Automatic, Real Time, Unsupervised Spatio-temporal 3D Object Detection Using RGB-D Cameras

Manal H. Alassaf[1,2], Kamran Kowsari[1], James K. Hahn[1]
[1]Department of Computer Science, Insitiute for Computer Graphics,
The George Washington University, Washington DC, U.S.A,
[2]Department of Computer Science, Taif University, Taif, Saudi Arabia
e-mail: MAlassaf@gwmail.gwu.edu, Kowsari@gwu.edu, Hahn@gwu.edu

***Abstract*— The evolution and affordability of depth cameras like Microsoft Kinect make it a great source for object detection and surveillance monitoring. Information available from depth cameras includes depth in addition to color. Using depth cameras, the provided depth information can be incorporated for object detection in still and video images, but needs special care to pair it with color information. In this work, we propose a simple, yet novel real time unsupervised object detection method in spatio-temporal videos. The RGB color frame is mapped into HunterLab color space to reduce emphasis on image illuminations, while the depth frame is back-projected into the 3D real world coordinate in order to distinguish between objects in space. Once combined, the mapped color information and the back-projected depth information are fed into automatic, unsupervised clustering framework in order to detect scene objects. The framework runs in parallel to provide real time spatio-temporal object detection.**

Keywords- Spatio-temporal, Object detection, Lab color space

I. INTRODUCTION

Automatically analyzing events and detecting objects in videos is not a trivial task [1, 2]. Challenges with automatic object detection include, changes in camera location, changes in scene illumination throughout the video frames, changes in the scene viewpoint, finding good cues to distinguish and detect the objects, introduction of objects occlusion or deformation, and the huge search space for tracking the detected objects bounding boxes over the video frames, among others.

Partitioning a sequence of images into coherent regions using motion cues and image features is referred to as spatio-temporal segmentation. Advances in two algorithms are key reasons behind the recent progress in spatio-temporal recognition [1], namely the optical flow [3] and feature pooling methods [4]. The techniques that are used to spatio-temporally detect objects in videos can be categorized into three broad categories. The first category includes generalization of graph-based image segmentation into temporal segmentation [2, 5, 6]. However, a general drawback of this category is the graph size, which depends on the number of pixels in the video. The second category includes global motion models that segment the motion in layers where at each layer the motion can be recognized [7-9]. Such techniques can fail with a lack of texture information. The third category depends on linking point trajectories in sparse number of locations [10-13] using algorithms like optical flow for example. The drawback of this technique appears when there are static objects that do not move throughout the video. Among these three broad categories reside supplementary techniques like spectral clustering [14, 15], Fisher vector [16, 17], and supervoxels and hierarchical image segmentation [5, 15, 18-20].

Viewed from an alternative perspective, some literature focuses on spatio-temporal clustering that groups objects based on their spatial and temporal similarities. This kind of research has been applied to data that includes geographic environmental information or weather data clustering over time, such as in [21, 22]. In these techniques, a baseline dataset is obtained by which to compare subsequent datasets in order to identify significant changes in spatio-temporal clustering. However, these spatio-temporal clustering techniques have not been employed for the purpose of object detection in video frames.

Among the previously discussed techniques, several have been developed to detect objects in still images, but are useful when extended to detect objects in videos [1]. Some techniques involved video clips that were carefully cropped around interested action [1]. Other techniques can be applied to videos in a spatio-temporal manner but require costly computation. Most of the time, when machine learning techniques are employed in spatio-temporal detection, entire video frames are needed in the training stage to learn from, and to tune the machine learning algorithm parameters. In this work, the proposed 3D object detection relies on online, unsupervised learning algorithm; k-means [23], where the frames are clustered as we receive them and results are immediately provided. Our algorithm firstly pre-processes the Kinect input frames by back-projecting the mapped depth frame with the color frame into 3D world space, and converting the *RGB* color frame provided by Kinect into *HunterLab* color space [24-26], resulting in 3D colored cloud points. Subsequently, the cloud points' normal vectors are calculated. Finally, the cloud points are fed into the clustering algorithm to detect different objects in the scene based on three prepared features of the points: position, color and normal vectors. This process is performed without the need to compute the optical flow on each frame or track the bounding boxes of the objects per frames. In addition, the

1550-6037/15 $31.00 © 2015 IEEE
DOI 10.1109/iV.2015.80

whole video frames are not needed and the detection is done in real time.

II. METHODS

A. Input

Kinect camera version 2 provides the input as color frame and depth frame, with 30 frames per second (fps) rate, albeit with the different resolutions, 1920×1080, for the color frame; and 512×424, for depth frame. After aligning the two frames, cloud points data $p_{i \to 0:n}$ can be obtained where each point is described with 6 parameters: 3 parameters, *x y z*, for the position and 3 parameters, *R G B*, for the color:

$$p_i = (x \;\; y \;\; z \; R \;\; G \;\; B)^T \tag{1}$$

We preprocess the input in such a way that allows us to cluster the scene into a number of distinct objects. The three steps involved in preprocessing are: back-projecting the points into 3D world space, mapping the color into *HunterLab* space (which we refer to as *Lab* in the rest of the paper), and finding the 3D points normal vectors.

B. Preprocessing step

1) *Back-projecting the depth frame into 3D world coordinate:* The Kinect input *x*, *y*, and *z* parameters are actually the color frame or the image 2D coordinates *x* and *y* along with the depth frame *z* data associated with them. To get an accurate 3D position of each point in the color frame, the 2D position is back-projected using the *z* data and the Kinect camera intrinsic parameters to obtain the correct cloud points 3D positions in the real world coordinate. Kinect captures depth in the range 0.4m to 4.5m. Points with no depth information are discarded and not included in the cloud points.

2) *Converting the color frame into Lab color space:* The Kinect color data is provided in an *RGB* format which is not perceptually uniform. This nonuniformity indicates that the *RGB* color is affected by environmental illumination, so that there is variation in the components values of two points having the exact same physical color. This variation depends on the points locations with respect to the light source in the scene. Thus, *RGB* is not suited color space to determine the differences between colors. Furthermore, perceived color differences is disproportional based on Euclidean distance between two *RGB* colors. To overcome this problem, *RGB* colors are mapped into the *Lab* color space. *Lab* or originally *HunterLab* color space [24], having its final formula released in 1966 by scientists who worked on uniform color space, is based on the color-opponent theory. Here, the dimension *L* is for lightness, and dimensions *a* and *b* are for the color-opponent. *L* represents light vs. dark, *a* represents red vs. green, and *b* represents yellow vs. blue as shown in Fig. 1. One can think of *a* and *b* coordinates as indirectly reflect hue and chroma but are difficult to interpret separately. This color space is visually uniform [27]. The problematic emphasis of the illumination that we have with *RGB* representation is reduced with this *Lab* depiction, whereas perceptual uniformity sought by *Lab* color space deals more effectively with high illumination variability [28, 29]. Therefore, the Euclidian distance between two different colors represented in *Lab* is proportional to the different between them perceived by the eye [30]. The mapping from *RGB* to *Lab* is done through intermediate system *XYZ* derived from *RGB* using the following equations [24]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2}$$

$$L = 10 * \sqrt{Y} \tag{3}$$

$$a = 17.5 * \frac{(1.02*X - Y)}{\sqrt{Y}} \tag{4}$$

$$b = 7 * \frac{(Y - 0.847*Z)}{\sqrt{Y}} \tag{5}$$

After this mapping, each point in the point cloud data $p_{i \to 0:n}$ is described as in (6):

$$p_i = (x \;\; y \;\; z \; L \;\; a \;\; b)^T \tag{6}$$

3) *Normal calculation:* Every point in the points cloud can be described using 9 parameters by adding the normal vector ($n_x \;\; n_y \;\; n_z$) to the description of p_i as in the following vector:

$$p_i = (x \;\; y \;\; z \; L \;\; a \;\; b \; n_x \; n_y \; n_z)^T \tag{7}$$

Normal vectors are calculated by using the neighboring information in the 3D cloud of points.

C. Clustering step

The aim of this step is to divide the points in the cloud into different clusters; each cluster represents an object in the scene. Each clustering technique depends on similarity measure between the input instances that allow for classification [31]. While more information about the points is provided to us using Kinect, e.g. the color and the normal, we extend the clustering algorithm similarity measure by considering the color difference and the angular difference between the points' normal vectors in addition to the Euclidean distance between them.

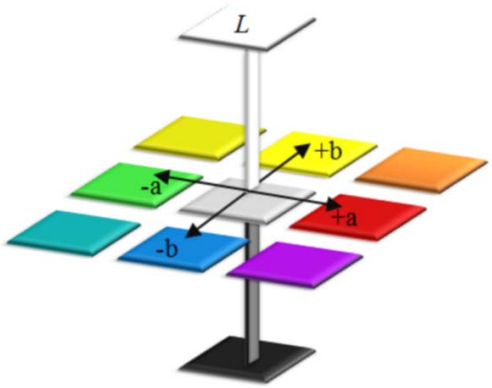

Figure 1. *Lab* color space illustration.

One of the well-known clustering algorithms is k-means. The k-means clustering algorithm is an unsupervised learning approach used to classify the points into different clusters. Initially, each cluster is represented by a center, also called a seed, and the initialization of these centers affects the algorithm final results. Having the clusters initial centers, each point in the cloud is assigned to the cluster with the closest center using a specific similarity measure like Euclidian distance. Then, the center positions are re-calculated by averaging the points that were assigned to their clusters. The process is iterative until the centers are stabilized.

In 2007, a smart automatic seeding technique, called k-means++ that depends on point distribution was proposed by [32]. By using k-means++ algorithm, the first of the k initial centers is randomly chosen from points cloud. Each remaining initial center is chosen with probability proportional to its contribution to the overall clustering using the set of chosen centers up to this point. More information about k-means++ algorithm can be found in [32]. We use k-means++ only to automatically obtain the centers from the first frame of Kinect input at the beginning.

Euclidian distance between points is a very useful similarity measure in many clustering algorithms including k-means. To consider the color difference of these points while clustering each frame cloud points, the *Lab* value is incorporated in the Euclidian distance between any two points p_i and p_j as defined in (8):

$$d(p_i, p_j) = \sqrt{\begin{matrix}(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \\ +\alpha^2\left((L_i - L_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2\right)\end{matrix}} \quad (8)$$

Where the scale α insures that geometric distance and color distance between two points are in the same order of magnitude.

We then define our new hybrid similarity measure as a function f of two terms; one for the Euclidian distance, d, between two points p_i and p_j and the other for the angle, θ, between the normal vectors of p_i and p_j, as described in (9):

$$f(p_i, p_j) = d(p_i, p_j) + \lambda\left(1 - \cos\theta(n_i, n_j)\right) \quad (9)$$

The value of f is increased when the value of the second term increases, as a result of the angle θ increase. Consequently, only d affects the value of f as a result of any two points having the same surface normal, as this causes the second term to become zero. The λ parameter is a weighting factor controls the effect of the angle θ, λ value is determined according to the features of objects inside the scene. This proposed function f satisfies the properties of a distance function in a metric space, as proven in Appendix 1.

By having the first frame initial centers and using this hybrid similarity measure as described above, we start the k-means clustering process. Out of the 30 fps provided by Kinect, we use the first frame of each half second and discard the rest, resulting in utilizing 2 fps: frame #1 and #16. In the first frame of first second, using the automatically suggested centers by k-means++ algorithm, we cluster the cloud points of the first frame and find the position of the new centers by averaging individual cluster points. The new centers are carried into the next selected frame at half second step to cluster its points, and the same process is repeated to the last second. The resulted clusters represent 3D objects in the scene. Eventually, each cluster contains points of object that have similar color, normal vector, and position in the scene. The results are obtained in real time, without the need to have all the frames beforehand to cluster one frame objects.

To obtain real time feedback using CPU computation, parallel processing is employed. In the proposed system, we reduce the computational time from 2 minutes per frame by using single-thread, to 2 fps by using the multiple threads. Hence, parallel processing is used as multiple threads algorithms. More specifically, several threads are used for creating the cloud points, calculating the points' normal, mapping to *Lab* color space, and clustering using k-means algorithm. This is done by using Microsoft API of multi-thread computation which is useful for parallelization of loops [33]. The used hardware is CPU: Intel Xenon E5 2630 (15 MB Cache, and support 12 thread, 6 cores) with 32 GB RAM.

III. Results and Discussion

To demonstrate the power of the proposed hybrid similarity measure, initial experimentation is done on a 3D cube cloud of points captured by Kinect. This captured cube has a different color for each face. We used k-means clustering algorithm to test the proposed similarity measure. The k-means centers were initialized by k-means++ algorithm. Incorporating the color information and the normal vectors information enhanced the clustering such that each resulting cluster's points have similar color and normal vectors as shown in Fig. 2 in addition to be placed closely in space. Therefore, we can say that the best clustering which clusters the scene's points into meaningful objects and distinguishes their points based on their similarity was achieved by using proposed similarity measure of points' position, color and normal vectors. The difference between clustering by considering the position and color (Fig. 2.e) verses considering the position, color and normal vector (Fig. 2.f) in the cube example is subtle because the points that have one color have very similar normal vectors (points of one face of the cube).

In the second experimentation, we captured the Kinect video frames in real time. The used values for α and λ are $\alpha \in [0.01, 0.032]$ and $\lambda \in [0.0001, 0.001]$ in the second through the fourth experiments. The seeds are obtained from the first frame using k-means++ algorithm. In Fig. 3.d, the first frame points are clustered using the proposed framework into 8 clusters. Then, at each following selected frame, we cluster based on the previous frame centers, yielding spatio-temporal object detection in the scene in real time. At iteration number 10 shown in Fig. 3.d, clustering using 8 centers resulted in detecting 9 objects in the scene including: the wall and table, the cup, the cup lid, the cube, the perfume bottle, the perfume cover, the ball, and the router. Here, one cluster represents two objects (the wall and

the area under the table). Fig. 3.e shows the detected objects of the same scene but using 10 centers and after 100 iterations. Hence, 11 objects are identified including: the wall (3 clusters based on the orientation), the table, under the table, the cup, the cup lid, the cube, the perfume bottle, the ball, and the router. In Fig. 3.d and 3.e, the black cup lid is distinguishable from the dark navy cup. The perfume bottle cover is identified as well in Fig. 3.d. Also, in Fig. 3.d we can see a leakage in the cup cluster that is due to the concentrated shade in the corner which we don't have in other locations in the scene. In fact, the number of iterations in Fig. 3.d is 10. Therefore, this leakage will fade away after more iterations as in Fig. 3.e. Using another number of clusters, 10, as in Fig. 3.e, and at frame number 100, we can see that the scene objects are clearly identifiable. Moreover, the wall is segmented into three parts based on depth and orientation. As demonstrated in this experiment, it is very important to mention that the object detection results of the proposed framework depend on two major factors: the recommended initial location of the centers by k-means++ algorithm, and the number of frames or iterations that the framework is running on. Each frame corresponds to one k-means clustering iteration, the more iteration we have, the more stable the results. The initial centers will eventually converge to locations that best segment the scene points into objects influenced by their colors, orientations and locations.

In the third experiment, we run the proposed framework and moved the Kinect to see if the spatially detected objects are temporally consistent. The results are shown in Fig. 4. Having some objects in the scene in all frames but in different spatial locations of the frame by moving the camera around, the system still recognizes same objects temporally in the different frames.

In the fourth experiment, we introduced a moving subject into the scene and examined the extent of the system recognition. As shown in Fig. 5, the moving subject is recognized based on his location from the camera in the scene (depth values) and the clothing color (*Lab* values).

Since the motion from one frame to another is minimal, the centers that we carry from one frame to the following frame play the role of temporal linkage between frames. Clustering using these centers will produce consistent spatial clusters using the proposed similarity measure which is based on the described features: color, position and orientation.

To the authors' best knowledge, this is the first 4D automatic, unsupervised and real time spatio-temporal object detection. Other works [34-36] are based on video 2D frames only. Moreover, *Lab* color space is used extensively in food production standards, monitoring and research [27]. In spite of object detection research, such a color space has not been used before. The only usage for *Lab* color space in computer vision was reported in registration related work [37] and the actual used color space in the cited work is the *CIE L*a*b** color space as opposed to *HunterLab* color space that we are using. This introduces a methodological uniqueness in the proposed framework. Differences between the two color spaces *CIE L*a*b** and *HunterLab* are very subtle but can be found in [38]. We found *HunterLab* color space more tolerable with the noisy Kinect input as opposed to *CIE L*a*b**.

IV. Conclusion and Future Work

In this paper, we proposed an automatic, unsupervised spatio-temporal object detection framework using RGB-D cameras. The proposed framework has proven to spatially detect 3D objects in different scenes. The detected objects are temporally consistence, yielding 4D object detection. Results are provided in real time. The proposed framework can be applied to any object detection application in the fields of robotics, vision, or for surveillance and monitoring. The accuracy of object detection in the proposed method is leveraged through the use of local and global, spatial and temporal information. The local information is encoded by detecting the objects as spatial clusters with respect to their location, color, and orientation in one frame. The global information is merged into the detection by passing the clustering centers from one frame to the following frame in a temporal fashion. Our future work includes leveraging the GPU computation power to cluster all the 30 fps provided by Kinect rather than only utilizing 2 fps, comparing between different clustering algorithms, adding semantic to the detection, and finding moving objects trajectories.

Appendix 1

The proposed distance measure, f, which combines the position and color Euclidean distance with the angle between normal vectors of two points, satisfies the three properties of the distance function d in a metric space, which are reflexivity, symmetry, and triangle inequality.

The reflexivity property implies that the distance between two points, x and y, is always nonnegative and can be zero if and only if x equals y: $d(x, y) \geq 0$. The proposed measure f satisfies the reflexivity according to the following proof. First, if $x \neq y$, then the Euclidian distance between them is positive, and the value of the angular term in f is always positive because the range of the cosine is from -1 to 1, which leads to the positive range of the angular term from zero to 2. Secondly, if $x = y$, the Euclidian distance between them is zero and the angle θ between their normal vectors also equals zero. The angular term as well as the distance term become zero resulting in $d(x, x) + \lambda(1-\cos 0) = 0$. Thus, the proposed measure satisfies the reflexivity.

The symmetry property guarantees that $d(x, y) = d(y, x)$. The symmetry property holds for f since the Euclidian distance between the two points x and y and the angle between their normal vectors don't change regardless of the starting point. Here, $d(x, y)+\lambda(1-\cos\theta) = d(y, x)+\lambda(1-\cos\theta)$ and therefore $f(x, y) = f(y, x)$.

The triangle inequality implies that $d(x, z) \leq d(x, y) + d(y, z)$. We can show this property holds true for the proposed measure f by using the previous two

properties as follows: by the reflexivity propriety we have $d(x, y) + \lambda(1-\cos\theta)) \geq 0$. By using the symmetry propriety, $2(d(x, y) + \lambda(1-\cos\theta))$ can be rewritten as $(d(x, y) + \lambda(1-\cos\theta)) + (d(y, x) + \lambda(1-\cos\theta))$. The triangular inequality implies that $(d(x, y) + \lambda(1-\cos\theta)) + (d(y, x) + \lambda(1-\cos\theta)) \geq d(x, x) + \lambda(1-\cos\theta)$. Here, the right hand side is equal to 0 by reflexivity propriety, and thus we have $d(x, y) + \lambda(1-\cos\theta) + d(y, x) + \lambda(1-\cos\theta) \geq d(x, x) + \lambda(1-\cos\theta)$.

Thus, f satisfies properties of metric space distance function ■.

REFERENCES

[1] D. Oneata, J. Revaud, J. Verbeek and C. Schmid. "Spatio-temporal object detection proposals," in *Computer Vision–ECCV 2014*, 2014.

[2] M. Ghafarianzadeh, M. Blaschko and G. Sibley. Unsupervised spatio-temporal segmentation with sparse spectral clustering. Presented at British Machine Vision Conference (BMVC). 2014 .

[3] B. K. Horn and B. G. Schunck. Determining optical flow. Presented at 1981 Technical Symposium East. 1981.

[4] Y. Boureau, J. Ponce and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. Presented at Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010.

[5] M. Grundmann, V. Kwatra, M. Han and I. Essa. Efficient hierarchical graph-based video segmentation. Presented at Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference On. 2010.

[6] Y. Huang, Q. Liu and D. Metaxas. Video object segmentation by hypergraph cut. Presented at Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference On. 2009.

[7] P. Smith, T. Drummond and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. *Pattern Analysis and Machine Intelligence, IEEE Transactions On 26(4),* pp. 479-494. 2004.

[8] A. S. Ogale, C. Fermuller and Y. Aloimonos. Motion segmentation using occlusions. *Pattern Analysis and Machine Intelligence, IEEE Transactions On 27(6),* pp. 988-992. 2005.

[9] J. Xiao and M. Shah. Accurate motion layer segmentation and matting. Presented at Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference On. 2005.

[10] S. R. Rao, R. Tron, R. Vidal and Y. Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. Presented at Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference On. 2008.

[11] M. Jain, H. Jégou and P. Bouthemy. Better exploiting motion for better action recognition. Presented at Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference On. 2013.

[12] J. Chang, D. Wei and J. W. Fisher III. A video representation using temporal superpixels. Presented at Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference On. 2013.

[13] H. Wang and C. Schmid. Action recognition with improved trajectories. Presented at Computer Vision (ICCV), 2013 IEEE International Conference On. 2013.

[14] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions On 22(8),* pp. 888-905. 2000.

[15] C. Fowlkes, S. Belongie, F. Chung and J. Malik. Spectral grouping using the nystrom method. *Pattern Analysis and Machine Intelligence, IEEE Transactions On 26(2),* pp. 214-225. 2004.

[16] D. Oneata, J. Verbeek and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. Presented at Computer Vision (ICCV), 2013 IEEE International Conference On. 2013.

[17] J. Sánchez, F. Perronnin, T. Mensink and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision 105(3),* pp. 222-245. 2013.

[18] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision 59(2),* pp. 167-181. 2004.

[19] S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. Presented at Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference On. 2007.

[20] J. J. Corso, E. Sharon, S. Dube, S. El-Saden, U. Sinha and A. Yuille. Efficient multilevel brain tumor segmentation with integrated bayesian model classification. *Medical Imaging, IEEE Transactions On 27(5),* pp. 629-640. 2008.

[21] M. Carrel, M. Emch, P. K. Streatfield and M. Yunus. Spatio-temporal clustering of cholera: The impact of flood control in matlab, bangladesh, 1983–2003. *Health Place 15(3),* pp. 771-782. 2009.

[22] S. Kisilevich, F. Mansmann, M. Nanni and S. Rinzivillo. *Spatio-Temporal Clustering* 2010.

[23] S. Lloyd. Least squares quantization in PCM. *Information Theory, IEEE Transactions On 28(2),* pp. 129-137. 1982.

[24] R. S. Hunter. Photoelectric color difference meter. *Josa 48(12),* pp. 985-993. 1958.

[25] C. I. De L'eclaIrage–CIE. Recommendations on uniform color spaces, color difference equations and psychometric color terms. *CIE Publication (15),* pp. 9-12. 1978.

[26] R. S. Hunter. *The Measurement of Appearance* 1987.

[27] K. Leon, D. Mery, F. Pedreschi and J. Leon. Color measurement in l*a*b* units from RGB digital images. *Food Res. Int. 39(10),* pp. 1084-1091. 2006.

[28] G. Paschos. Perceptually uniform color spaces for color texture analysis: An empirical evaluation. *Image Processing, IEEE Transactions On 10(6),* pp. 932-937. 2001.

[29] A. Macedo-Cruz, G. Pajares, M. Santos and I. Villegas-Romero. Digital image sensor-based assessment of the status of oat (avena sativa L.) crops after frost damage. *Sensors 11(6),* pp. 6015-6036. 2011.

[30] R. W. G. Hunt, M. R. Pointer and M. Pointer. *Measuring Colour* 2011.

[31] M. H. Alassaf, Y. Yim and J. K. Hahn. Non-rigid surface registration using cover tree based clustering and nearest neighbor search. *Proceedings of the 9th International Conference on Computer Vision Theory and Applications* pp. 579-587. 2014.

[32] D. Arthur and S. Vassilvitskii. K-means : The advantages of careful seeding. Presented at Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. 2007.

[33] Microsoft Visual Studio 2013, "Auto-Parallelization and Auto-Vectorization," *Http://Msdn.Microsoft.Com/En-Us/Library/hh872235.Aspx,* Retrieved January 2015.

[34] M. Weber, M. Welling and P. Perona. *Unsupervised Learning of Models for Recognition* 2000.

[35] M. P. Daigavane and P. Bajaj. Real time vehicle detection and counting method for real time vehicle detection and counting method for unsupervised traffic video on highways unsupervised traffic video on highways. *IJCSNS 10(8),* pp. 112. 2010.

[36] W. Hu. Real-time on-line video object segmentation based on motion detection without background construction. *International Journal of Innovative Computing, Information and Control 7(4),* pp. 1845-1860. 2011.

[37] M. Korn, M. Holzkothen and J. Pauli. Color supported generalized-ICP. *In Proceedings of the 9th International Conference on Computer Vision Theory and Applications* 2014.

[38] Hunter Associates Laboratory, "Measuring Color using Hunter L, a, b versus CIE 1976 L*a*b*," *Application Note 1005.00,* 2012.

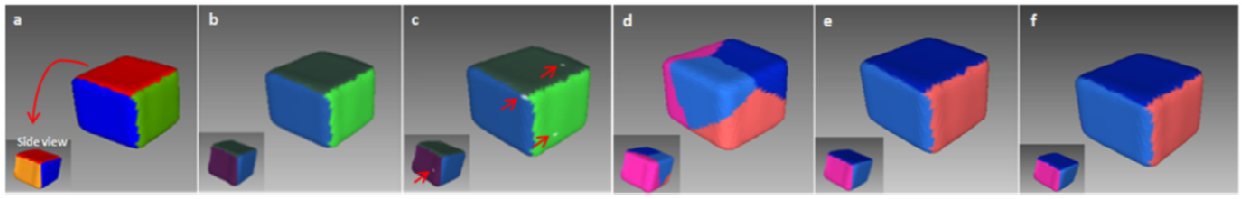

Figure 2. Using k-means clustering on a cube: a) Initial point cloud captured by Kinect. b) After *Lab* color transformation. c) Locating the k = 4 centers using k-means++. d) K-means clustering using the position Euclidian distance only. e) K-means clustering using the position and the color Euclidian distance as given in Eq.8 with $\alpha = 0.032$. f) K-means clustering using the position and the color Euclidian distance along with the normal vector as given in Eq.9 with $\lambda = 0.5$.

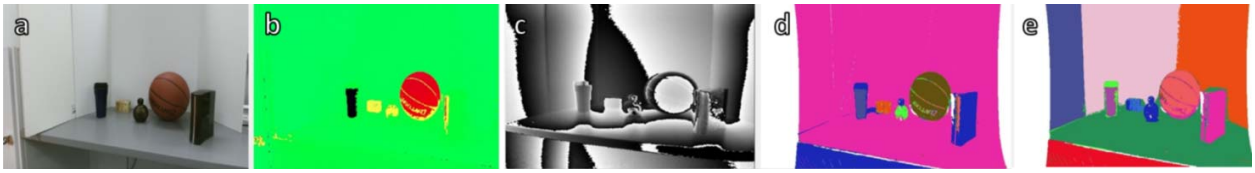

Figure 3. a, b, and c) The captured scene Kinect color frame in *RGB*, in *Lab*, and depth frame, respectively. d) The 3D clustered point cloud using k=8 clusters and detecting 9 objects. e) The 3D clustered point cloud using k=10 clusters and detecting 11 objects.

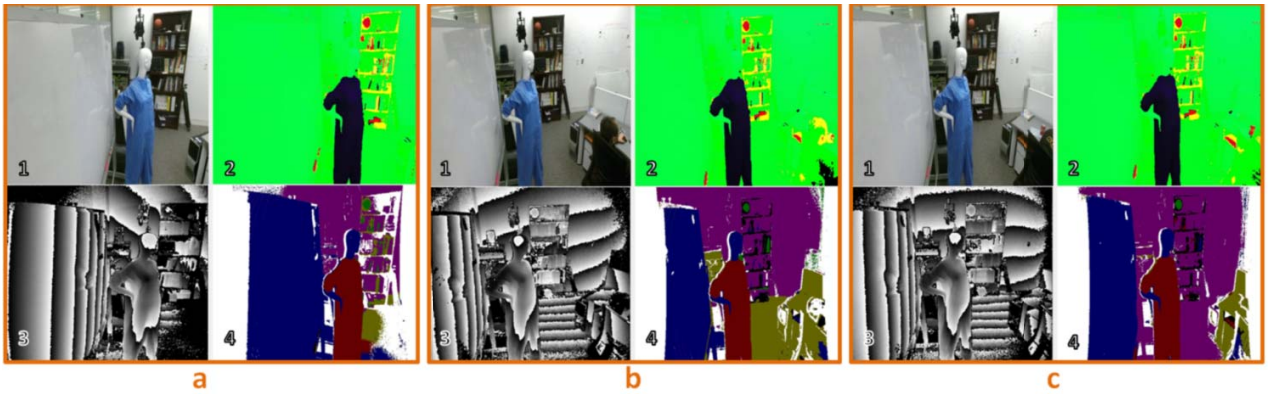

Figure 4. Moving the Kinect camera experiment. In 1 and 3, the captured scene Kinect color frame and depth frame; 2, the color frame in *Lab* space; 4, the 3D clustered point cloud. a) Using k=5 clusters, the scene objects are identified as in a.4: the board, the wall with the book shelves, items in the book shelves, the manikin, and the computer. The manikin head and hand are distinguished from the rest of the body because of their color difference. b) By moving Kinect to the right, the clusters are spatially consistent with previous frame. c) By rotating Kinect, the system still recognizes the same objects temporally.

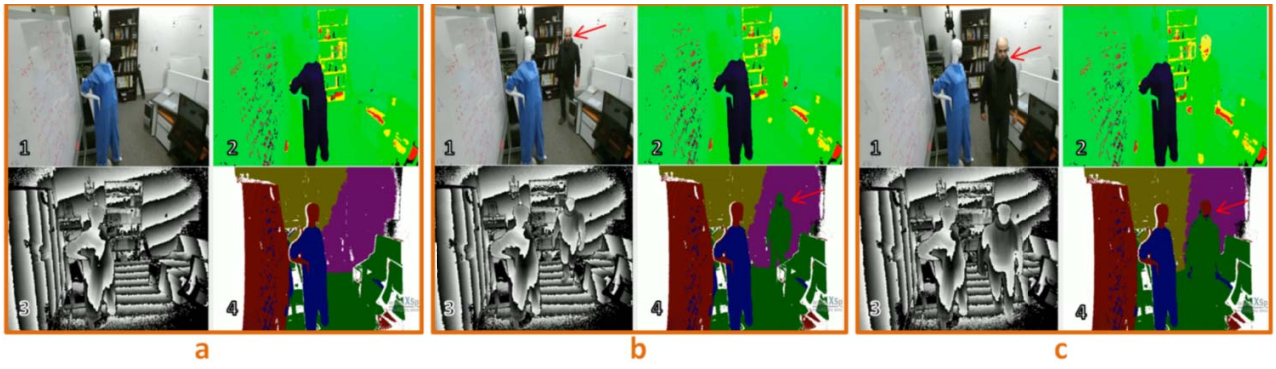

Figure 5. Introducing moving subject in the scene experiment. In 1 and 3, the captured scene Kinect color frame and depth frame; 2, the color frame in *Lab* space; 4, the 3D clustered point cloud. a) Using k=5 clusters, the scene objects are identified as in a.4: the board, the background wall as two clusters, the manikin, and the desk. The manikin head and hand are distinguished from the rest of the body because of their color difference. b) Subject in the right is entering the scene and gets identified and clustered with the closest cluster, the green. Note that the previously clustered objects are spatially consistent with previous frame. c) Another frame where the subject is still moving, head and hands started to be recognized as the subject moving closer to the camera due to their color difference. In addition, the system still recognizes the same objects temporally.