

OBJECT LOCALIZATION USING KINECT

Irina Mocanu¹
Iulius Curt²

ABSTRACT

Ambient intelligence is an emergent topic today and it involves scene understanding and object recognition. Because for scene understanding the position of objects is needed, a binary classification that decides if an object is present or not in the scene is not sufficient. The present paper proposes a system for localization of objects by a 3D bounding box. This is achieved by augmenting the 2D location, extracted using a sliding-window-based method, with 3D information acquired from a stereo-camera. While the sliding window is very computationally intensive, a branch-and-bound approach is used, which reduces the processing time, typically running in sub linear time, without discarding the optimality guarantee. An SVM is employed for classification purposes based on the bag-of-visual-word representation to obtain discrimination between object classes. The system is tested on real-world objects using a Microsoft Kinect sensor.

Keywords: 3D, branch-and-bound, localization, SVM, bag-of-visual-word, stereo-vision

1. INTRODUCTION

Ambient intelligence has the purpose to improve the human-computer interface such that technology aids the person in everyday activities with minimum interaction between the two. Scene understanding is a form of context awareness. When understanding a scene, after the objects of interest are detected and located, a model of the scene is generated. Various relations and conclusions can be, then extracted from this model. For example, an action may be triggered when a person is located in the proximity of a specific object.

This paper describes a system for recognition and 3D localization of objects in a scene. The module is meant to come as a black box with the input connected to the output of a Microsoft Kinect sensor and the output connected to other modules that use object positioning to obtain relationships between objects and to do higher-level reasoning. There are many different ways an object's location can be represented. Location can be specified by the object's center, its pixel-wise segmentation, its bounding box, as a relation to other objects etc. The proposed system uses the bounding box approach, meaning that it aims to find the smallest rectangle box that encloses the object.

The system can be integrated into an Automated Surveillance System for Disabled Persons. Such a system would consist of the following modules: (1) Object recognition and localization module; (2) Human subject localization and posture recognition module (3)

¹Lecturer, University POLITEHNICA of Bucharest, Splaiul Independentei No. 313, Bucharest 060042, Romania, irina.mocanu@cs.pub.ro

²Master Student, Artificial Intelligence, University POLITEHNICA of Bucharest, Splaiul Independentei No. 313, Bucharest 060042, Romania, iulius.curt@gmail.com

Module that extracts semantic relationships between the recognized objects (4) Decision taking module.

The rest of the paper is organized as follows. Section 2 describes some theoretical methods used for object recognition. The description of the proposed system is given in Section 3. Section 4 presents the current evaluation of the proposed system. Conclusions and future works are listed in Section 5.

2. RELATED WORKS

In the last decade, many innovations have been made in the field of computer vision. New ideas and techniques were developed on all levels, from the low-level feature descriptors [Lowe, 2004] [Bay, 2006], to specialized classifiers [Joachims, 2009] [Yu, 2009] and high-level algorithmic approaches [Lampert, 2008] [Felzenszwalb, 2008]. The problem of object detection and the problem of object localization are two different problems that involve different approaches. The object detection answers questions of the form: in an image, is such an object present or not? On the other hand, the localization problem, in the affirmative case of detection, also asks for the position of the existing object. Although, in recent years, the state-of-the-art algorithms and techniques for object detection performs well, they tend to become either inefficient or intractable on the harder problem of object localization. A different approach from the one used in this paper is the deformable part-based objects model [Felzenszwalb, 2008]. This approach treats objects as sets of object-parts linked together. For the example of a human person, parts would be: the torso, the head, the hands and legs. The descriptors that can be found in the present implementation are SIFT (Scale Invariant Feature Transform) [Lowe, 2004] descriptors, dense SIFT and SURF descriptors. All of those descriptor types aim to represent visual features in a robust way, invariant to a number of image deformations. All three descriptor types are scale-invariant, highly distinctive feature descriptors that, also, have some degree of invariance to rotation, illumination and viewpoint.

SIFT key points detection algorithm is based on blurring the image by convoluting it with Gaussian filters at different scales and subtracting them, as in Figure 1. Key points are chosen at points of local maxima. Because the SIFT descriptor is compound of 128 integer values, features described by it are placed in a 128-dimensional vector space for clustering purposes.

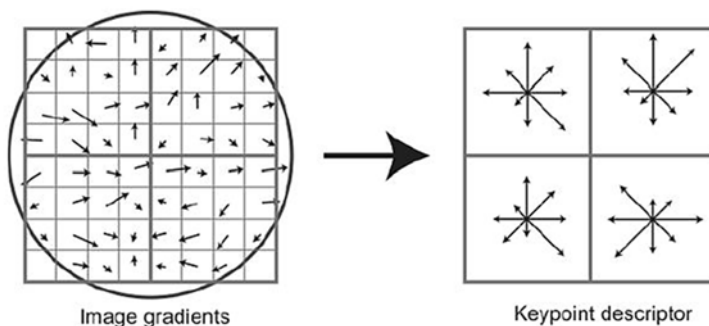


Figure 1. Orientation assignment principle, as applied for SIFT method (as in [Lowe, 2004]).

The major characteristic of such approaches is the fact that the descriptor is computed only for key points within the image. These are points which allow constructing descriptors which are invariant to scale and rotation and robust to change of illumination [Lowe, 2004].

The main benefits of SIFT method, as indicated by the author [Lowe, 2004] are: it is scale invariant (in other words different sizes of the object do not make a difference) and rotation invariant (perspectives from different angles do not make a difference). It is also quite robust to affine distortion, noise and changes in illumination. The image descriptor is a so called local descriptor, computed not for the entire image, but only for selected keypoints in the image. The algorithm consists of two principal parts, keypoints identification and descriptor computation. The keypoints identification process consists on several main steps: determine the interest points (points in the image which are not affected by different image scales), outlier rejection to obtain the keypoints and orientation assignment (selecting the dominant orientation for each keypoint).

The descriptor computation involves computing the gradients in a 16×16 region around each keypoint, dividing the region in 16 (4×4) blocks and for each compute a 8-bin histogram. For this, the orientation of the gradients is expressed relative to the keypoint orientation. The descriptor around each keypoint will be a vector with a $16 \times 8 = 128$ size.

The major stages in the algorithm are [Lowe, 2004]:

1. Scale-space extrema detection - identifies locations and scales that can be repeatedly assigned under differing views of the same object.
2. Keypoint localization - to reject points with low contrast or poorly localized.
3. Orientation assignment - to assign a consistent orientation to each keypoint based on local image properties. The descriptor can be represented relative to this orientation and achieve invariance to image rotation.
4. Keypoint descriptor - to compute a descriptor for the local image region that is highly distinctive yet is as invariant as possible to remaining variation.

Based on these results [Bosch, 2007], [Bosch, 2006] proposed the dense SIFT descriptors. DSIFT is usually accompanied by a clustering stage, where the individual SIFT descriptors are reduced to a smaller vocabulary of visual words, which can then be combined with a bag-of-words model or related methods [Csurka, 2004], [Lazebnik, 2006].

Experimental results on SIFT descriptors applied to image classification shows that better classification results are often obtained by computing the SIFT descriptor over dense grids in the image domain opposite to sparse interest points (practically skipping the first stages in the algorithm, which is the selection of the keypoints). A larger set of local image descriptors computed over a dense grid usually provide more information than corresponding descriptors evaluated at a much sparser set of image points.

3. SYSTEM DESCRIPTION

The paper describes a general purpose object location extractor that can be trained on any object class by giving it positive and negative examples of objects. Hence, it can be used in

many domains. The application is based on a machine learning system that works in two phases: a learning phase and a detection-localization phase. The system needs to first be trained before it can be run on input data, to produce expected output. The main modules of the system reads input data end extracts interesting features form it, encodes those features and using them to either learn how objects look like or to detect already learned objects from new input data. The general architecture of the system is described in Figure 2.

In Figure 3 is given the detailed specification of the system. The main modules of the application are:

- Feature Extraction Module
- Feature Encoding Module
- Training Classification Module
- Object Detection Module
- Depth Extractor Module

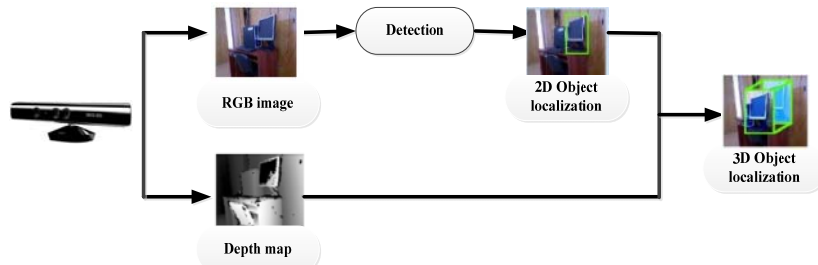


Figure 2. The general system architecture

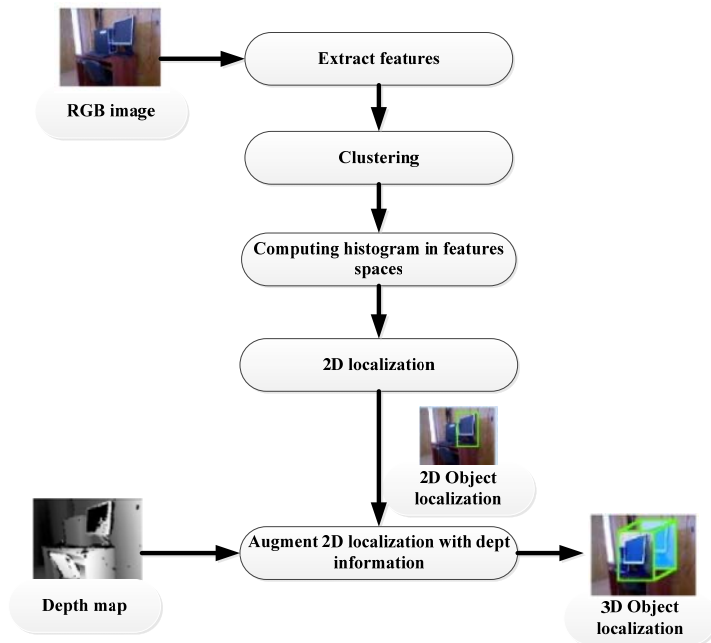


Figure 3. The main modules of the system

I. Feature Extraction Module

To be able to represent an object in an efficient way, some descriptors must be selected, such that they specifically identify the object, but they also offer some degree of variance (in object's scale, rotation, brightness etc.). For this reason, some interesting regions inside the object are chosen, called feature points. Feature points may use information such as color of the pixels in the region it represents, lines and edges, gradients or/and many other aspects. In the present implementation feature descriptors are based on gradients.

II. Feature Encoding Module

Because it is very improbable to find an exact specific feature twice in the data set, some degree of freedom needs to be added. This is achieved by encoding the raw extracted features into a more versatile form. The encoding step consists of clustering features into a predefined number of groups. All the features in a cluster are represented by the cluster's centroid. The resulted centroids are gathered in a codebook and saved for further reference. The codebook is generated only once, in the training stage.

A codebook is a collection of entities that are given generic names, or codes. The trivial example of such codes is a 0-based or 1-based index of the entity in the collection, over some ordering. In computer vision, a popular usage of codebooks is to hold specimens of visual descriptors and map them to a unique index. This enables a more efficient encoding of visual features. From the codebook creation process results a label mapping for each cluster centroid. When a new feature specimen is extracted from test data, it is assigned to the best fitting cluster and a label from the codebook is assigned to it.

In the classification stage, it is read from the disk and used for the raw feature descriptors extracted from the test data. These last mentioned features are given a tag with the most similar centroid in the codebook. To improve generalization, a clustering process is applied over the raw set of the extracted features. Such, a newly extracted feature vector must not be the exact copy of a vector that was previously learned and can be located at a small distance in the vector space of the features and still be recognized. The extracted features from the entire train data set are clustered to a relatively small number of clusters. The centroids of the resulting clusters make up a codebook. For each image file in the training set, its extracted feature vectors are clustered under the nearest centroid in the codebook and a histogram quantization is generated. K-Means clustering algorithm is used as the clustering method. The number of clusters is chosen equal to the minimum between the square root of the total number of extracted features and 200. The use of the maximum value of 200 clusters was reached based on observations from [Lazebnik, 2006].

The k-means algorithm [Queen, 1967] is a centroid-based clustering algorithm that aims to group the input data points in k regions, with k given, based on their relative positions in a vectorial space. Because the optimization problem of centroid-based clustering is an NP hard problem, there is no guarantee of a global optimum to be reached. Also, the k-means algorithm is not guaranteed to converge. The algorithm, in its basic formulation, starts by randomly placing all the k centroids in the same space with the data points. Then, an assignment step followed by an update step are iteratively repeated until the system converges, or other heuristic condition is met. In the assignment step, each data point is

assigned to the nearest centroid. In the update step, each centroid is moved such that the sum of distances from it to each data point in its cluster is minimized. In the k-means algorithm the update step is achieved by computing the mean position of all the data points in the cluster.

In the testing phase, extracted features are assigned to cluster centroids and have codebook labels associated. Using these labels, a histogram of the test image is generated. Such a histogram can be placed in a n-dimensional vector space (where n is the size of the codebook). In this vector space a distance can be computed between two images.

III. Training Classification Module

The purpose of a classifier in the present architecture is to discriminate between good features and bad features in the context of appurtenance to a class of objects. For this purpose, a SVM is trained for each object class. The classifier training module is present only in the training phase, when the classifier model is generated and saved on disk for further use. Later, in the testing stage, the trained SVM model is loaded from disk. Linear support vector machines (SVM) are linear supervised classifiers that aim to find a hyperplane in the space of the features, such that the margin around the hyperplane to the nearest points is maximized. This ensures better generalization. In this case we use a linear SVM is trained to discriminate between objects classes. The training data consists of positive and negative examples of images for a specific class of objects. The SVM learns from these examples to classify feature points from the test data as belonging to the objects class or not. Hence, it can be used as an object detector. After the SVM is trained, it is computed the maximum-margin hyperplane, in the vector space of the feature descriptors. This hyperplane can be uniquely identified by a set of weights, the same parity as the codebook size.

IV. Object Detection Module

The module is able to detect and 2D localize object classes that were learned beforehand in the training stage. After feature extraction and feature encoding are applied on the test data, a classifier is used together with a localization strategy. The SVM model for the desired object class is loaded from disk and used to discriminate between positive and negative features. A sliding-window algorithm, optimized by a branch-and-bound approach is used for searching in the space of image sub-region candidates. The objects detection module outputs one or more 2D bounding box localized objects.

For object 2D localization the branch-and-bound sliding window technique is applied. This implies repeatedly computing scores for sub-regions in the test image by isolating the features located in the sub-region and summing their weights. To be able to do it efficiently, the integral image technique is used. Two integral images are computed before the branch-and-bound's main loop, one containing the positive weights and the other containing the negative weights. These are used by the upper-bounding function in the branch-and-bound algorithm. The branch-and-bound algorithm involves extracting the element with the maximum score from a set. This is efficiently achieved with the help of a priority queue implemented as a heap data structure.

The Branch-and-bound approach is described as follows. Sliding window search is a method used to locate the best rectangle-shaped window in an image (or, more generally, a matrix). It is widely used for the purpose of bounding box localization. In its basic form, sliding window search checks or scores every possible sub-region of the image, at every location and every scale. This is too computationally intensive to be done exhaustively for medium to large image sizes. Several heuristic methods can be applied to reduce the number of considered sub-regions, e.g. enforce a specific window aspect ratio or check only the candidates of specific size. These are approximate solutions and they don't offer any guarantee of finding the global optimum. In this case we use a branch-and-bound technique based on the idea developed in [Lampert, 2008], as in Figure 4.

To reduce the number of candidates that need to be evaluated for the optimal result to be found, an upper bound for the evaluation function can be used on wide groups of candidates. In this way, the search can be stopped early when a candidate is found to have the score larger than or equal to the upper-bound scores of all the other candidate groups. For the purpose of applying the branch-and-bound strategy on the problem of image sub-region search, windows of different sizes and positions are grouped together in sets of rectangles. A quality function upper bound is defined over rectangle sets, such that it meets two conditions:

- it is always larger than or equal to the exact quality function of the best rectangle in the set;
- it is equal to the quality function when only one rectangle is in the set.

The branch-and-bound strategy offers the guarantee to find the optimal solution for any quality function upper bound that respects the aforementioned rules. The strategy evaluates candidates in a best-first manner and stops the search when the best candidate consists of a unitary rectangles set. Because the upper bound function is equal to the quality function for the unitary rectangle set, the current rectangle candidate is known to have higher score than or equal score to the upper bound of any other candidate, which in turn is greater than or equal to the quality function of any component of the other candidate sets. Hence, the first unitary set evaluated is the optimal solution.

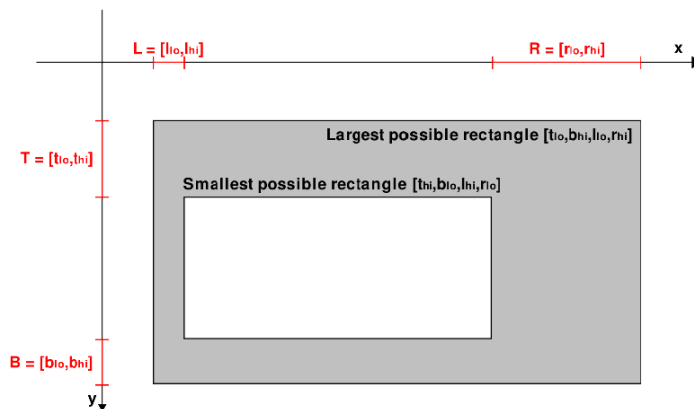


Figure 4. Rectangle computation based on 4 intervals (as described in [Lambert, 2008])

An ideal upper-bound function would be equal to the maximum score in the set. A trivial way to achieve this is to iteratively consider all the rectangles in the set and pick the best, which reduces the algorithm to the basic sliding-window. In practice, a compromise between the tightness of the upper-bound function and its time complexity must be made.

The branch-and-bound method implies repeatedly computing scores for sub-regions in the test image by isolating the features located in the sub-region and summing their weights. To be able to do it efficiently, the integral image technique is used. Two integral images are computed before the branch-and-bound's main loop, one containing the positive weights and the other containing the negative weights. These are used by the upper-bounding function in the branch-and-bound algorithm. The branch-and-bound algorithm involves extracting the element with the maximum score from a set. This is efficiently achieved with the help of a priority queue implemented as a heap data structure.

The integral image is a technique to efficiently compute sums of the values from sub-regions of an image (a matrix). Is called an integral image a matrix where each cell holds the sum of all the cells in the original image located above and to the left, including the same column and row as the current cell, and the cell itself. An integral image is obtained from the original matrix by iteratively summing each cell with its neighbor above, in the first step, and to its neighbor to the left in the second step.

To obtain the sum of a sub-region of the matrix using an integral image of it, the following relation is used:

$$S = \sum M(t, l, b, r) = I[b, r] - I[b, l] - I[t, r] + I[t, l]$$

where $M(t, l, b, r)$ is the sub-region of the matrix defined by (top, left, bottom, right) coordinates and I is the integral image, as shown in Figure 5. The use of integral images facilitates the computation of sub-regions' sum in $O(1)$ time complexity, with the trade-off of additional memory being used.

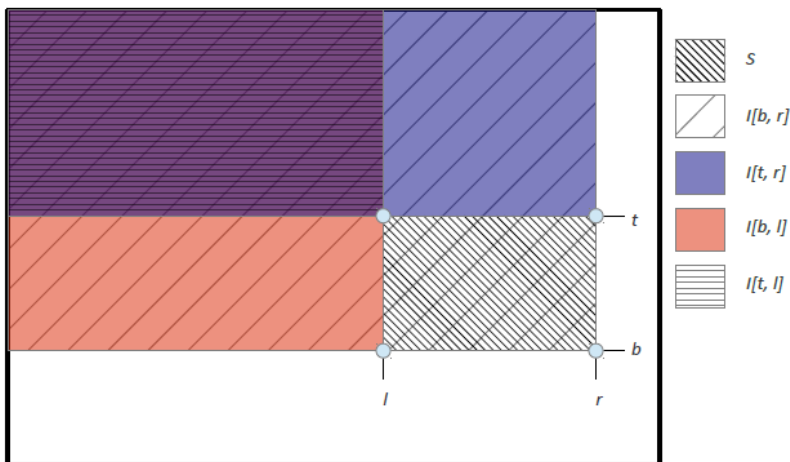


Figure 5. An integral image to compute the sum of elements in a sub-region of the matrix.

V. Depth Extractor Module

Having a 2D location of the object and a map with depth information as inputs, this module takes care of detecting the distance to the object on the third dimension. The output is a fully 3D localized object.

A depth map generated from a stereo-camera has shadowed areas where the camera could not see. These exist because some regions of the image are visible only to one of the two cameras (note that the working principle of the stereo-camera may vary). The first step in depth extraction is to filter out the shadowed areas. The filtered depth map is quantized in 5 distinct levels of depth by clustering using the k-means algorithm. Figure 6 shows the clustered depth map and the obtained 5 levels are given in Figure 7. The pixel from the center of the object (the center of the bounding box obtained in the previous step) is labeled under the nearest cluster centroid. The latter determines the depth distance of the object, as shown in Figure 8.

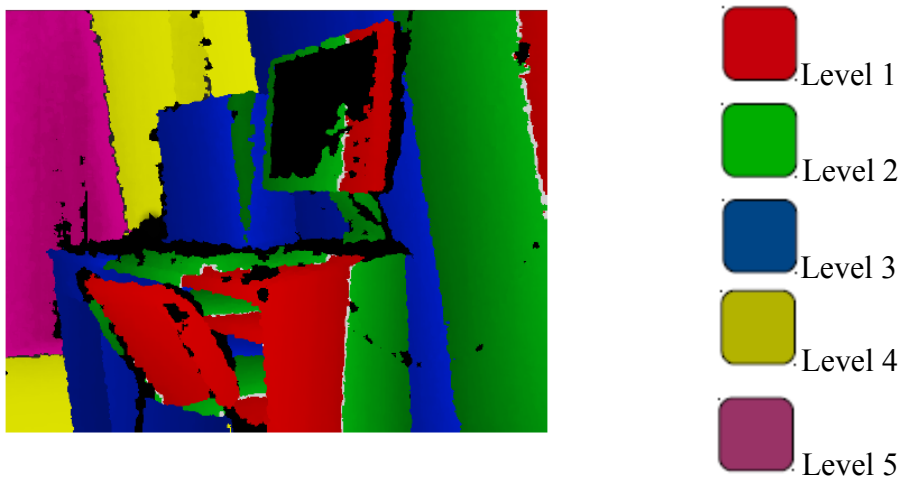


Figure 6. Clusters in the depth map

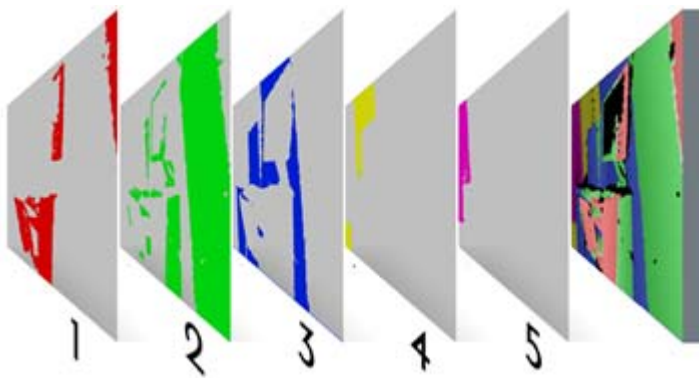


Figure 7. The 5 levels obtained using the k-means algorithm

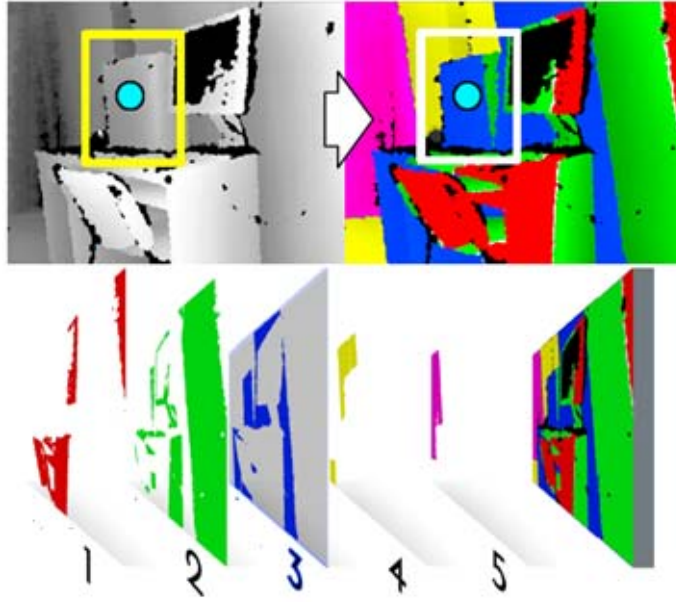


Figure 8. The centroid of the 2D bounding box

4. SYSTEM EVALUATION

External tools are used for features extraction. Multiple different descriptor types are available inside the present work: SIFT descriptors [Lowe, 2004], dense SIFT (which are SIFT descriptors extracted on a dense grid instead of detected keypoints) and SURF descriptors [Bay, 2006]. For the SIFT feature descriptors, the detector and extractor binaries provided by David G. Lowe are used. The SURF feature descriptors are extracted using the OpenCV. For the dense SIFT descriptors, the VLFeat open source library is used. For the support vector machine implementation, SVMLight is used, with command line callable binaries. SVMLight is a C-based tool developed by Thorsten Joachims at Cornell University [Joachims, 1999]. To extract the weights of the linear SVM from the model outputted by SVMLight, svm2weights tool [Cohen, 2011] is used. For training, for each class of objects, the current system uses sets of images for positive and negative examples of images. The positive image examples contain representative specimens of the class. The negative examples contain common backgrounds and environments where the objects are usually found. After feature extraction and encoding, the SVM is trained to discriminate between the positive and the negative features.

The main advantage of SURF over SIFT is speed. SURF descriptor is also smaller in size (64 integer values) which reflects in fewer dimensions feature space in the clustering and coding phases. The dense SIFT algorithm extracts SIFT descriptors from a dense grid, instead of detected key points. This ensures that features are extracted from places with low contrast, too. To extract SIFT descriptors from a dataset of images two implementations are provided: single threaded and multi-threaded. The running times of each one on the same dataset can be seen in the Table 1.

	Type of implementation	Running time
Fresh run	Non-threaded	643s
	Threaded	341s
Run with descriptors already generated	Non-threaded	100s
	Threaded	196s

Table 1. The running time

The system was trained on 3 classes of objects: laptop, mug and mouse (computer mouse). In Figure 9 there are some sample results for the class laptop. In Figure 9 c) only a fraction of the object is covered by the bounding box (about 50%). This doesn't raise a problem for depth detection in most cases, because the distance to the captured fraction of the object is usually a good approximation of the mean distance to the object. This kind of match is, also, good enough for extracting positional relations between objects. The images from Figure 9 a) and c) as examples, the relation between the laptop and the white plastic cup can be extracted with very similar precision in both cases a) and c). In case of Figure 9 d), the area that is captured by the bounding box is too large, and it includes other objects beside the laptop. This is a bad situation, both for depth extraction and for positional relating.

The ground truth bounding box is human drawn. The detected bounding box coverage in this case is of 72%. Figure 10 contains examples for laptop and mug localization.

Figure 11 presents a sample of localization with the ground truth also represented. One factor that decreases the coverage, in this case, is the perspective view of the object that enforces a larger bounding box (that includes larger non-object regions). Although the coverage percent is medium, enough pixels of the actual object are captured for a successful depth extraction. In the figure, the middle of the detected bounding box is represented as the blue dot. Since the depth level of the object is sufficiently represented as pixels on the depth map and since the middle pixel belongs to the object, the mean depth of the object is detected properly.



(a)



(b)



Depth: 93.4cm

(c)



Depth: 127.7cm

(d)

Figure 9. Results for laptop class



Figure 10. Object localization for laptop and mug objects

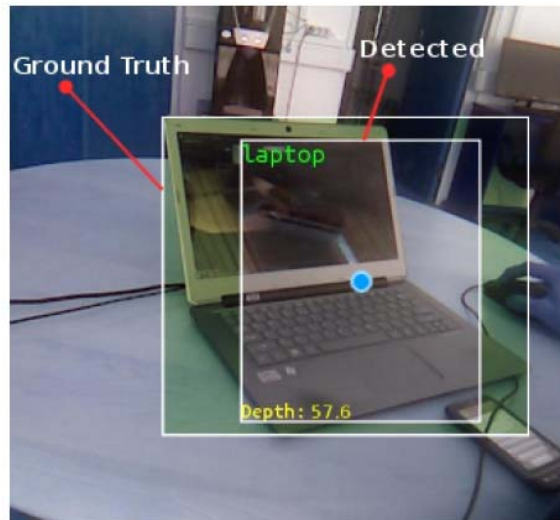


Figure 11. Average quality localization with ground truth manually determined. The cyan dot marks the middle of the bounding box.

Figure 12 presents two aspects. First, the extracted depth is almost the same, despite the difference in bounding box coverage percent. Second, the laptop object has a part with major changes in appearance between (a) and (b): the display. This shows how an unstable part of the object can radically change the detection result.

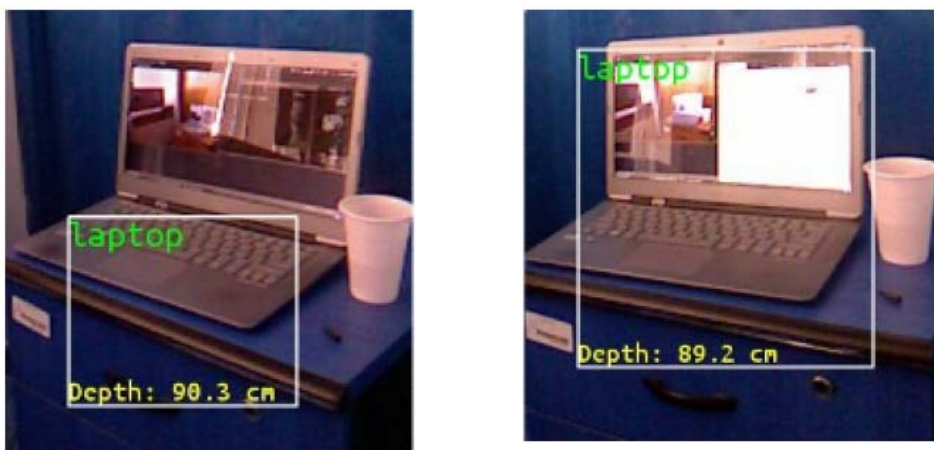


Figure 12. Detection example in case of changes in appearance

A sliding window approach for 2D bounding box object localization is intractable (having a time complexity of n^4 , where n is the length in pixels of one side of a square image) because it has to consider all the possible image sub-regions at any scale to be able to select the optimum. Heuristic restrictions can be added for sub-region candidates selection, but this would invalidate the optimal solution guarantee. The branch-and-bound driven sliding window approach converges to a global solution (that has the guarantee to be optimal) much faster. According to experiments in [Lampert, 2008], a maximum of $O(n^2)$ time complexity is obtained for a tight enough upper bound function.

The following time measurements were recorded on a Ubuntu Linux laptop computer with the hardware configuration: Intel(R) Core(TM) i5-2467M @ 1.60GHz dual core CPU with HyperThreading(TM), 2GB of RAM DDR3 memory, 5400RPM hard-drive. On a dataset of 43 positive examples and 51 negative examples of an average size of 600 x 400px each, the training stage runs in 105 seconds when SIFT features extractor is used. The longest running module in the training stage is the features clustering and codebook creation component. The second most time consuming is the feature extractor. On the testing stage, the processing power of the system is a frame each 7 seconds, for one object being localized. For more objects, the time increases linearly.

5. CONCLUSION AND FUTURE WORK

The paper describes a general purpose object location extractor that can be trained on any object class by giving it positive and negative examples of objects. Hence, it can be used in many domains. The limitation to its discriminative power is set by the representation model. The present representation is the bag-of-visual-word, which treats features in an unordered manner. The depth detection process is insensitive to a certain degree to the quality of the object 2D bounding box produced by the object detection module. It can overcome a bounding box coverage percent of 72%. The system can process one frame every 14 seconds. This performance suffices for the AmI application, where the status of the surveyed person is checked periodically. Better performance may be achieved by using depth information in the 2D localization process. This would require training data to be augmented with depth maps, though, which is much harder to obtain and increases the complexity of the training dataset creation process. This led to the decision of limiting the 2D localization process to use only intensity maps from images. One improvement that can be made is the use of Spatial Pyramid Matching (SPM) [Lazebnik, 2006] for scoring matched features. SPM uses multiple levels of exponentially denser grids to generate histograms of features. Thus, it retains spatial information, while the bag-of-visual-words discard it.

The system described in this paper can be integrated into an ambient intelligence application for supervising people. Consider a system that periodically checks the status of a surveyed person. This system is mainly concerned with health related issues and its aim is to prevent danger and alert the human personnel when critical situations appear. Such a system would consist of the following components: (1). Object recognition and localization module, (2) Human subject localization and posture recognition module, (3) Module that extracts semantic relationships between the recognized objects, (4) Decision taking module.

6. REFERENCES

- [Bay, 2006] H. Bay, T. Tuytelaars, and L. J. V. Gool. Surf: Speeded up robust features. In A. Leonardis, H. Bischof, and A. Pinz, editors, ECCV (1), volume 3951 of Lecture Notes in Computer Science, pp. 404–417. Springer, 2006.
- [Bosch, 2006] A., Bosch, A. Zisserman, and X. Munoz, “Scene classification via pLSA”, Proc. 9th European Conference on Computer Vision (ECCV’06) Springer Lecture Notes in Computer Science 3954, pp 517- 530, 2006.

- [Bosch, 2007] A., Bosch, A. Zisserman, and X. Munoz, “Image classification using random forests and ferns”. Proc. 11th International Conference on Computer Vision (ICCV'07) (Rio de Janeiro, Brazil), pp. 1-8, 2007.
- [Cohen, 2011] Ori Cohen. Compute the weight vector of linear SVM based on the model file [Python program] . 2011. <http://oricohen.com/dev/2011/05/19/svmlight-a-python-script-compute-the-weightvector-of-linear-svm-based-on-the-model-file>
- [Csurka, 2004] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints”, Proc. ECCV'04 International Workshop on Statistical Learning in Computer Vision (Prague, Czech Republic): pp.1-22, 2004.
- [Felzenszwalb, 2008] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. CVPR, pp. 1-8, 2008.
- [Joachims, 2009] T. Joachims, T. Finley, and C. N. Yu. Cutting-plane training of structural SVMs. Machine Learning Journal, 77(1), 2009.
- [Lampert, 2008] C. H. Lampert, M. B. Blaschko and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. Proc. IEEE CS Conf. Computer Vision and Pattern Recognition, pp. 1- 8, 2008.
- [Lazebnik, 2006] D. Ramanan and C. Sminchisescu. Training deformable models for localization. Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 1, New York, NY, pp. 206–213, 2006.
- [Lowe, 2004] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. IJCV, 60(2):91-110, 2004.
- [Queen, 1967] MacQueen, J. B. Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1. University of California Press. pp. 281–297. MR 0214227.Zbl 0214.46201. 1967.
- [Yu, 2009] T. Joachims, T. Finley, and C. N. Yu. Cutting-plane training of structural SVMs. Machine Learning Journal, 77(1), 2009.