structure of sequentially recorded query images and IMUs were not used.

In [6] also SIFT points were used as visual features. The underlying topological structure of an image dataset was generated as a

neighborhood graph of features. Graph pruning was used to get simplified structures and motion continuity in the query video was exploited to demonstrate that the results, obtained using a video sequence, are more robust than using a single image. The ratio of correct retrieval increased to 80% with the method from only 20% of single image queries in case of 100 objects while the complexity was not discussed. Besides using computationally intensive feature extraction only visual sensors were used in the recognition process.

Recently used multilayer deep learning recognition approaches discover intricate structure in large data sets by using the back propagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation of the previous layer. While there are such successful techniques for object recognition in large databases [21,22], these techniques require tremendous performance regarding processing power and memory, far from the capabilities of autonomous and mobile devices.

In [3,4] we introduced a novel visual retrieval mechanism involving the camera's orientation sensors. Now, we improve the sensor-fusion model with a candidate voting mechanism and with tree indexing resulting in the significant increase of hit-rate at relatively fast operation. Also by the extension of the test dataset we could give better analysis and comparisons with the results of others.

3. Retrieval with the fusion of optical and IMU sensors

3D objects can be represented using either object-centered or view-centered models. Object-centered representations use the features of the objects, like boundary curves, 3D points, surfaces to define the volumes of space or the texture of the boundary surfaces. Contrary, view-centered representations model the outlook of objects as captured from different viewpoints. Since objects can look very differently from different viewing directions image feature descriptors, the storing database, the search mechanism, and the feature similarity measure should be carefully designed to minimize the amount of data space, retrieval time and to maximize the hit-rate of retrieval. Fig. 1 illustrates the view-centered approach where the object of interest is in the focus of the camera. To get a complete object model a larger number of different Azimuth and Elevation angles are required. However, for average applications the elevation can be limited (in our test we used only one elevation angle typical for an object placed on the table).

3.1. Feature extraction

There are three main aspects of choosing the right features for a specific image retrieval task: to carry enough information to distinguish images; to be invariant to possible distortions; to be subject of fast and robust comparisons. Previously (see [10,16]) we investigated different types of descriptors in real-life circumstances: MPEG-7 based methods (MPEG7 CLD, MPEG7 EHD, MPEG7 SCD, MPEG7 Fusion); local feature based methods (SURF, SURFVW [1], SIFT); Compact Composite Descriptors [1,2] (Compact CEDD, CEDD, Compact FCTH, FCTH, JCD, CCD Fusion, Compact VW); and others (Tamura texture descriptor, Color Correlogram and Correlation (ACCC) [11], MPEG7-CCD Fusion [2]). We found two main effects that could seriously degrade the performance of image descriptors in real-life conditions: the change in appearance of colours under different lighting conditions and colour balance settings, and the loss of contrast due to motion blur typically occurring when the image is taken under low lighting conditions with a handheld camera. Detailed descriptions and results of those tests are available in [10,16]. Contrary to the popularity of SIFT (and similar descriptors in its family such as SURF) in image retrieval we found serious drawbacks such as running time, touchiness to blur and high dimensionality. CEDD was found one of the most robust, fast and compact among those. In [10] and in [16] it is showed that CEDD is quite tolerant for different noises and can be computed fast in today's mobile platforms. CEDD [1] is a block-based approach where each image block is classified into one of 6 texture classes (non-edge, vertical, horizontal, 45-degree diagonal, 135-degree diagonal, and nondirectional edges) with the help of MPEG7 EHD (Edge Histogram Descriptor). Then for each texture class a 24 bin color histogram is generated where each bin represents colors obtained by the division of the HSV color space. The values of the generated histogram of length 6x24 are then normalized and quantized to 8 bits. Besides its robustness and compactness, however, we should note that there are two disadvantages of CEDD compared to some other popular local point descriptors:

- in its original form it is not rotation invariant;
- as it is a global descriptor it needs proper area selection for the target.

The first issue can be handled with a proper similarity measure (see Section 3.2), the second can be fulfilled with manual or automatic segmentation methods. While in our recent application and tests a bounding rectangle around the target was designated manually (or rather the camera was moved to have the object within a bounding box), a good choice for an automatic method could be the Grabcut algorithm known from [17]. It is clear that there are always newer and better global and local descriptors (for an overview see [12]), the selection of the most appropriate one is out of focus of this paper. Our idea for sensor fusion and for the search mechanism could be implemented with other feature descriptors.

Besides optical information we propose to use the orientation information of the camera. Since the position of the target object is not fixed (not only buildings or statues are being recognized) relative orientation (degree) information between the object views is to be estimated. The accuracy and precision of todays' IMU sensors allow their application on many fields (for an overview of these see [28]) including our framework. Fig. 2 shows the distribution of absolute orientation error measured in 6 cycles (8 positions checked in each) with our tablet (see specification in Section 4.5). While in most of our experiments we used the same device, in case of experiments with the ALOI dataset we used these statistics to simulate IMU noise (see Section 4).

3.2. Image difference estimation

The difference of CEDD descriptors is efficiently given with the help of the Tanimoto Coefficient [1]. Let q_i be the descriptor of the ith view from the query and c_j be the descriptor of the jth view of a candidate. The Tanimoto Coefficient is then:

$$TC(q_i, c_j) = \frac{q_i^T c_j}{q_i^T q_i + c_j^T c_j - q_i^T c_j} \quad (1)$$

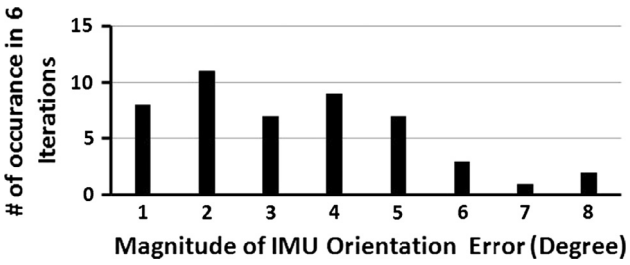

Fig. 2. IMU measurement error distribution.

where q_i^T is the transpose vector of the descriptor q_i. In case of absolute congruence of the vectors, the Tanimoto Coefficient takes the value 1, while in case of maximum deviation the coefficient tends to 0. The difference of CEDD vectors is:

$$T(q_i, c_j) = 1 - TC(q_i, c_j) \tag{2}$$

We need a modified Tanimoto coefficient to achieve rough rotation invariance:

$$TC^R(q_i, c_j) = \max_{roll} TC(q_{i,roll}, c_j) \tag{3}$$

where $roll \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$ and $q_{i,roll}$ means that orientation specific texture class positions are shifted within the CEDD vector of the query. (Please note, that the extraction of CEDD descriptor values should not be changed, only comparisons take more time to fit the actual candidate c_j best).

Please note, that since objects have different appearances from different directions, they will be represented with several frames and the corresponding descriptors. These will be denoted such $c_{j,f}$ (that is descriptor of frame f of object j). Reasonably for queries we use only one index to denote the view.

3.3. Object retrieval concepts

In this section, before introducing our proposed retrieval method, we describe different retrieval mechanisms: for the ease of understanding we start with simple extensive search of only visual data. Our purpose is to give a comparison between the concept of pure image retrieval and our new sensor fusion approach and explain our motivation for the proposed voting approach. We give the complexity of the concepts (not including special indexing techniques) based on the core complexity of comparing the descriptors of two images. Since the whole retrieval processes contain other time consuming steps the final complexity and the applicability can be checked by testing the implementations (which is given in Section 4.5).

3.3.1. EIS (Extensive image search)

In this case only visual CEDD descriptors are used for retrieval: N_f^q frames are taken from the video as query frames to compute the average distance resulting in complexity $O(N_c * N_f^q * N_f^c)$, where N_c is the number of candidate objects, and N_f^c is the number of frames in candidates. The distance function between query q and candidate object c_i is:

$$T^{EIS}(q, c_i) = \frac{\sum_{k=1}^{N_f^q} \min_f T(q_k, c_{i,f})}{N_f^q}. \tag{4}$$

Note that we did not utilize the *spatial relation* of query images, they were handled independently. See Fig. 3 for the illustration of the approach, where the blurred query images are compared to all views of candidates.

3.3.2. SIIS (Selected image and IMU search)

In this approach we show that testing only one (selected) frame from the query against all model frames then using the known relative orientation for the evaluation of other query frames results in much lower complexity without sacrificing retrieval rate. That is we define the following distance function between a query q and candidate c_i:

$$T^{SIIS}(q, c_i) = \frac{\min_f T(q_k, c_{i,f}) + \sum_{\forall l, l \neq k} T(q_l, c_{i,\triangle\alpha(k)})}{N_f^q} \tag{5}$$

where $c_{i,\triangle\alpha(k)}$ means the candidate model frame from object i which has the same (or very close) relative orientation difference $\triangle\alpha$ from $arg\ min_f T(q_k, c_{i,f})$ as frame l from frame k in the query. That is first we find the best matching frame of a selected query frame in candidates based on visual information then compare the visual descriptor of other c_i frames found at the same (or closest) orientation positions. The selection of the query frame, used for extensive visual search, can be based on its quality, information content, or time order; in evaluation experiments we used a randomized selection. The complexity of this approach can be described as: $O(N_c * (N_f^c + 2 * (N_f^q - 1)))$. Since there is no guarantee that we find a frame at the exact relative position in the candidate model we use the best matching of the left and right neighbors in the closest available orientations (this explains the multiplication by 2 in the above equation). Please see Fig. 4 for illustration. In Section 4 we will show that in spite of its lower complexity the hit-rate is slightly above of EIS thus orientation information from IMUs could compensate the omitted extensive search of multiple visual descriptors.

3.3.3. EIIS (Extensive image and IMU search)

While the previous approach can be vulnerable to the only query frame used to find the right view in each candidate, by involving all queries in the extensive search we can utilize all available information of the query. Thus we test all frames from the query against all views of the candidate but when computing the distance we keep the constraint that the frames should be in the same relative orientation in the candidates and in the query. The distance is then:

$$T^{EIIS}(q, c_i) = \frac{\min_{c_{i,\triangle\alpha(k)}} \sum_{\forall k} T(q_k, c_{i,\triangle\alpha(k)})}{N_f^q} \tag{6}$$

where $c_{i,\triangle\alpha(k)}$ denotes the candidate model frame which has the same (or very close) relative orientation $\triangle\alpha$ as k frame in the query. The complexity of the EIIS approach can be described as $O(N_c * N_f^q * N_f^c)$.

3.3.4. Voting of candidates (VC)

The disadvantage of the previous methods was that those could not handle outliers properly. In real-life cases it can easily happen that an image is taken accidentally, or a query frame has so poor quality (e.g. due to the shaking of the hand) that it is should not be considered as a valid query. Since it is not easy to reliably evaluate the quality of a query itself, rather we consider lists of candidates that could correspond to the sequence of queries. This evaluation is made by a voting process: elements of independent retrieval lists will vote for the most reminder object view.

3.3.4.1. Hough transformation paradigm. The classical Hough transform was originally concerned with the identification of lines, later it has been extended to identify positions of arbitrary shapes (circles and ellipses) in images. Recently, Hough transformation is known to be efficiently used for object detection [24], object tracking [26] and action recognition [25]. In general there are two main steps of recognition in this framework:

Step 1. Feature extraction: visual, depth or other observation data can be used to generate descriptors ($d_i \in D$) to gather information about a space/time event or object.

Step 2. Voting: each occurrence of a descriptor votes for a candidate c_i with a weight $\Theta(c_i, d_j)$. In many cases the weights are determined by training and their value can also depend on other factors (such as the temporal distance from the candidate).

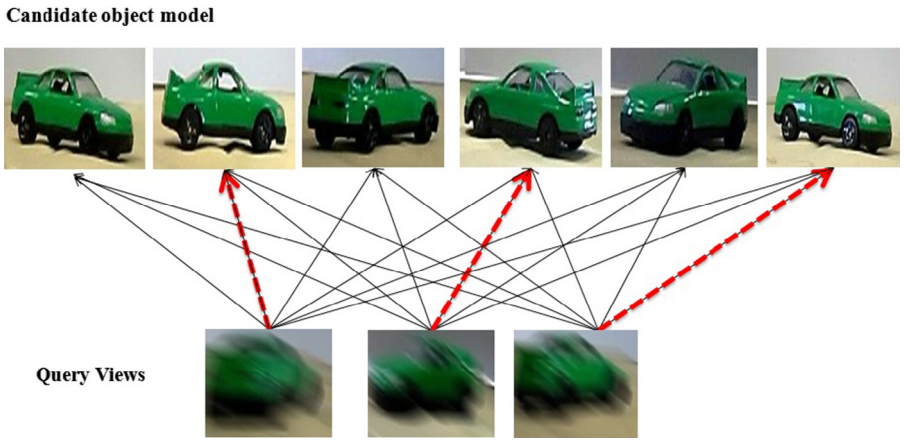

Fig. 3. Illustration of the EIS approach: only the images of the query are compared to the images of the candidates independently. The similarity of the query sequence and candidates are based on the sum of Tanimoto Coefficients.

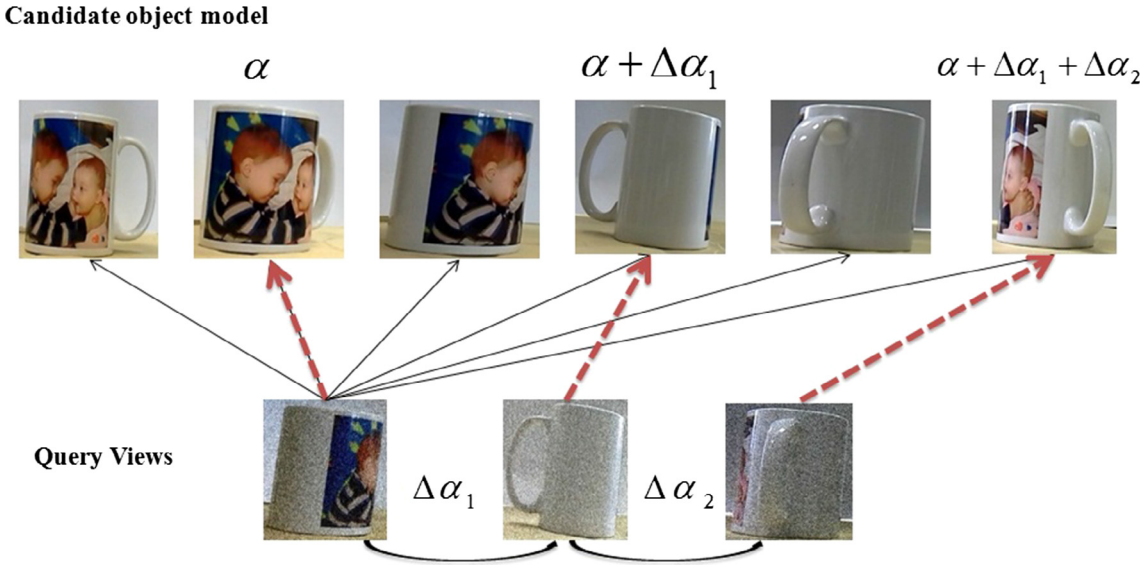

Fig. 4. Illustration of the SIIS approach: after finding the best matching frame of a query and a candidate, other frames are also compared selected on the bases of their similar relative positions as the frames of the query.

In general the Hough Score for a candidate is given:

$$H(c_i) = \sum_{d_i} \Theta(c_i, d_i), \tag{7}$$

and the recognition is done by selecting the object with the highest score:

$$\hat{c} = \arg \max_i H(c_i). \tag{8}$$

In our framework d_i will be the CEDD descriptors and IMU data, and Eq. (7) will be evaluated over the set of candidates of independent retrieval lists as described below.

3.3.4.2. Voting of candidates. In our case the time instances are limited to the query occasions. Unfortunately, the orientation data served by the IMU sensors can't be used as part of the descriptor, since those are relative values, rather the difference between two views can be used (as will be introduced into the voting itself).

In our retrieval process all queries q_i (some frames of input the video sequence) generate their own retrieval list $L_i(l)$ with limited length (e.g. $N_L = 4$) by running independent searches in the object models. That is we have a sequence of retrieval lists, one for each query view, and all the retrieved candidates give votes based on visual similarity and relative orientation. As the Tanimoto Coefficient measure similarity between the query and the actual candidate $TC(q_i, c_{j,f})$ will be one term of the vote, while the weighted term will be responsible for the orientation difference: $|\triangle \beta_{i,k,l} - \triangle\alpha_k|$ where $\triangle\beta_{i,k,l}$ is the difference between the orientation values of two frames of candidate object i:

$$\triangle\beta_{i,k,l} = \alpha(c_{i,k}) - \alpha(c_{i,l}), \tag{9}$$

and $\triangle\alpha_i$ is the difference between the orientation values of consecutive queries:

$$\triangle\alpha_i = \alpha(q_i) - \alpha(q_{i+1}). \tag{10}$$

So for the Hough Score we get:

$$H(c_i) = \max_{\mathbf{j}\ s.t.\ c_{i\mathbf{j}_k} \in L_k} \left(TC(q_{N_f^q}, c_{i\mathbf{j}_{N_f^q}}) + \sum_{k=1}^{N_f^q - 1} (TC(q_k, c_{i\mathbf{j}_k}) - w|\triangle\beta_{i\mathbf{j}_k\mathbf{j}_{k+1}} - \triangle\alpha_k|) \right), \tag{11}$$

where vector $\mathbf{j}$ stores the indices of frames of object i in relevance of query k (please note, that we have separate $\mathbf{j}$ for each candidate

object), and w is a constant weight. That is $\mathbf{j}$ will contain indices of c_i frames maximizing the Hough Score for object i ($\mathbf{j}_k$ for query q_k).

Practically, satisfying Eq. (8) means the evaluation of Eq. (11) when traveling through all paths connecting the views of the same objects on the different retrieval lists (see Figs. 5 and 6 for illustration). Since the length of the lists is limited it can easily happen that no representative of an object appears on the lists. If an object is not on any of the lists then simply it is out of focus of our search. However, if it appears on any but missing from some, then those lists are extended with a view of the object s.t. its relative orientation corresponds to the relative orientation of the actual query.

The complexity of this approach is roughly the same as of EIS, which can be considered as high.

3.3.4.3. Feature indexing. View-centered video-based object recognition and retrieval approaches can use thousands of views and thus can easily suffer from high complexity. In our model we have not only one but several CEDD descriptors of the objects extracted from different viewing directions. Object features might overlap among the images of the same object which requires special attention to keep the size of model database at minimum. Moreover, as we have to run several queries in the VC approach fast searching mechanisms are required. KD-Tree is an efficient data structure, established by Friedman, Bentley and Finkel [7], and is often used for fast indexing and retrieval. In [8] authors improved the KD-Tree for a specific usage: indexing a large number of SIFT and other types of image descriptors. They also extended priority search to search among multiple trees in a simultaneously way. In [9] parallel KD-Trees were explored for ultra large scale image retrieval in databases containing dozens of millions of images.

For the VC method we use a single KD-Tree containing the CEDD descriptors of all objects, as the number of candidate views (typically below 100,000) does not require the use of such multiple tree solutions. The tree was built in the standard way, based on the variance and mean of CEDD descriptors, as given by [7]. The complexity of the VCI (Voting Candidates + Indexing) approach now changes to: $O(N_f^q * N_f^{leaf}) + O'(N_L, N_f^q)$, where N_f^{leaf} is the number of frames in the KD-Tree leaf node (typically around 14). Please note that the complexity of the combinatorial evaluation of possible paths through the retrieval list can be high (exponentially increasing for N_f^q) so it is given by O', since not being on the core bases of comparing two CEDD descriptors. That is, besides the rough estimations summarized in Table 1, we need time measurements to get a better picture of time complexity as will be given in Section 4.5.

3.3.4.4. Post filtering for recognition. In many applications it is possible that untrained objects are targeted to be recognized. To avoid false recognition we have to evaluate the confidence of the best hypothesis found by the VCI method. If we find the confidence of the best candidate low, we should reject it and declare that the views of an unknown object were captured. For this purpose we extended the object models to store the CEDD differences (Eq. (2)) with c_j^{Noise} the views loaded with distortion (resulting in matrices of size $N_f^c * N_f^c$). Now, we are able to estimate a view-dependent average "typical" distance of model views and noisy approximations:

$$T^{Typ}(c_{i,(l,m,\ldots)}) = \frac{1}{N_q^f} \sum_{\forall k \in (l,m,\ldots)} T(c_{i,k}, c_{i,k}^{Noisy}) \tag{12}$$

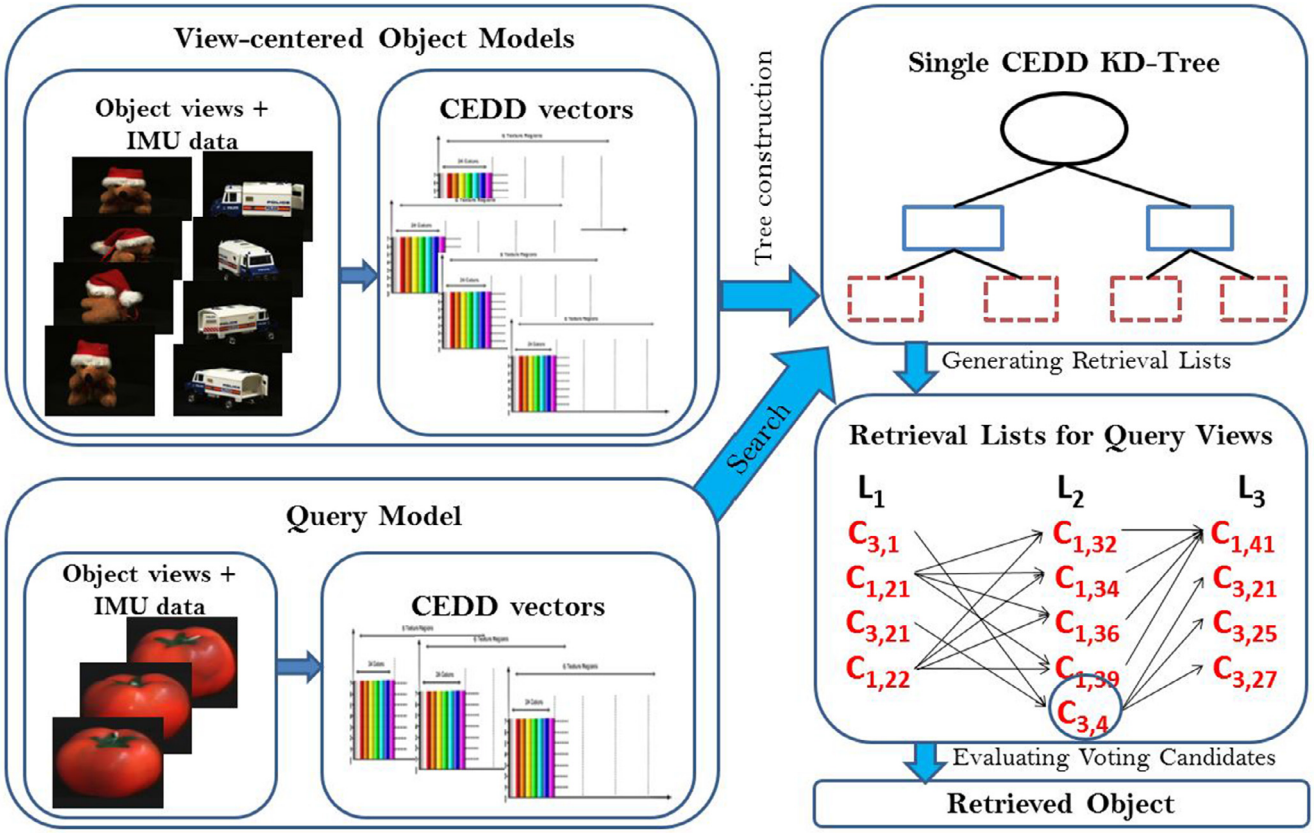

Fig. 5. Illustration of the VCI approach with three retrieval lists ($N_f^q = 3$). Since C_3 was not on L_2 but on L_1 and L_3, $C_{3,4}$ was added based on its relative orientation.

Algorithm: VCI search

Input:

1- One binary KD-Tree (K) for all CEDD vectors ($\mathbf{x_i}$) and orientation data for all object model views in the training database; each tree node k_i contains also two variables k_{max} (the index of the highest value of the variance vector calculated from all vectors $\mathbf{x_i} \in k_i$), and k_{mean} (the mean of vectors in node k_i at k_{max}).

2- The CEDD vectors ($\mathbf{y_i}$) for query Q, i is the index of query views, and $\mathbf{o_i}$ is the orientation data for query views.

Output: Best matching candidate object for query Q.

Operation:

1. For each vector $\mathbf{y_i} \in Q$, traverse K from root node with $\mathbf{y_i}$ as follows:
   - i. If traversed node k_i is leaf node:
     - (a) For each view $\mathbf{v} \in k_i$, compute Tanimoto Coefficient $T(\mathbf{v}, \mathbf{y_i})$.
     - (b) Choose the four $\mathbf{v}$-s with the smallest T distances.
     - (c) Generate retrieval list $L_{\mathbf{y_i}}$ for $\mathbf{y_i}$, consisting of object views of the $\mathbf{v}$-s of point (b) above.
   - ii. If traversed node k_i is not leaf node:
     - (a) If $\mathbf{y_i}[k_{max}] \leq k_{mean}$, then traverse to the left child node of k_i, else traverse to the right child node of k_i.
2. Let R_L be the set of retrieval lists for Q, then evaluate candidate object c_i based on retrieval lists:
   - i. If c_i appears on any retrieval list but missing from some, then those lists are extended with a view of c_i s.t. its relative orientation corresponds to the relative orientation of query.
   - ii. For each c_i being on the retrieval lists R_L, compute $H(c_i)$ using Equation 11.
3. The object which has highest value is the best matching candidate object for the query Q.

Fig. 6. VCI (Voting Candidates + Indexing) search algorithm.

Table 1
Complexity of different approaches at $N_c = 16, N_f^c = 50, N_f^{leaf} = 14$, and different N_f^q.

Method	$O()$	Query Views (N_f^q)			
		2	4	8	16
EIS, EIIS, VC	$O(N_c * N_f^q * N_f^c)$	1600	3200	6400	12,800
SIIS	$O(N_c * (N_f^c + 2 * (N_f^q - 1)))$	832	896	1024	1280
VC + KD-Tree (VCI)	$O(N_f^q * N_f^{leaf}) + O'(N_L, N_f^q)$	$28 + O'$	$56 + O'$	$112 + O'$	$224 + O'$

A hypothesis can be tested by thresholding:

$$Decision = \begin{cases} \text{Accept candidate} & \text{if } T^M/T^{Typ} \leqslant Th \\ \text{Reject candidate} & \text{if } T^M/T^{Typ} > Th \end{cases} \quad (13)$$

where T^M is the average distance of the query and the candidate according to the views on the retrieval lists of the VCI method. The reader should be aware that this confidence model is orientation adaptive: i.e. some views of objects can be modeled with high sensitivity to noise (if T^{Typ} is relatively small) while other views of the same object with low sensitivity if (T^{Typ} is relatively large). Please note, that this post-filtering step will be evaluated only in the tests of Section 4.3.

4. Experiments and results

In this Section we introduce the various (fully 3D-shaped) object model datasets, the test queries, the hit-rate of the different methods and the measured time complexity.

4.1. Object model datasets

We used three separate datasets, with different size and nature, for testing purposes: SUP, ALOI, and SUP-25.

SUP is a small dataset including 16 objects, where 44–73 views per object were captured by our tablet from the same elevation but from different azimuth leading to approximately 900 images. Image sizes and side ratios varied a lot as shown in Fig. 7.

The second one is the ALOI dataset [13] including 1000 small objects, where 72 images of each object were recorded by rotating the object in the plane at 5° steps, as examples show in Fig. 8.

The SUP-25 dataset, illustrated in Fig. 9, includes 25 different objects, where 64–77 images of each object were recorded from different views at the same elevation leading to approximately 1800 images. This dataset was created by our tablet to test the VCI approach with automatic tracking applied for the generation of queries.

4.2. Retrieval performance on SUP and ALOI datasets

The purpose of these experiments is to show the advantage of the retrieval performance of the VCI over the other approaches. Since each object was tested 10 times, the query datasets (separate for the different datasets) are composed of 10×8 ($N = 8$) randomly selected images of each object, either distortion free, strongly distorted with motion blur, or by additive Gaussian noise. (Please note, that while the 8 views are set to be different, the 10 random test cases have common views at chance.) We have chosen these two types of distortions since in previous evaluations [16] we found many image descriptors to be most vulnerable to these common quality degradations happening often in real life. We used the built-in function of Matlab *imnoise* with standard deviation $sd = 0.012$ to generate additive Gaussian noise (GN) and made motion blur (MB) by *fspecial* with parameters $len = 15$, and angle $\theta = 20$ degrees. Some examples of the distorted queries are shown in Fig. 10.

To be more realistic we removed not only the query images from each model during testing: we deleted every view from the models within 10° angle from a possible query. (We have chosen 10° since it is slightly above the accuracy of the IMU sensor as shown in Fig. 2.) Since the queries were randomly selected the closest orientation angle between a query and an available model view was observed to be between 10 and 30 degrees. Thus the size of the model sets reduced to 34% for the SUP, and to 69% for the ALOI dataset. In tests with ALOI the IMU data of the queries were loaded with the noise measured in Section 3.1.

We tested all approaches using the SUP and the ALOI dataset, all with distortion free and with strongly distorted queries (MB and GN).

Figs. 11 and 12 contains the hit-rates at different N_f^q-s. As it is easy to see the three basic methods run close to each other; EIS, SIIS, EIIS is the increasing order. That is the orientation information, included into the different models, could add valuable information to improve the hit-rate of these methods. Our proposed VCI model overcomes all these three significantly except for only very few data points. Higher number of queries results in higher hit-rates unexceptionally and GN makes more problem for the descriptor than MB.

We also evaluated the role of w in Eq. (11). Fig. 13 contains information for only the ALOI dataset (due to space limitations). It seems that the optimal value for w is somewhere between 0.1 and 0.5 and comparing with the case $w = 0$ the maximum improvement is below 4%. That is a large amount of the contribution of orientation information is not via the orientation term of Eq. (11), but by the fact that the retrieval lists are occasionally extended by missing candidates based on the relative orientation of the actual view.

4.3. Recognition performance with the SUP dataset

To achieve recognition instead of retrieval we should evaluate the confidence of the best candidate and make a decision about a possible false acceptance (see Section 3.3.4.4). To test the performance of our approach from this aspect we added 9 untrained

Fig. 7. Test model object examples from the SUP dataset.

Fig. 8. Test model object examples from the ALOI dataset.

Fig. 9. Test model object examples from the SUP-25 dataset.

object to SUP (see some examples on Fig. 14) and applied MB for the queries.

Fig. 15 shows the results for different *Th* thresholds of the post-filtering method. As would be expected the results are worse than for retrieval with the 16 objects. Now, a hit was counted only when an object was correctly recognized or an untrained object was recognized as unknown. In these tests the increase of the number of query views had also a positive effect on the hit-rate giving the best results at $Th = 2$.

4.4. Retrieval performance with automatic segmentation on the SUP-25 dataset

In some human operated real-life applications users are to select a target object then the application should make the queries automatically from different views. Queries for the experiments with SUP-25 were recorded with the tablet. The user was asked to mark the target object with a bounding box on the live image then continuous adaptive mean-shift was to track it when moving the object around (typically 90° change in viewing direction at the same elevation). While the targeted objects were not occluded they were surrounded with others and the background was different than in the models. The automatic tracking can adjust the position and the size of the bounding window by using invariant moments, these windows are given to the query engine. The principle of the tracking algorithm is given in details by [33]. Fig. 16 illustrates some tracked windows and the environment of the objects.

Table 2 contains the hit-rates for different number of queries telling that even a simple tracking algorithm was successful to

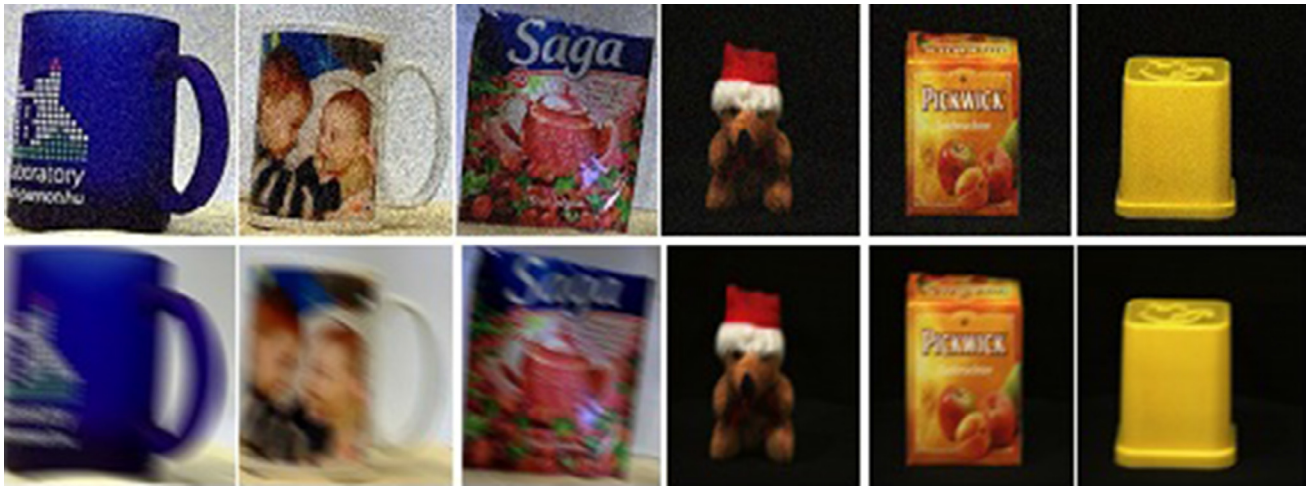

Fig. 10. Noisy and blurred query examples from the SUP, and ALOI datasets.

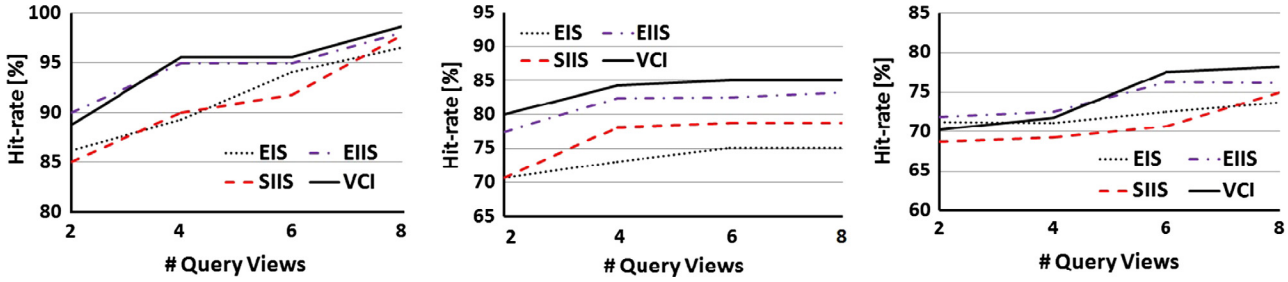

Fig. 11. Average hit-rate for query images without distortion (left), with strong motion blur (middle), and heavy additive Gaussian noise (right) for the SUP dataset and $w = 0.5$.

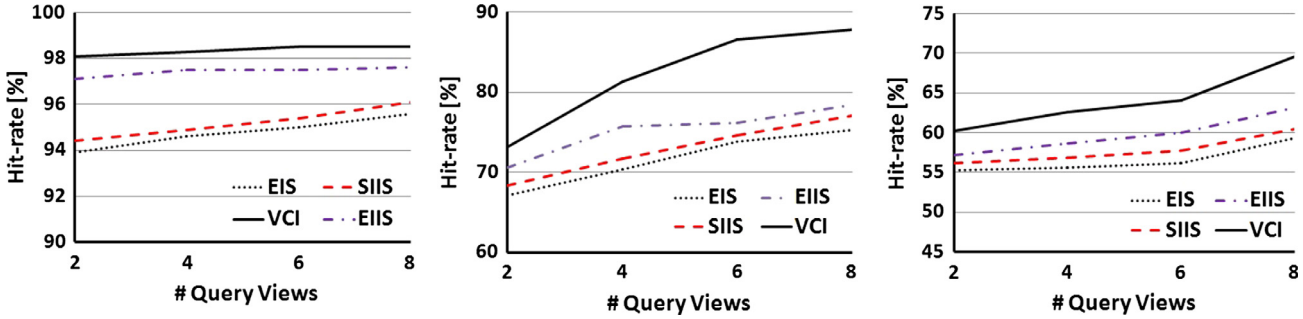

Fig. 12. Average hit-rate for query images without distortion (left), with strong motion blur (middle), and heavy additive Gaussian noise (right) for the ALOI dataset and $w = 0.5$.

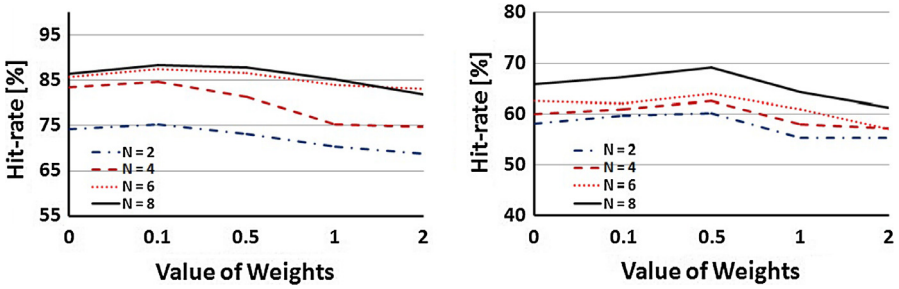

Fig. 13. Average hit-rate of the VCI method for motion blur (left) and additive Gaussian noise (right) at different query views (N) and w settings for the ALOI dataset.

Fig. 14. Examples for untrained objects to test recognition.

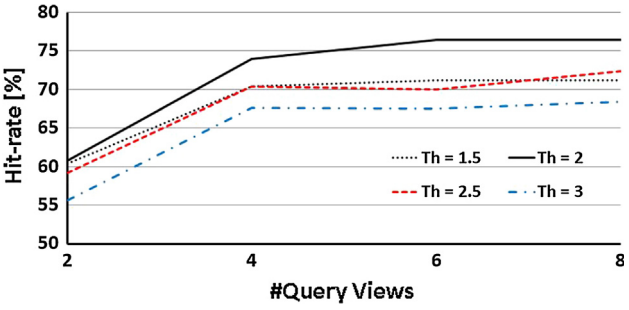

Fig. 15. Average hit-rate for the SUP dataset with 9 extra untrained queries with different thresholds.

generate queries to reach almost 100% for $N_f^q = 8$. Comparing these results to those of Figs. 11 and 12 we find that our previous estimations were not far from these real-life tests.

4.5. Running times

As we have seen it is not convenient to evaluate the complexity of the methods since the different parts of the algorithms react differently for the various parameters. For this reason we implemented them on a tablet and made time measurements. Tests were run on a Samsung SM-T311 tablet equipped with Android 4.2.2 Jelly Bean, 1 GB RAM, and ARM Cortex A9 Dual-Core 1.5 GHz Processor. Fig. 17 contains the average running time of 10 measurements on a database of 100 objects. Only the retrieval mechanisms are considered in this graph for comparison, the generation of CEDD descriptors (which is about 0.04 s for a frame of size 640×480) and the tracking algorithm (0.7 s per frame) is not included. As expected from Table 1 SIIS has low running time just as VCI for low N_f^q. However, while SIIS increases almost linearly with N_f^q, VCI grows exponentially due to the $(N_L)^{N_f^q}$ combinatorial evaluations of possible paths through the retrieval lists.

Fig. 16. Top row: examples for the results of tracking. Bottom row: objects and their environment.

Table 2
Hit-rate of VCI with object tracking for SUP-25.

Method	Query Views (N_f^q)			
	2	4	6	8
VCI	72%	84%	92%	96%

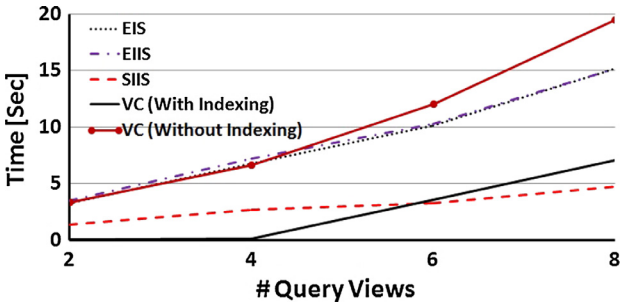

Fig. 17. Average running time for EIS, SIIS, EIIS, VC and VCI approaches on a dataset of 100 objects.

Finally, we can conclude that our VCI code (without special code optimization or code parallelism) can achieve slightly below 8 s/8 queries, for a database of 100 objects, resulting in practically real-time operation on a mobile device (for SUP it is 0.34 s/8 queries). The memory space required for a database of 100 objects is around 1.2 MB.

5. Conclusion

The recognition of 3D objects is important for future's autonomous systems and for other intelligent applications. However, it is not only the 3D shape, but the 2D views from different directions of objects, that can help us in the retrieval or recognition of 3D objects. To get orientation information of the camera, the cheap and reliable IMU sensors can be used. Our main contribution is showing that view-centered models can easily fuse visual and orientation information resulting in the increase of retrieval performance. We proposed a method based on the Hough paradigm named Voting Candidates where the orientation information was involved in supplement of answers to queries and also considered in the evaluation of similarity. The time complexity (without GPU or code optimization) is close to real-time and allows the application of the model in lightweight units such as embedded systems in the near future. Our results are supported by tests on three datasets, using large number of queries loaded with significant noise, and real-life retrieval combined with tracking. We believe not only CEDD but other descriptors could also benefit from IMU sensors and the performance of the approach could even be improved using HMMs to analyze the probability of candidates through a sequence of observations.

Acknowledgment

The work and publication of results have been supported by the Hungarian Research Fund, grant OTKA K 120367 and by Szchenyi 2020 under EFOP-3.6.1-16-2016-00015.

References

[1] S.A. Chatzichristofis, Y.S. Boutalis, accurate image retrieval based on compact composite descriptors and relevance feedback information, Int. J. Pattern Recog. Artif. Intell. (2010) 207–244.
[2] S.A. Chatzichristofis, Y.S. Boutalis, M. Lux, Selection of the proper compact composite descriptor for improving content based image retrieval, in: 6th International Conference on Signal Processing, Pattern Recognition and Applications (SPPRA), 2009, pp. 134–140.
[3] L. Czúni, M. Rashad, Lightweight video object recognition based on sensor fusion, in: International Workshop on Computational Intelligence for Multimedia Understanding (IWCIM), 2015, pp. 1–5.
[4] L. Czúni, M. Rashad, View centered video-based object recognition for lightweight devices, in: International Conference on Systems, Signals and Image Processing (IWSSIP), 2016, pp. 1–4.
[5] A. Bruno, L. Greco, M. Cascia, Video object recognition and modeling by SIFT matching optimization, in: ICPRAM, 2014, pp. 662–670.
[6] H. Noor, S.H. Mirza, Y. Sheikh, A. Jain, M. Shah, Model generation for video-based object recognition, in: Proceedings of the 14th Annual ACM International Conference on Multimedia, 2006, pp. 715–718.
[7] J. Friedman, J. Bentley, R. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Trans. Mathe. Soft. 3 (1977) 209–226.
[8] C. Anan, R. Hartley, Optimised KD-trees for fast image descriptor matching, in: IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2008.
[9] M. Aly, M. Munich, P. Perona, Distributed Kd-trees for retrieval from very large image collections, in: Proceedings of the British Machine Vision Conference (BMVC), 2011.
[10] L. Czúni, P.J. Kiss, A. Lipovits, M. Gál, Lightweight mobile object recognition, in: IEEE International Conference on Image Processing (ICIP), 2014, pp. 3426–3428.
[11] A. Tungkasthan, S. Intarasema, W. Premchaiswadi, Spatial color indexing using ACC algorithm, in: 7th International Conference on ICT and Knowledge Engineering, 2009, pp. 113–117.
[12] O. Miksik, K. Mikolajczyk, Evaluation of local detectors and descriptors for fast feature matching, in: 21st International Conference on Pattern Recognition (ICPR), 2012, pp. 2681–2684.
[13] J. Geusebroek, G.J. Burghouts, A.W.M. Smeulders, The amsterdam library of object images, Int. J. Comput. Vis. 61 (2005) 103–112.
[14] O. Javed, M. Shah, D. Comaniciu, A probabilistic framework for object recognition in video, in: International Conference on Image Processing (ICIP), 2004, pp. 2713–2716.
[15] S. Gammeter, A. Gassmann, L. Bossard, T. Quack, L. Gool, Server-side object recognition and client-side object tracking for mobile augmented reality, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2010, pp. 1–8.
[16] L. Czúni, P.J. Kiss, A. Lipovits, M. Gál, Mobile Object Recognition: An analysis of Image Quality Factors, Technical Report, UofP, 2014.
[17] C. Rother, V. Kolmogorov, A. Blake, Grabcut: Interactive foreground extraction using iterated graph cuts, ACM Transactions on Graphics (TOG), vol. 23, 2004, pp. 309–314.
[18] H. Bülthoff, S. Edelman, Psychophysical support for a two-dimensional view interpolation theory of object recognition, Proc. Nat. Acad. Sci. 89 (1992) 60–64.
[19] F. Fang, S. He, Viewer-centered object representation in the human visual system revealed by viewpoint aftereffects, Neuron J. 45 (2005) 793–800.
[20] C.E. CH, L.F. Pau, P.S.P. Wang, Viewer-centered representations in object recognition: a computational approach, in: Handbook of Pattern Recognition and Computer Vision, 1993, pp. 863–882.
[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[22] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1106–1114.
[23] J.D. Hol, T.B. Schön, F. Gustafsson, A new algorithm for calibrating a combined camera and IMU sensor unit, in: 10th International Conference on Control, Automation, Robotics and Vision(ICARCV), 2008, pp. 1857–1862.
[24] J. Gall, V. Lempitsky, Class-specific hough forests for object detection, in: International Conference on Computer Vision and Pattern Recognition, 2013, pp. 143–157.
[25] A. Chan-Hon-Tong, C. Achard, L. Lucat, Deeply optimized hough transform: application to action segmentation, in: International Conference on Image Analysis and Processing, 2013, pp. 51–60.
[26] J. Gall, A. Yao, N. Razavi, L. Van Gool, V. Lempitsky, Hough Forests for object detection, tracking, and action recognition, IEEE Trans. Pattern Anal. Mach. Intell. (2011) 2188–2202.
[27] C. Shi-Kuo, E. Jungert, Symbolic Projection for Image Information Retrieval and Spatial Reasoning, Academic Press, London, 1997.
[28] N. Ahmad, R.A.R. Ghazilla, N.M. Khairi, V. Kasi, Reviews on various inertial measurement unit (IMU) sensor applications, Int. J. Sig. Process. Syst. 1 (2) (2013) 256–262.
[29] J. Han, L. Shao, D. Xu, J. Shotton, Enhanced computer vision with microsoft kinect sensor: a review, IEEE Trans. Cybernet. 43 (5) (2013) 1318–1334.

[30] M. Godec, C. Leistner, H. Bischof, A. Starzacher, B. Rinner, Audio-visual co-training for vehicle classification, advanced video and signal based surveillance (AVSS), in: 7th IEEE International Conference, vol. 1, 2010, pp. 586 – 592.
[31] M. Irani, P. Anandan, About Direct Methods, in: International Workshop on Vision Algorithms, Springer, Berlin Heidelberg, 1999, pp. 267–277.
[32] P.H. Torr, A. Zisserman, Feature based methods for structure and motion estimation, in: International Workshop on Vision Algorithms, Springer, Berlin Heidelberg, 1999, pp. 278–294.
[33] D. Exner, E. Bruns, D. Kurz, A. Grundhfer, O.J. Bimber, Fast and Robust CAMShift Tracking, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 2010, pp. 9–16.