Electronic Letters on Computer Vision and Image Analysis 12(2):78-87; 2013

Detection and Classification of Multiple Objects using an RGB-D Sensor and Linear Spatial Pyramid Matching

Michalis Dimitriou[*], Tsampikos Kounalakis[**], Nikolaos Vidakis[*] and Georgios Triantafyllidis[***]

* Applied Informatics and Multimedia Dept., Technological Educational Institute of Crete, Heraklion, Greece

** Dept. of Electronic and Computer Engineering, School of Engineering and Design, Brunel University, London, UK

*** Medialogy Section, Aalborg University Copenhagen, Denmark

Received 5th Nov 2012; accepted 4th Mar 2013

Abstract

This paper presents a complete system for multiple object detection and classification in a 3D scene using an RGB-D sensor such as the Microsoft Kinect sensor. Successful multiple object detection and classification are crucial features in many 3D computer vision applications. The main goal is making machines see and understand objects like humans do. To this goal, the new RGB-D sensors can be utilized since they provide real-time depth map which can be used along with the RGB images for our tasks. In our system we employ effective depth map processing techniques, along with edge detection, connected components detection and filtering approaches, in order to design a complete image processing algorithm for efficient object detection of multiple individual objects in a single scene, even in complex scenes with many objects. Besides, we apply the Linear Spatial Pyramid Matching (LSPM) [1] method proposed by Jianchao Yang et al for the efficient classification of the detected objects. Experimental results are presented for both detection and classification, showing the efficiency of the proposed design.

Key Words: Depth Map, Object Detection, Microsoft Kinect, Image Segmentation, Feature Extraction, Classification, Linear Spatial Pyramid Matching.

1 Introduction

In biological vision, humans are able to see and identify multiple objects in a scene. In computer vision the past few years we have seen great progress and promising results for object detection and classification. However, most algorithms do well in classifying the dominant object in a scene, but fail when multiple objects need to be classified in a single image. Our proposed system takes advantage of the new technology

Correspondence to: < gt@create.aau.dk>

Recommended for acceptance by < Angel Sappa>

ELCVIA ISSN: 1577-5097

Published by Computer Vision Center / Universitat Autonoma de Barcelona, Barcelona, Spain

of simple and real-time depth map generation in combination with effective image processing techniques and classification algorithms to efficiently solve the problem of complex scene analysis and object classification. We have designed a detection system that uses the depth information of a scene provided by the Kinect sensor [2], which is using a combination of an IR light projector and a simple camera to produce an RGB plus Depth image pair. In this context we detect the different objects in a scene and then we segment the RGB image into several isolated object images, which can be classified more efficiently with the use of the Linear Spatial Pyramid Matching (LSPM) classification algorithm. The key to our detection method is that we use edge detection algorithms [3] directly on the depth image to detect sharp changes of depth instead of sharp changes in luminosity. This method of detection is fast, accurate and has many advantages over traditional object detection.

The rest of the paper is organized as follows: in Section 2 we analyse the steps we take for object detection. Section 3 explains the classification algorithm used in our system, while Section 4 presents the dataset that was created for testing. In Section 5 we will show some results for both object detection and classification. Finally, we will close with our conclusions and some thoughts on the possible future work to be done in Section 6.

2 Object Detection

The proposed system design for multiple object detection in a 3D scene is divided in four basic sections illustrated in Figure 1 (with different colors). Each section is analyzed in subsections represented as diagram blocks in the same figure.

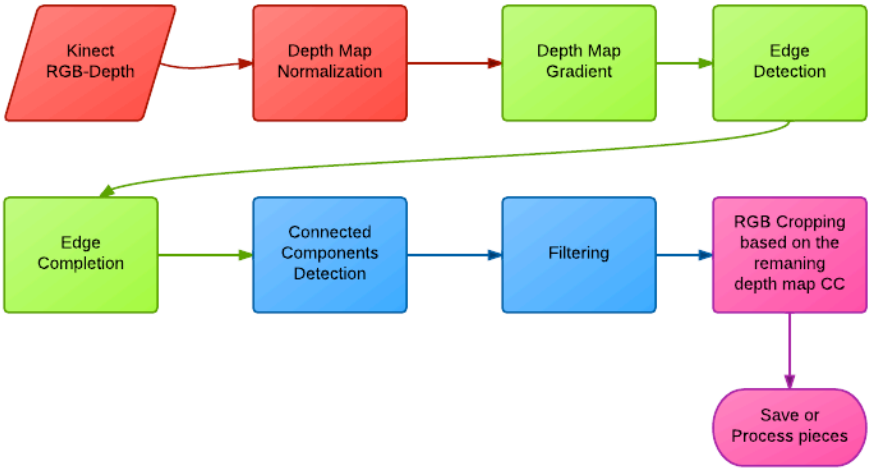

Figure 1. Flow diagram the proposed detection system

2.1 Acquisition and pre-processing

The first step is to capture an RGB image and the corresponding depth image from an RGB-D sensor such as the Kinect sensor. This in our system is achieved by using the OpenNI drivers [4] and Kinect – Matlab MEX functions [5]. After capturing these images, we perform a simple pre-processing to the depth image, targeting to solve a specific problem of the captured depth image.

The original depth image returned by RGB-D sensors contains some regions with no information about the depth of that region. This is because the IR light does not reflect well on all surfaces. Also Kinect is designed to measure distances ranging from 60cm up to several meters. Any obstacles located out of that range will be

assigned the depth value zero. These areas will produce misleading results if not corrected. To this goal, we created a fast algorithm that that process the captured depth images and replaces zero values of depth with the nearby non-zero values and by that way we eliminate that problem. This process we call depth image normalization. Figure 2 shows an example.

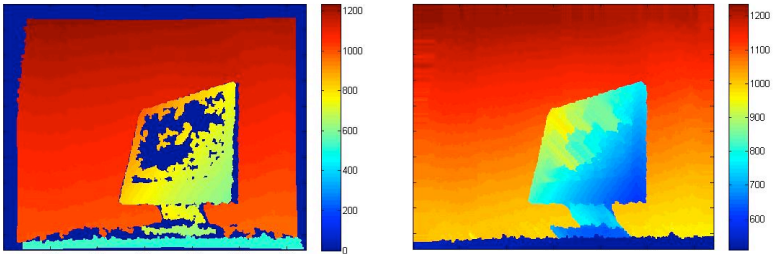

Figure 2. Depth image before and after depth normilization.
Notice that in the second image there aren't any zero values.

2.2 Depth map analysis

The normalized depth image is processed with convolution by a two dimensional Gaussian filter to produce the gradient image where edges of the objects have very large values and all other area having very small near zero values. Image edge detection without the use of a depth map is of course possible: using the gradient of the RGB image. But since RGB edges appear where luminosity changes sharply, RGB edges may not always mean different objects, but just sudden changes in the colour of the same object, resulting in edges that do not represent the real objects boundaries. On the contrary, depth images captured by Kinect are obviously unrelated to luminosity changes and this is why the image edge detection algorithm to the depth image is much more robust (compared to the RGB image) in terms of the object detection.

Then a threshold should be defined to adjust the sensitivity of this edge detection algorithm. This threshold is actually based on an estimation of the number of the objects that we have in our scene, since if we use high threshold value, we will end up missing important parts of the edges, and if the threshold is too low, a lot of unimportant detail edges will make the image useless.

In the resulting image of the edge detection of depth map, small fragments of the edges might be missing. In order for the connected components algorithm to work we need the boundaries of the object to be solid separating the different objects. At this point we use the simple morphological closing algorithm [6], which is using a structuring element to complete the almost completed lines (see Figure 3). The depth image is then ready for the application of the connected components algorithm [7].

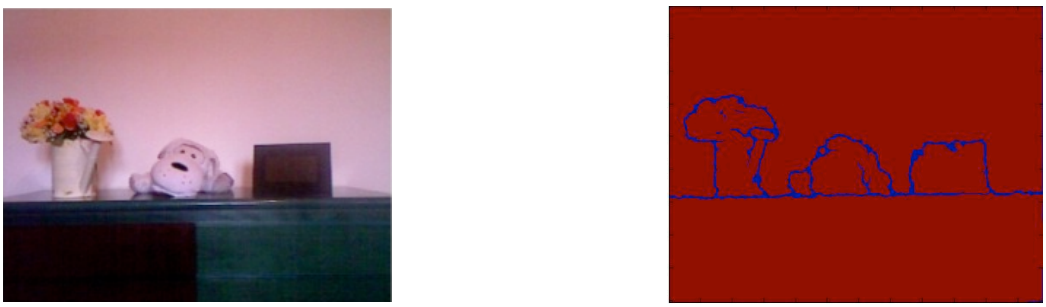

Figure 3. The objects edges after the edge detection and the closing algorithm

2.3 Object detection and filtering

The connected components algorithm [7] returns the pixel indexes for each connected component image region (see Figure 4), along with some useful information for each object such as the area of each object in pixels, the extrema points, bounding box for each component etc.

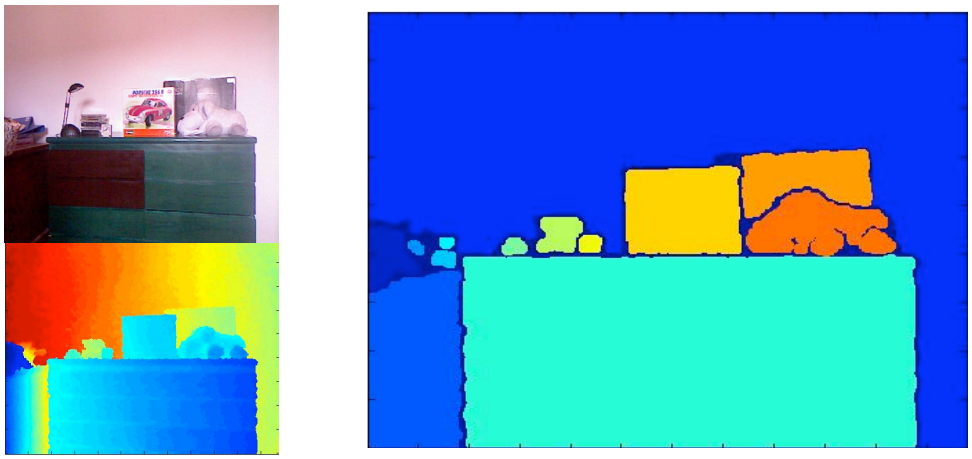

Figure 4. Connected components filled with diferrent colors

The next three steps are used to filter out "bad components" and exclude them from the list of component we will use to crop the RGB image. There are three phases of this filtering; phase 1 will filter objects based on their relative size, phase 2 will filter based on their width or height in pixels and phase 3 will filter the components considered to be background. Of course, each of these filters might be modified or omitted depending on the user needs.

We will now explain how each of these phases is performed in our system (see also Figure 5). At phase 1, we check for image components (segments), of which the area (in pixels) is within the range between the minimum and maximum area allowed. The minimum and maximum area thresholds are defined heuristically depending on system's needs as a percentage of the original RGB image area. For example if an object is larger than 50% of the RGB image or smaller than 1%, will not pass to the next phase. At phase 2 of filtering, only the components that don't expand all the way in width or height of the original RGB image will be forwarded to the next phase. Finally phase 3 will discard any object considered to be background, leaving only non-background components to proceed further. An object is considered as background if the corresponding connected region belongs to an area with depth values close (defined by the user) to the maximum depth value of the depth map.

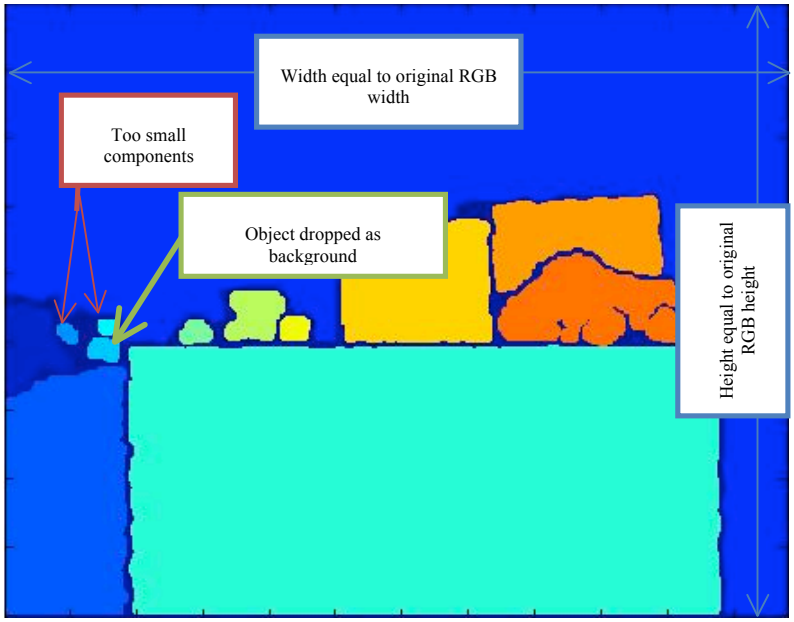

Figure 5. Components that didn't pass the filters.

2.4 Image cropping

Any components made it through the filters are cropped from the original RGB image using the corresponding bounding box of the component. We can optionally set to zero all pixels in the bounding box that doesn't belong to the object using the object properties we have.

3 Object classification

3.1 Background work in object classification

Image classification is a well-established problem in computer vision. The complexity of this field is due to the number of different types of images, such as scenery images, face images, object images. In our paper we deal with the problem of object classification.

Objects are usually treated as individual images that pass through a classifier for identification. The problem usually is the representation of those images. The representation has to integrate information about the image in a way that there will be similarities with other images of the same class (same or similar objects). Also the representation has to be "cooperative" with a classifier, so the classifier will provide good results with small processing cost. Because of the multitude of the object, that has as a consequence the multitude of images, an object classification system has to cope with that load.

Object classification systems usually use "gradient features" for representation, such as the well known SIFT[8] and SURF[9] algorithms; although the use of just the descriptors provided by those algorithms is not capable of achieving high rates of recognition. The first good results were provided by the Bag Of Words method (BOW) (or Bag Of Features method). This method represents normal features in relation with a "visual codebook" that provides general information about all the known classes in a system. The advantage of this method is the low complexity of the classifier. The biggest improvement of that algorithm was presented in 2006 by Svetlana Lezebnik with Spatial Pyramid Matching algorithm (SPM) [10]. This method was a combination of bag of words with the pyramid matching scheme of Grauman and Darrel [11]. The method deals with the bag of words representation in a combination of different spatial levels. The advantages of this method is more robust representation of images, even less complexity than the classic BOW method, and better results.

3.2 Linear Spatial Pyramid Matching

Linear Spatial Pyramid (LSPM) [1] is an extension of the classic SPM (see Figure 6). For both methods the descriptors of each image are extracted from a dense spatial grid. In classic SPM a codebook is generated using K-means on randomly selected SIFT descriptors. But in LSPM this codebook generation is done by optimizing the K-means algorithm with L1-minimization. Every descriptor is quantized with the codebook; this process gives the relationship between the descriptor and the codebook. Each code represents the combination of the contents of the codebook to form the descriptor. In the case of LSPM the algorithm used is L1-minimization and the codes that are generated are sparse. With the use of a pyramid match kernel we can make combinations of those codes. A pyramid matching kernel divides the image in spatial regions and into levels. Each region provides an overall code that will be used in the representation of the pyramid. At the classic, and most used, form of that pyramid matching kernel the first level is the combination of all the codes in an image. The second layer is divided in four regions and the combinations now refer only to the codes of that region. The third layer is a finer layer with sixteen regions that have the same effect as the second layer. So we have a representation of combinations for each sparse code in the image. The final section of LSPM is a linear SVM which is responsible for the classification. Sparse coding improves the former algorithm by providing linear image representation with L1-minimization. The results are improved and the complexity on classification drops even more because of the linear classifier used. The novelty of our proposed system does not lie on a major change in the method of LSPM, but in the fact that it was never used before in three-dimensional extracted data.

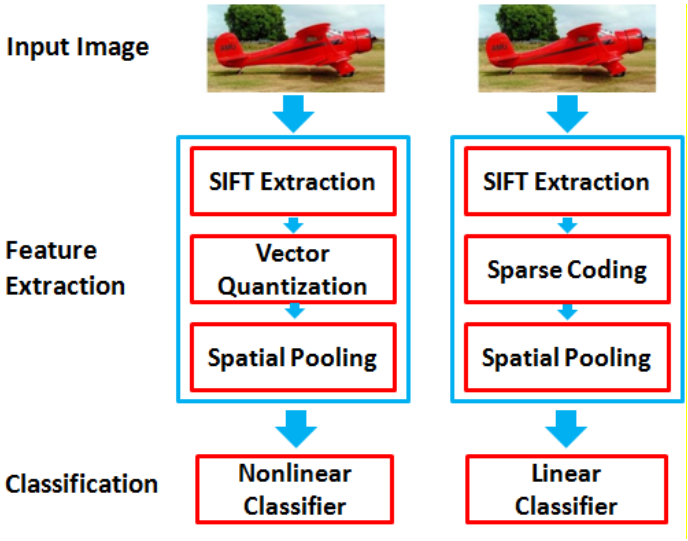

Figure 6. Schematic comparison of the original nonlinear SPM (a) with linear SPM (b) based on sparse coding. The underlying spatial pooling function for nonlinear SPM is averaging, while the spatial pooling function in LSPM is max pooling [1].

4 Dataset of multiple objects in 3d scenes

For the experimental results of this paper a dataset of images has been created that contains 10 classes of objects with 10 instances (images) in each class. The object images are extracted from complex scenes using the detection method explained in section 2. Figure 7 shows some examples. All the scenes are indoors, since Kinect has a limited distance range and they contain mostly household and office objects.

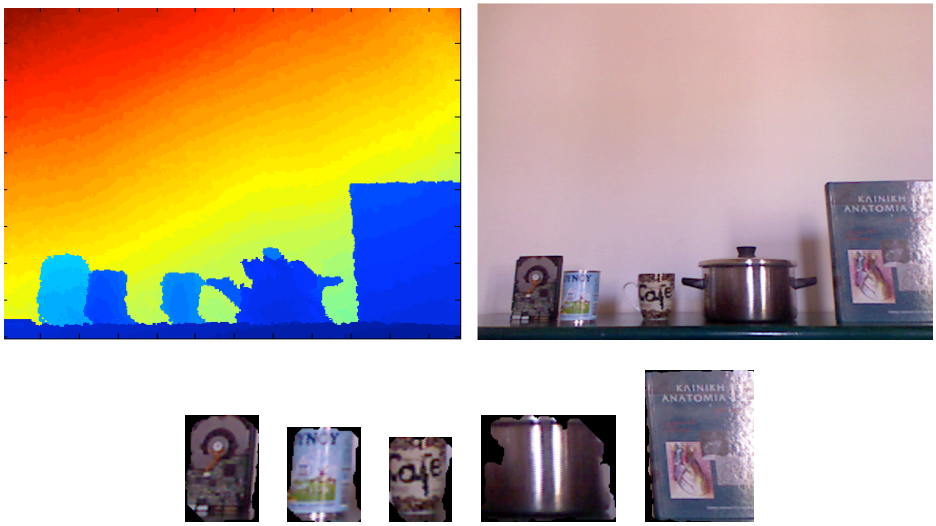

Figure 7: Object detection for database creation

5 Experimental Results

For our experiments we used an Intel I5 laptop with 4GB ram, running Microsoft Windows 7 OS and Mathworks Matlab. For the detection and the classification algorithms to work, various thresholds had to be set in order for the system to run properly. Table 1 shows the parameter values chosen for the detection algorithm, while Table 2 shows the settings chosen for the classification algorithm in our configuration.

Property description	***Chosen value***
Sensitivity for the edge detection	Adjusts automatically based on max depth
Structuring element used	Disk with 6 pixels diameter
Maximum object area	1/2 of the original image
Minimum object area	1/800 of the original image
Background	90% of the maximum depth
Drop object as background	If 30% is background
Allow object width/height equal to original	False
Black or whole background in bounding box	Black

Table 1: Detection algorithm parameters

.

Property description	***Chosen value***
SIFT descriptor extraction grid spacing	6
SIFT patch size	16
Training number of bases	1000
Training number of samples	500
Beta regularization	1e-5
Dictionary training epochs	10
Pyramid	[1, 2, 4]
Gamma	0.20
Random tests	30 rounds
Lambda regularization	0.1
Training number per category	7

Table 2: LSPM classification algorithm parameters

In Figure 8 we illustrate the objects detected by the proposed system in one scene of the database which consists of the initial RGB image, the normalized depth image, the connected components image and the detected objects. The computational time needed for the detection is 0,3Sec for each scene.

Table 3 shows the classification results using a test dataset that consist of 100 images in 10 different categories which are: Spray cleaner, book, bottle, hard disk, box, can, pot, mug, shampoo and shoe. All the images have been acquired using our detection system. The time required for the classification of each object is 5ms.

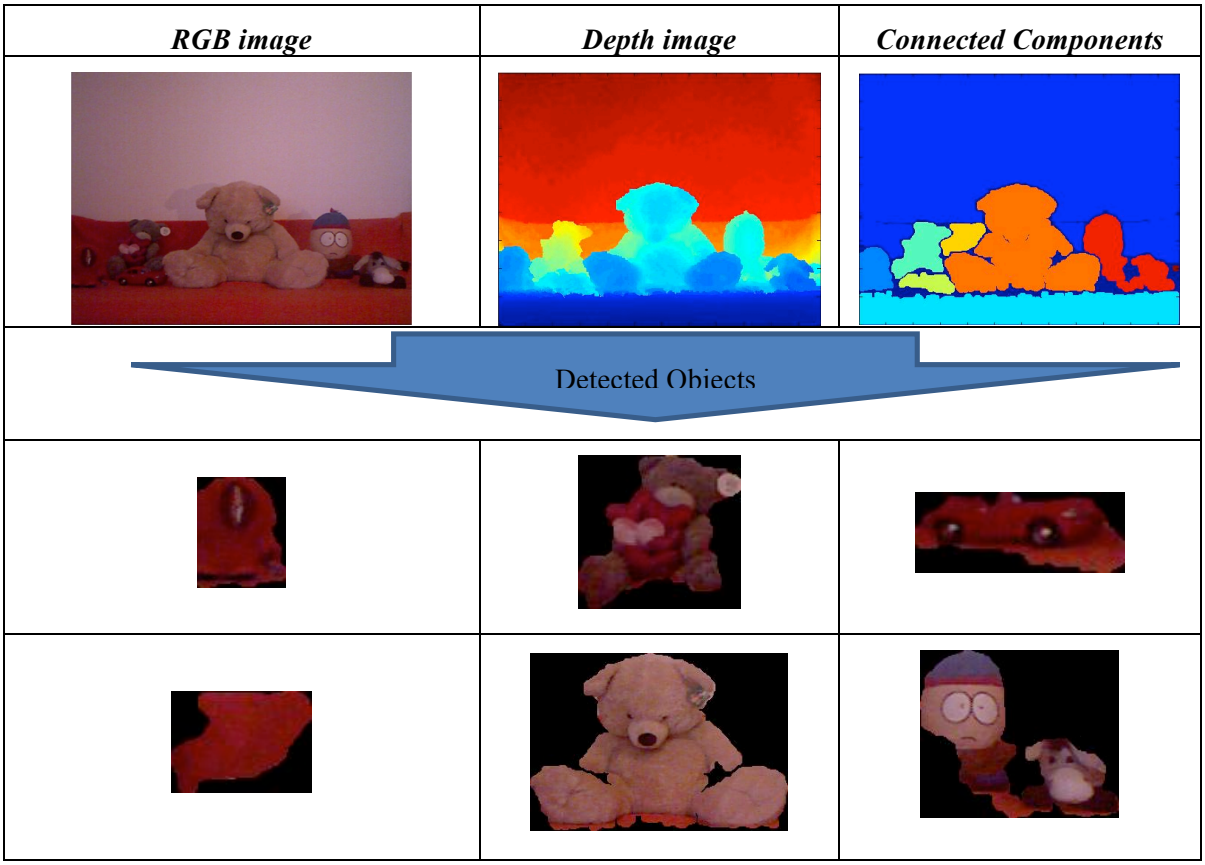

Figure 8. Scene with multiple objects detected.

Num. of classes	*Objects in each class*	*Used for training*	*Used for testing*	*Rand. Rounds*	*Mean classification percentage*
10	10	70%	30%	30	84.33%

Table 3: Classification Results

6 Conclusion & Future work

We have proven that using an RGB-D sensor such as the Microsoft Kinect sensor for depth and RGB image acquisition may lead in impressive results in object detection and classification. Object detection with this method is fast and accurate and leads to better classification of multiple objects from just one scene. Future work will be focused on a bigger dataset for classification , improvement of the detection method and classification comparison of more state of the art classification algorithms. We may also implement object tracking to detect multiple moving objects in a scene.

.

References

[1] Jianchao Yang, Kai Yu et. al. Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification, CVPR 2009

[2] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake, "Real-Time Human Pose Recognition in Parts from Single Depth Images", 24th IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, USA, June 20-25, 2011.

[3] N. Senthilkumaran, R. Rajesh, "Edge Detection Techniques for Image Segmentation – A Survey of Soft Computing Approaches", School of Computer Science and Engineering, Bharathiar University, Coimbatore.

[4] OpenNI framework and NITE middleware http://openni.org/, page accessed 19/10/2012

[5] Kinect for MATLAB, http://sourceforge.net/projects/kinect-mex/ , page accessed 19/10/2012

[6] Mathworks Matlab imclose() function, http://www.mathworks.com/help/images/ref/bwconncomp.html, page accessed 19/10/2012

[7] Mathworks Matlab bwconncomp() function, http://www.mathworks.com/help/images/ref/bwconncomp.html, page accessed 19/10/2012

[8] Lowe, David G. Object recognition from local scale-invariant features. Proceedings of the International Conference on Computer Vision. 2. pp. 1150–1157. doi:10.1109/ICCV.1999.790410

[9] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346--359, 2008

[10] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In CVPR, 2006

[11] K. Grauman and T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Beijing, China, October 2005