# Object Tracking and Spatial Recognition

*For Creating Percussion Instruments*

*Vishal V Kole*
Graduate Student
B. Thomas Golisano College of Computing and Information Sciences, RIT
vvk3025@rit.edu

***Abstract*—Object tracking has always been one of the blazing topics in Computer Vision. We could harness this feature to create various application and simplify the problems. This paper is one such attempt to use object tracking and spatial detection to simplify creating percussion instruments.**

***Keywords*—Object tracking; spatial recognition; computer vision; Lucas-Kanade; optical flow; percussion instruments.**

## I. Introduction

Object tracking has always been one of the blazing topics in Computer Vision and there are many applications to it. I have used this feature to create one such application and to simplify creating percussion instruments.

The target of this project for me was to understand the different methods for object tracking and motion detection. The problems and constraints in motion detection and object tracking algorithms and the benefits of it. I faced a lot of problems with the detection of the drum sticks and with the motion blur which I avoided with some tricks. Spatially recognizing the motion of the object was also not straightforward and had to be finetuned for the application. I have used two approaches to solve this problem, first one being the simple approach of computing the linear locations of the drum stick, second on is by using the Lucas-Kanade optical flow algorithm.

## II. Previous Work

I did not find any literature on solving the percussion instruments. But, there are lot of papers on the optical flow algorithms and on the motion and object tracking part. I have referenced them in the references and cited the numbers wherever it was used.

## III. Experiments

### A. Approach One

I wanted the project to run on a simple camera and drum sticks. I began by taking the images from my laptop camera of a single drum stick at first. This was followed by the detection of the foreground and background pixel. The simple way to detect a foreground pixel from a background pixel is to use background subtraction.

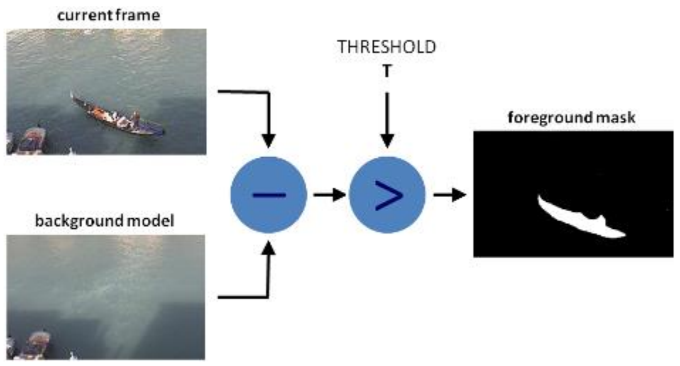

Fig. 1. *Example of background subtraction with threshold [1]*

I took a picture of my background, which was a dark colored cloth on the wall and a picture with the drumstick in it. Using the background subtraction, I got the below image from the process.

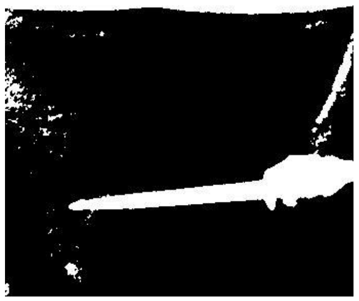

Fig. 2. *Drumstick after background subtraction with thresholding*

As we could see, there was lot of noise in the image. This noise was due to the uneven background surface from the cloth as well as due to the changing lighting conditions. Restricting the noise from the lighting condition was important as it would have given varying results for the same algorithm. I managed to do this by manually setting the camera properties to a fixed value, changing slightly according to the time of the day.

There was still noise in the image which was clearly salt noise for my dark background. I used the erosion with dilation to retain the actual shape of the drumstick. I got a decent image from it.

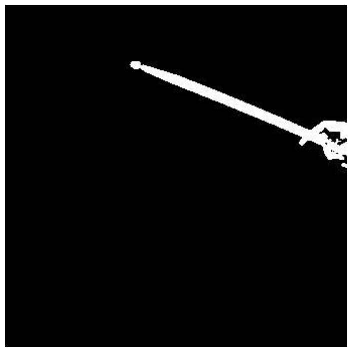

Fig. 3. *Foreground image after noise removal.*

I started testing the and taking images while mimicking to play the instrument. The problems did not end with there, I had motion blur with the stick moving very fast on the background creating a very noisy image with no clear boundaries.

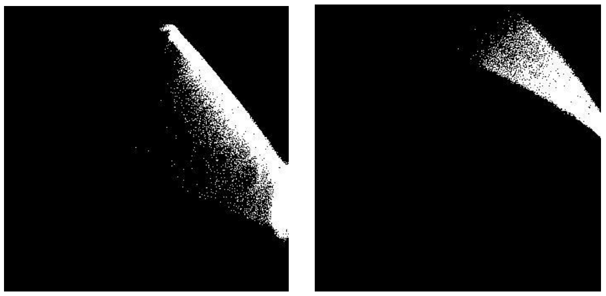

Fig. 4. *Foreground image of motion blur.*

This was a major concern at this point because it gave no clear indication as to where the stick was in the image. It this made it difficult to track it in successive images. The motion blur also resulted into the tip of the drumstick to disappear which made it even harder for me to locate in the spatial domain for the sound to be played. There was also some bending of the stick observed due to the fast motion. After some research on the topic I found out that this could only be avoided with a fast capturing camera.

At this point I dropped the idea of using the background subtraction and started searching for some other approach to work with.

## *B. Approach Two*

I then tried a simple approach which I had in the back of my mind, it was to use a color blob to detect the end of the stick. This could eliminate the disappearing of the end of stick and gave me more pixels to play with. I tied a red tape to one stick and a green tape to other sticks' end and tried taking the same images with me virtually playing the drums.

There was less noise this time with the thicker blob at the end which gave me good images after noise removal. This also allowed me to distinguish the two sticks from one another.

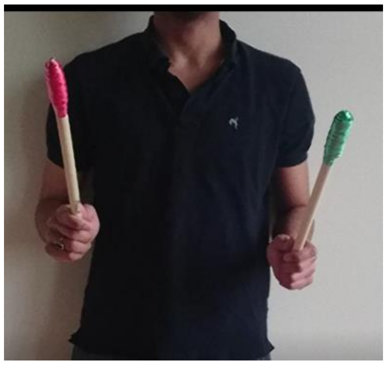

Fig. 5. *Drumsticks with the color blobs.*

A grayscale image was obtained using this RGB colored image and it was then subtracted from the green channel of the original image. This gave me only the green pixels in the image. I did the same for the red pixels as well giving me the red stick as well. Doing this made it really easy for me to eliminate the need of having a controlled background. Although, I had to control the colors in the background, but only the large ones. The small green and red color noises were handled by the code by using the region properties.

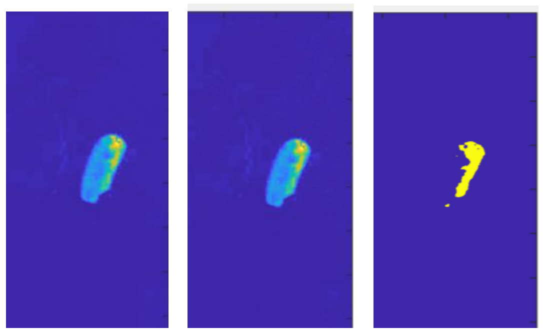

Fig. 6. *1.Green channe image for the drumstick. 2. Noise removal after median filter. 3 Binarized image to get the blob.*

After doing this I processed the image for any noise using the median filter and then binarizing the image to get just the green and red blob in the image. This binarized image is again processed for some salt noise using the erode and dilate morphological operations.

After the noise removal, I select the biggest blob in the image eliminating any other blobs that might create problems. Using region properties of the image and the blob I take the centroid of the blob and track the centroid to create a length hundred queue to track the flow of the blob.

The blob's motion is then analyzed and a proper sound is generated after looking at the change in the direction of the blob, this happens when the stick hits the bottom of the image. Continuous tracking of the centroid lets us utilize space in the image and can segment the space into different levels. For the current purpose, I have divided the space into two different sections, the left and the right.

The problems I faced in this approach was that the processing of the image took some time and my video input was giving me no more than 30 frames per second. This created a lag and I got around 12-13 frames per second after optimization. This was not good to detect fast moving object and not good to detect the drumsticks as it resulted into a huge change between two successive images.

After researching about the techniques to make it faster, I came across the video streaming class and video streaming objects in MATLAB. The allowed me to take the images in real time from a stream and avoided the need to call a fetch command whenever I need an image. This improved the performance but still did not reach the mark where the sticks were detected with accuracy and the data shift was consistent.

### *C. Approach Three*

There is another way I tried doing it. It was to use the optical flow for an image. The problem with optical flow which I realized after using it and playing with it in my previous version was that it too needs continuous images without high change in the successive frames. Getting 30 frames per second was a major task and to compute the optical flow was necessary. I then switched to recorded videos to get the 30 frames per second mark.

A video of me playing the drums was then recorded in 120 frames per second to try and test the optical flow model. Lucas-Kanade algorithm seemed appropriate to use to detect the flow. It gives back the velocity of the pixel in X, Y direction. Another feature is it also gives the magnitude and orientation of the pixel.

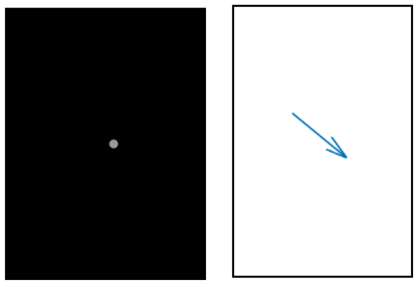

Fig. 7. *Object centroid and the optical flow estimation by using Lucas-Kanade method for optical flow.*

## IV. PROBLEMS

Background subtraction without the color segmentation gave good foreground pixels but it created lot of problems with the motion blur and detection of the tip of the drumstick as mentioned above. Second approach worked well giving decent results and lot of tweaking of parameters were necessary to make it work more efficiently and accurately.

Lucas-Kanade algorithm for optical flow had an issue as well. The algorithm detects only within a certain threshold of value, limiting the search for the pixel to note the flow. Although, the threshold could be changes, it still requires the images to have less shift in them to get the accurate results. This required me to take 120 frames per second video to make it work.

## V. FUTURE WORK

Future work would be to perfect the Lucas-Kanade optical flow model and detect the exact point of contacts in the space and to detect the shift in the velocities to measure the intensity of the hit to give different sound effects.

## REFERENCES

[1] http://docs.opencv.org/trunk/d1/dc5/tutorial_background_subtraction.html

[2] Richard Y. D. Xu 1, Joshua M. Brown 2, Jason M. Traish 3, Daniel Dezwa, "A Computer Vision Based Camera Pedestal's Vertical Motion Control" Pattern Recognition, 2008. ICPR 2008. 19th International Conference on 8-11 Dec. 2008

[3] Kang Xue, Patricio A. Vela, Yue Liu "A modified KLT multiple objects tracking framework based on global segmentation and adaptive template" Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012). 11-15 Nov. 2012

[4] Olivier Barnich and Marc Van Droogenbroeck. "ViBe: A Universal Background Subtraction Algorithm for Video Sequences". IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 20, NO. 6, JUNE 2011

[5] Mrs.I.Manju Jackin, Mr.Manigandan M "Wireless Vision Based Object tracking using Continuously Adaptive Mean Shift Tracking Algorithm" 2009 International Conference on Advances in Recent Technologies in Communication and Computing.

[6] http://www.mathworks.com/matlabcentral/fileexchange/24677-lucas-kanade-affine-template-tracking