OBJECT RECOGNITION THROUGH KINECT USING HARRIS TRANSFORM

Azeem Hafeez
Assistant Professor of Electrical Engineering Department, FAST - NUCES
Hafsa Arshad
Ali Kamran
Rida Malhi
Moiz Ali Shah
Muhammad Ali
Saad Malik
Final Year Student of Electrical Engineering Department,
FAST - NUCES, Faisal Town, Lahore, Pakistan

Abstract

With the growing research in the field of computer vision and image processing, numerous applications of Object Recognition have been developed. This paper discusses the recognition of objects through image processing. For this purpose RGB camera is being used. Initially, templates of objects (ball, box and bottle) are stored and processed for template matching. At real time original image data is taken and then divided into four frames, containing an object in each frame. These frames have resolution equal to the resolution of the templates. All the processing is done on colour images. Then, Harris Transform is applied. Using the co-variance matrix, error is calculated and compared with the threshold to match objects. Object is basically recognized by calculating the difference in the co-variance of images.

Keywords: Co-variance, frame extraction, down-sampling, harris transform, speech recognition, kinect

Introduction

Speech recognition is an active region of research in the field of computer vision. In this system to recognize the voice commands, built-in speech recognition of Kinect was used. A speech grammar was created in this system and a code in C# language was written to start the speech engine and match commands. The input to this system was the voice command from the microphone array of Kinect. Anyone can give a voice command and the software acts according to the response stored for that command. For voice commands, two to three words were stored corresponding to each image. We tested the system on four objects. Images of four objects were being used as samples. The object recognition was done using image stream data from the RGB camera. After acquiring the pixel data of the image frame, processing was done. First, the background colour was identified (in this case the colour of the background was red) and removed, turning everything, except for the object, black. The resulting pixel data was of the object with a black background. The R, G and B arrays were separated for further processing. The covariance was used to compare the similarity between the template image and the image taken at run time. In this way, Harris Transform was used for object recognition.

Implementation of the proposed algorithm

This section describes in detail the implementation and the steps of the different algorithms used in this object recognition system. The steps of the system are as follows:

Take images of the objects and store them after scaling to required resolution (160x160 in this case) so that they can be used as templates for matching.

Receive a voice command to search the template image of the object in the database.

Remove the background of both, the image taken at run time and the template image.

Extract pixel information from image stream data of Kinect.

Apply Harris transform on both the template image and the image acquired at run time from Kinect to compare the similarity between both.

The steps mentioned above are explained in sub-sections given below:

Speech Recognition

A library of speech grammar for identification of commands given by user was build first. The templates were stored corresponding to each object name in the database. User's voice command was taken as an input through the microphone array of Kinect to find the desired object. The C# language code was executed to start the speech engine and match commands. If the command was in the library, template image of the object was searched and loaded from the data base. Otherwise no further processing was done. Built in library of Kinect was used for speech -recognition.

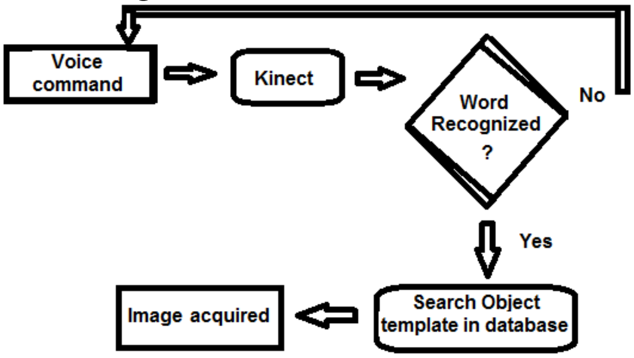

Fig1: Speech Recognition through Kinect

Algorithm for reduction of data for fast computation

The template image resolution was 640 x 480 initially. It was down sampled so that its resolution matches that of the divided frame of the image taken at run time. The template image was down sampled by a factor of 12 to make the image resolution 160 x160. Template image pixel data had 640 columns and 480 rows. To reduce resolution of columns from 640 to 160, pixel data was collected as pixeldata0, pixeldata4, pixeldata8,. . . . . , pixeldata640. Consecutive 3 pixels data was removed. Similarly for the rows, pixel data was collected as pixeldata0, pixeldata3, pixeldata6, ... ,pixeldata480. Consecutive 2 pixels data was removed and rows were reduced from 480 to 160. The new pixel data obtained resulted in the down sampled image which was further processed. Fig2 illustrates how down sampling was done on the template image.

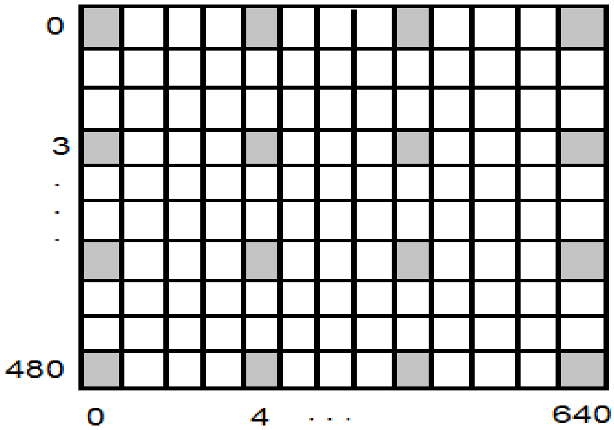

Fig2: Downsampling of template image.

Fig3:Result of Downsampling

Extraction of frames

Image obtained from the image stream data of KinectCam was of resolution 480 x 640. There were four objects in the image taken at run time. Those four objects were placed at equal distance. Therefore four frames were captured from the image taken at run time, each frame containing one object. The image of each object was of 160 x160 resolution, which was equal to the template resolution. Harris Transform was then applied to the resulting images of 160x 160 resolution.

Fig4a: Image captured through Kinect.
Fig4b: Extracted images.

Harris Transform

Harris transform compares similarities between different images using co-variance. Image data from the Kinect image stream is in the form of B,G, R, and α.

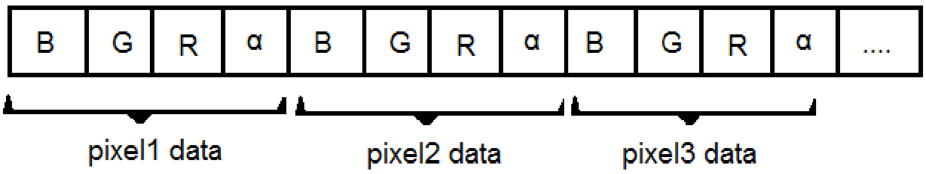

Fig5: Data captured through Kinect.

For each image frame extracted from the image taken at run time, separate arrays were made for blue, green and red (α was not required).The template image data was also put in separate arrays of blue, green and red and Harris transform was applied to those arrays as well. Harris transform was used to compare similarities between the template image and the image frames of image taken at run time using co-variance. The general formula for the covariance is:

$$C_{xy} = \frac{1}{N-1}\sum_{i=0}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

For image covariance, covariance was calculated of red data of image1 with red data of image1, red data of image1 with blue data of image1, red data of image1 with green data of image1. Similarly,for blue data of image1 and green data of image1. In the same way image2 covariance was also calculated.

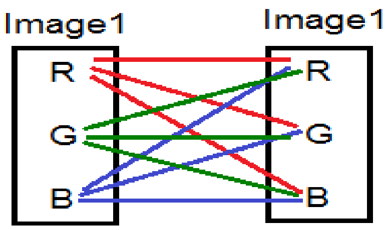

Fig6: combinations for co variance

The matrix obtained by nine covariance was as shown below:

$$\begin{bmatrix} C_{RR} & C_{GR} & C_{BR} \\ C_{GR} & C_{GG} & C_{GB} \\ C_{BR} & C_{BG} & C_{BB} \end{bmatrix}$$

The next step was to calculate error between covariance of two images correspondingly. The formula used to calculate error is as given below:

Error= $C_{RR1} - C_{RR2} / C_{RR1}$

	Object1			Object2			Object3			Object4		
1	24.99335	22.89622	17.59371	51.07847	26.35609	32.52685	43.24374	25.42037	32.19248	41.04235	31.87159	99.13445
	22.89622	28.34861	330.9726	26.35609	17.0443	11.11979	25.42037	21.77558	12.45923	31.87159	3.130935	7.642942
	17.59371	330.9726	177.5095	32.52685	11.11979	11.26592	32.19248	12.45923	13.3127	99.13445	7.642942	17.08174
2	24.21977	22.53645	18.36702	50.78252	25.72562	31.59342	42.89225	25.65707	30.97227	40.56097	33.16213	107.2696
	22.53645	29.34286	319.5949	25.72562	17.1488	10.57032	25.65707	20.23027	12.65719	33.16213	2.116496	7.759339
	18.36702	319.5949	180.288	31.59342	10.57032	10.32699	30.97227	12.65719	14.99495	107.2696	7.759339	16.73827
3	13.29746	36.49498	10.44239	43.40431	39.3715	48.40616	34.33672	38.99794	48.2064	31.77067	9.601142	51.37378
	36.49498	58.90314	16.84206	39.3715	73.6028	82.56217	38.99794	61.53821	77.92561	9.601142	67.44975	82.05721
	10.44239	16.84206	27.55469	48.40616	82.56217	77.06865	48.2064	77.92561	70.43803	51.37378	82.05721	78.76218
4	11.61481	32.81418	19.74363	27.01686	35.7318	53.52562	15.45042	35.23882	53.177	12.21654	15.0202	38.01219
	32.81418	3.428459	236.649	35.7318	37.73593	29.04361	35.23882	9.586931	10.84381	15.0202	22.76824	26.86203
	19.74363	236.649	110.8644	53.52562	29.04361	32.73036	53.177	10.84381	13.43057	38.01219	26.86203	37.50193
5	7.567974	11.73985	50.33292	29.81376	15.62198	14.25821	18.43217	15.08733	14.00688	15.39327	51.59891	153.7547
	11.73985	29.44913	184.1551	15.62198	54.68345	41.84698	15.08733	34.29628	26.96678	51.59891	44.00459	40.34494

Conclusion:

Object Recognition was done quite efficiently. We have done more than 20 experiments from which five have been shown in the above figure. Minimum error of 8% and maximum error of 20% occurs between the covariance of the template image and the matched object image. This method uses the extraction of frames, downsampling of the image and covariance of image. Image matching has been done on the basis of their R, G and B value. A distinguishing feature of this method is that it works for objects placed at a different angle than the template image.

References:

Jason Owens, “Object detection using kinect,” in Title of His Published Book, Army Research Laboratory

David Katuhe, Programming with the Kinect Windows SDK

D.G.Lowe, Distinctive image features from scale invariant key points