American Journal of Computing Research Repository, 2014, Vol. 2, No. 2, 28-32
Available online at http://pubs.sciepub.com/ajcrr/2/2/1
© Science and Education Publishing
DOI:10.12691/ajcrr-2-2-1

Developing Kinect-like Motion Detection System using Canny Edge Detector

Skander Benayed*, Mohammed Eltaher, Jeongkyu Lee

Department of Computer Science and Engineering, University of Bridgeport, Bridgeport
*Corresponding author: sbenayed@my.bridgeport.edu

Received January 17, 2014; Revised March 18, 2014; Accepted April 21, 2014

Abstract With the advance in video technology, motion-based computing system has an effect on both hardware and software, such as video surveillance, security alarm systems and game entertainment. For example, Kinect is a webcam style add-on peripheral for Xbox 360 game console manufactured by Microsoft, which is featured by RGB camera and multi-array microphone. Kinect is a motion-based software technology that can provide 3-D motion detection, skeleton motion tracking, and voice and facial recognitions. It is currently considered the cutting edge in the gaming world. In this paper, we developed a Kinect-like motion detection system for video streams without using real Kinect device. There are many approaches of motion detection for video streams. However, most of them are based on frame-based comparisons, which could be resource intensive and make it challenging to keep up with the continuous need for speed. Therefore, we present a novel method for motion tracking and identification that are based on the Canny edge detector. The experimental results show that the method is fast and effective in simulating applications, such as Microsoft Kinect motion detection and video surveillance system.

Keywords: *Kinect, Canny, edge detection, Fast Motion identification, video processing*

Cite This Article: Skander Benayed, Mohammed Eltaher, and Jeongkyu Lee, "Developing Kinect-like Motion Detection System using Canny Edge Detector." *American Journal of Computing Research Repository*, vol. 2, no. 2 (2014): 28-32. doi: 10.12691/ajcrr-2-2-1.

1. Introduction

Many techniques have been used for tracking motions of video streaming, but most of them are based on image features [5]. Such image features are usually a subset of image domain that highlights specific points of interest in the image. Feature detection is the process by which human brain responds to visual stimulus of a specific feature, which was discovered by David Hubel and Torsten Wiesel at Harvard University. Their accomplishment won them the 1981 Nobel Prize [5].

A tracking algorithm utilizes the detected features along multiple frames to track motions. Therefore, the reliability and repeatability of feature detection are critical for the success of any motion tracking system. Therefore, we employ edges for the image feature of Kinect-like motion detection system. Edges are typically a set of points with a strong gradient magnitude; these points can be chained together by some algorithms and give a contour of the object being tracked. Canny algorithm is one of popular edge detectors and it defines the optimal edge finding as a set of criteria that maximize the probability of detecting true edges. Canny significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image [2].

Motion-based computing system has an effect on both hardware and software, such as video surveillance, security alarm systems, and game entertainment, which require reliable tracking algorithm. For example, Kinect is a webcam style add-on peripheral for Xbox 360 game console manufactured by Microsoft, which is featured by RGB camera and multi-array microphone. Kinect is a motion-based software technology that can provide 3-D motion detection, skeleton motion tracking, and voice and facial recognitions. In this paper, we develop Kinect-like 2-D motion detection system for video streams without using a real Kinect device. Although there have been a lot of efforts on this area to improve accuracy and efficiency, there is no versatile solution in a way.

The method we propose in this paper is more flexible than the existing ones, such as template matching. The proposed approach is also more efficient than existing ones, since it does not require feature matching or image matching. In addition, it does not require any pre-existing database and associated parsing cost. In this paper we explore briefly some of the common techniques used to track and identify motions, and then propose a novel method of tracking algorithm based on Canny edge detector.

2. Methodology

2.1. Contours

A contour is a list of points that represent a curve in an image [4]. However, there are several ways to represent a curve. For example, contours can be represented by sequences in the way that every entry in the sequence encodes information about the location of the next point on the curve. Contours are extracted from black and white images. Outer or internal boundaries of white area are stored as polygons [7]. Figure 1 bellow shows how the black and white letter "S" is represented.

We have chosen to use contours for motion tracking and detection because they significantly reduce the amount of data to be processed which should significantly increase performance and speed for the application. Figure 2 shows an example of such a data reduction to be processed. As shown in the figure, the number of data points to be processed using contours is 1,000 times less than that of using grayscale image.

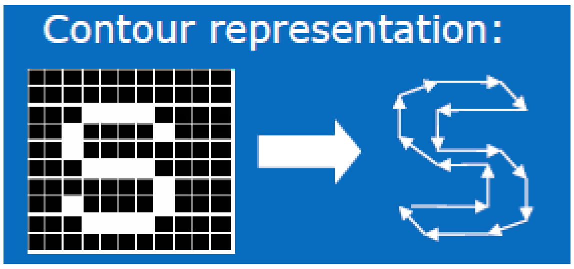


Figure 1. Contours in OpenCV [6]

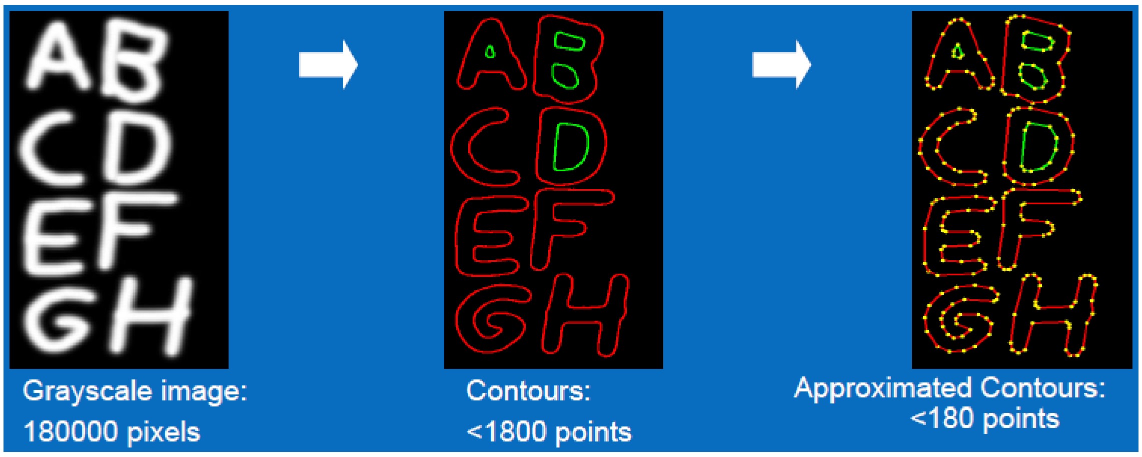


Figure 2. Reduction of the number of data points using contours [6]

2.2. Canny Edge Detector

Canny edge detector is an operation of edge detection that uses multi-stage algorithm relying on a number of adjustable parameters. Such parameters can affect the computational time and effectiveness of the outcome. An evaluation of the edge detection algorithms shows that canny edge detection algorithm is the most popular edge detector because of its simplicity and accuracy [1]. The stages of the algorithm can be described as following [8]:

1. *Noise reduction*: This type of edge detection performs some noise reduction by convolving the raw image with a Gaussian filter.
2. *Finding gradient of the image*: The edges should be marked where the gradients of the image has large magnitudes.
3. *Non-maximum suppression*: after gradient is estimated, only local maxima should be marked as edges.
4. *Edge tracking and Hysteresis thresholding*: Canny uses thresholding with hysteresis which requires two thresholds -high and low hysteresis is used as a means of eliminating streaking. Streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold [9].

In general, Canny edge detection algorithm has two flexible parameters: the size of the Gaussian filter and the threshold. In our case we optimized the thresholds as shown below in Figure 3; the left image is the original representation of the object whereas the right one is the processed image using canny edge detection.

The blue contour around the object as shown in Figure 3 preserves the overall shape of the object. Moreover, the green circle represents the virtual circumference and the red circles represent the hit points which are the intersection between the contour and the virtual circumference. These will be discussed further in the next sections but just note that these are the key components that are critical for our subsequent processing.

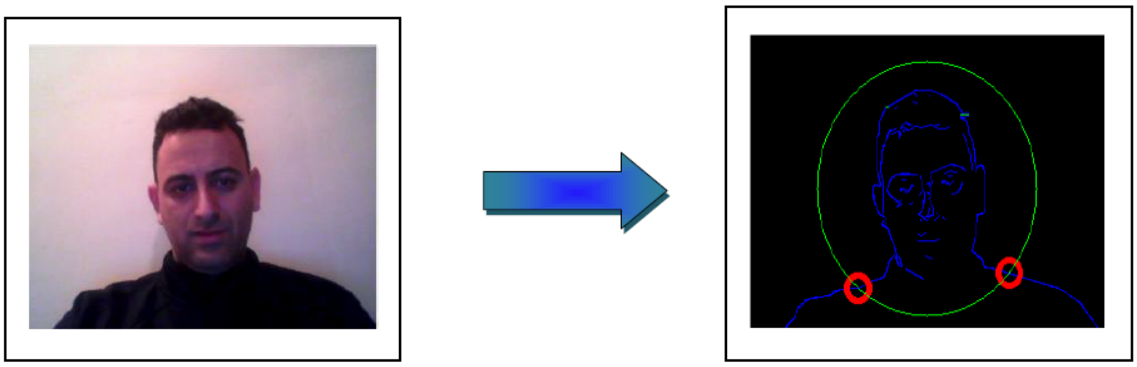


Figure 3. Example of canny edge detection

2.3. Virtual Circumference

Note that the original picture on the left in Figure 3 does not have any circle around the object, as it is shown in the processed image on the right. The circle in the processed image is called the virtual circumference and it is displayed only for explaining what is happening behind the scene for each frame during the processing.

The virtual circumference as shown on Figure 3 is the green circle around the object. It is a representation of an expected range of object motion, which can lead to their identification. This identification is accomplished by setting a matrix in Figure 6 that maps a set of points from within the virtual circumference to a particular motion. The way it works is while the video is played and the frames are processed the intersection between the contour of the object and the virtual circumference in each frame as described earlier called hit points, will be used along with the matrix in Figure 6 for that virtual circumference to identify the movement.

If this virtual circumference has a geometrical shape that can be defined by a mathematical equation, then we can verify the set of points forming the contour around the object with that equation. Therefore, we can claim that an intersection between the contours and virtual circumference was made and the motion can be identified. In our case, we noticed that the set of expected movements as shown in Table 1 are within a circle surrounding the object. We can use the Euclidian distance to measure the distance between the center of the object and the points on the contour. When it is equal to the radius of our virtual circle circumference, it implies that we have an intersection. That means we have hit one of our expected target points and the motion is identified.

The Euclidean distance is given by the Pythagorean formula. By using this formula as a distance, Euclidean space becomes a metric space [3].

$$d(C,P)=\sqrt{(p_1-q_1)^2+(p_2-q_2)^2}$$

Where:

C is Center

P is point

C (p_1, p_2) is the center of gravity of the object we are tracking.

P (q_1, q_2) is a point on the object contour.

In this paper, for the proof of concept we use the circle. However, it could be any shape as long as it is expected and the intersections identify uniquely a movement, this method can be applied.

3. Experimental Results and Discussion

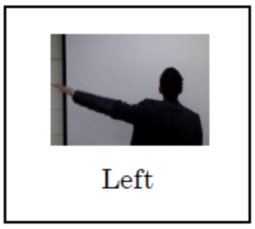
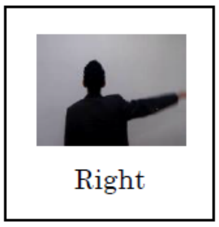
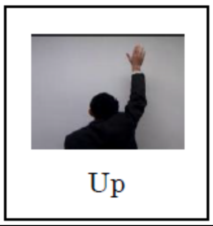
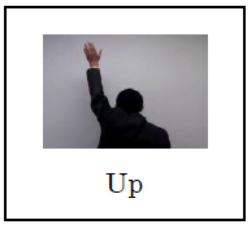
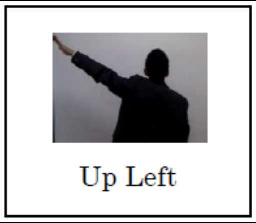
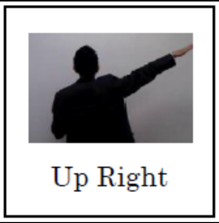
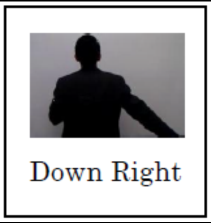
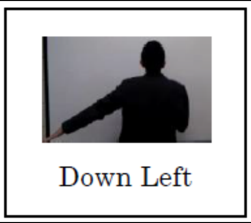
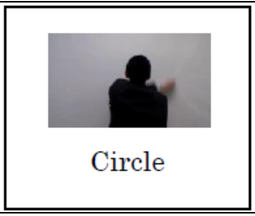
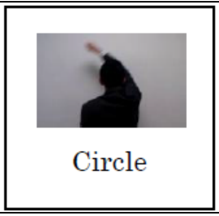
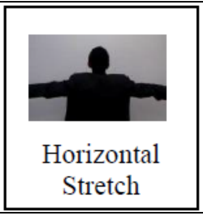
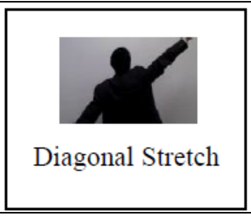
3.1. Tools

For our implementation, we used the OpenCV (Open Source Computer Vision Library) which was initially introduced by Intel and released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000. This library is implemented in C/C++. Thanks to Heiko Kiessling who made a wrapper around this library and made it available through code project it was possible for us to use a more recent programming language C# build recently from the ground up by Microsoft under the .NET platform. Using visual studio 2010, C# 4.0 and .NET framework 4.5 along with the wrapper interfacing OpenCV called cvlib allowed us to implement a nice application that illustrates the reasoning behind this paper, and a walkthrough this methodology.

3.2. Experiment

Table 1 below describes the set of movements that needs to be identified, the methodology described above is applied and the results and discussion will follow thereafter.

Table 1. Expected movements with the details of each movement

Movement	**Details of the Movement**			
Basic Movement				
Diagonal Movement				
Advanced Movement				

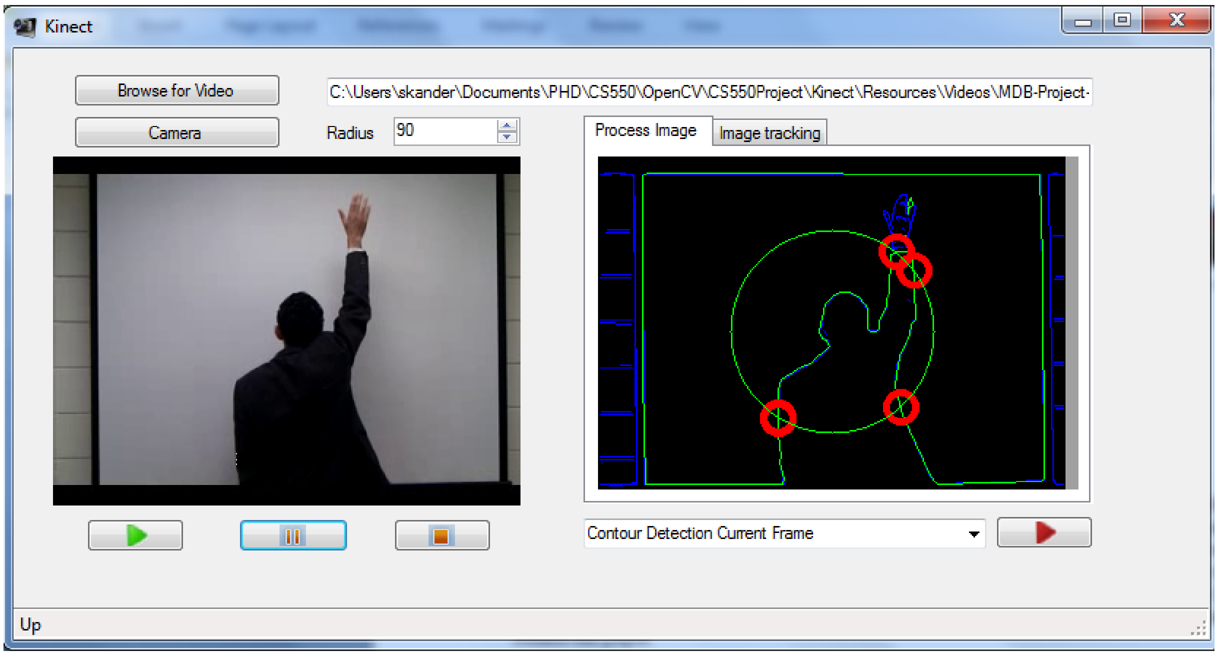

Kinect
Browse for Video
C:\Users\skander\Documents\PHD\CS550\OpenCV\CS550Project\Kinect\Resources\Videos\MDB-Project-
Camera
Radius
90
Process Image
Image tracking
Contour Detection Current Frame
Up

Figure 4. Up Movement identified

3.3. Results

All movements were identified successfully using the technique explained earlier. Figure 4 illustrates screen shots from processing the basic movement video for a simple movement.

Figure 5 is another example from playing the advanced movement video. The same principle applies but in this case note that a second virtual circle displayed with a smaller radius, that was required to detect a slightly more complex movements such as a circle detection, where we needed extra timer to insure that the start and end of the circle is done within a certain amount of time other than that advanced movement such as horizontal stretch which is really a combination of a two simple movements Left and Right is successfully identified as shown below.

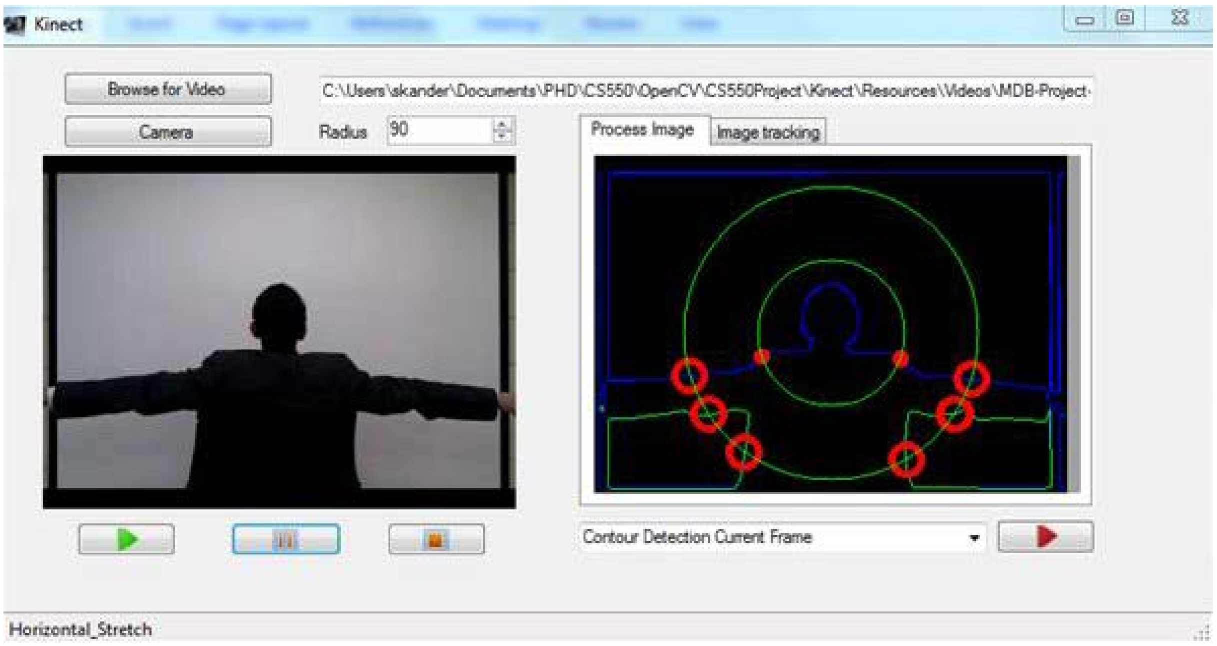

Kinect
Browse for Video
C:\Users\skander\Documents\PHD\CS550\OpenCV\CS550Project\Kinect\Resources\Videos\MDB-Project-
Camera
Radius
90
Process Image
Image tracking
Contour Detection Current Frame
Horizontal_Stretch

Figure 5. Horizontal Stretch identified

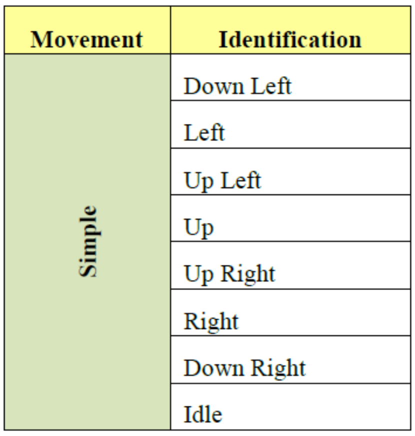
First of all, we started by playing the video for the basic movement. The format is AVI at a rate of 25 FPS (frames per second). Then, we paused at a particular frame. However, the movement as displayed is an "Up" movement identified in the status bar as shown in Figure 4 while processing the video.

In order to better understand what is going on behind the scene, we added a tab that helps braking up the steps performed in real time to accomplish the motion identification.

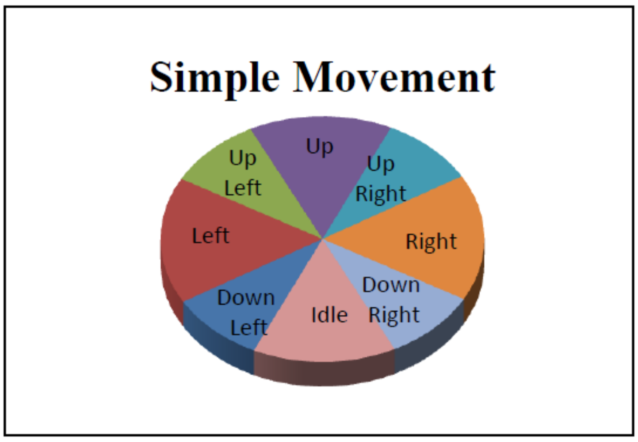
Step 1: Determine the radius of the virtual circumference in our case it's a circle that is good enough to capture all expected movements, now as its name implies it's not a real circle it's a theoretical one but we displayed it for the purpose of demonstration.

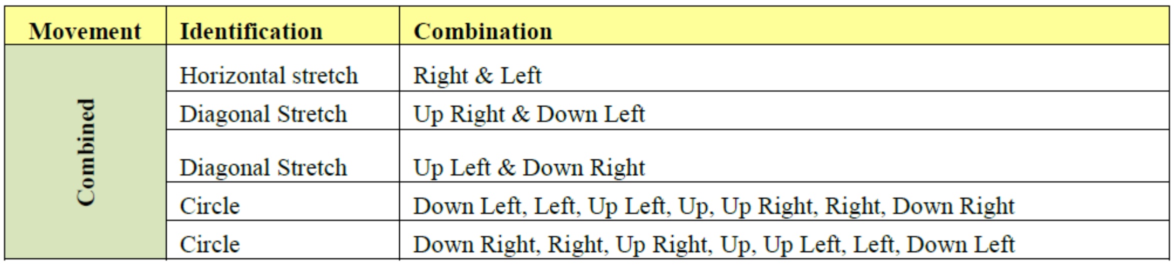
Step 2: Compute the distance between the center of gravity of the object, in our case the person making the movements, and the points on the contour detected by Canny. This was done using the Euclidian norm as explained earlier. If that distance happen to be equal to the radius of our virtual circle that means there is an intersection which is marked by the small red circles as the edges of the contour crosses the virtual circle. Note that the two bottom red circles are skipped from the interpretation as they denote the idle position. All points within a certain threshold that fall into the idle category are taken care of by our algorithm.

Step 3: The final step to this process is the motion identification, based on the intersection points flashed out in step2 and a known Matrix of the expected movement as shown in Figure 6, the identification is accomplished and is displayed on the status bar furthermore as a bonus we did add sound to the identification to enhance the user experience as well as kept the log of all identifications in the image tracking tab for reference.



Movement	Identification
Simple	Down Left
	Left
	Up Left
	Up
	Up Right
	Right
	Down Right
	Idle





Movement	Identification	Combination
Combined	Horizontal stretch	Right & Left
	Diagonal Stretch	Up Right & Down Left
	Diagonal Stretch	Up Left & Down Right
	Circle	Down Left, Left, Up Left, Up, Up Right, Right, Down Right
	Circle	Down Right, Right, Up Right, Up, Up Left, Left, Down Left

Figure 6. Movements Identification Matrix

3.4. Discussion

Like any other tracking technique, there is no such thing as one fits all. Depending on the application it may or may not be the method of choice. However, this technique cannot handle cases where the set of motion is unexpected. In addition, it is also sensitive to the noise despite Canny noise reduction especially surrounding the virtual circumference area. Also there are thresholds to be set for optimal Canny detection for a particular application but despite that there are some challenges and rooms for improvement notably smoothing, noise reduction and background segmentation.

Besides the speed this methodology showed, it's also important to mention the versatility aspect for instance in the case study we presented the object can change and the system still performs well unlike template matching which is tied to what have been saved in the database.

4. Conclusion

In this paper, a new method for Kinect-like motion identification was proposed. It relies on edge detection feature which is accomplished using Canny multiple steps algorithm as well as a virtual circumference around the center of gravity of the object being tracked. The intersection between the virtual circumference and the contours points are hit points that signal an expected motion that is then identified using a matrix that maps virtual circumference points to appropriate expected motion. Although processing each and every frame at 25 frames per second was not necessary, but it was used to show the low cost, low overhead of this method. It still performed well at an amazing speed. It is our belief that this technique could be applied not for the gaming applications but also for surveillance cameras. The speed and versatility could be a great asset for the appropriate application.

References

[1] Bowyer, K., Kranenburg, C., and Dougherty, S. Edge detector evaluation using empirical roc curve. *Comput. Vision Image Understand.2001, pg.10*, 77-103.

[2] Canny, J., *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 1986, pg.679-698.

[3] Elena Deza and Michel Marie Deza, "*Encyclopedia of Distances*", Springer, 2009, pg.94.

[4] Gary Bradski & Adrian Kaehler. Learning OpenCV: Computer Vision with the OpenCV Library, O'REILLY, 2008.

[5] http://en.wikipedia.org/wiki/Feature_detection_(computer_vision), March 2012.

[6] http://www.codeproject.com/Articles/28465/Easy-to-use-Wrapper-DLL-for-Intel-s-OpenCV-Library, March 2012.

[7] http://www.focuspeinc.com/UploadFile/Download/2009090716450143.pdf, March 2012.

[8] http://en.wikipedia.org/wiki/Canny_edge_detector, March 2012.

[9] http://www.matlabcodes.com/2011/01/canny-edge-detection.html, March 2012.