Spot Detection using Predictive Tracking Model

Digital Signals and System Architecture Lab

ELE551C

Pradyumn Agarwal

5PME - 2140327

September 27, 2023

CERTIFICATE

This is to certify that the work titled "Spot Detection using Predictive Tracking Model," is a report of the Mid Semester Project based on the topic Signal Processing, has been carried out by Pradyumn Agarwal(5PME - 2140327) under my supervision in partial fulfillment, to award marks for Continuous Internal Assessment - II for the course Digital Signals and System Architecture Lab - ELE551C, during the academic year 2023-2024.

Place: Bengaluru
Date: September 27, 2023

Dr. Vineeth V.
Assistant Professor
Department of Physics and Electronics
Christ(Deemed to be University)
Bengaluru - 560029

Dr. Benny Sebastian
Head (Electronics)
Department of Physics and Electronics
Christ(Deemed to be University)
Bengaluru - 560029

CONTENTS

I Introduction 1

II Goals 1

III Methodology 1

IV Algorithm 1

V Implementation 2
V-A Finding the Spot Location 2
V-B Finding Absolute Position 2

VI Results 2

VII Conclusion 2

Acknowledgment 3

Appendix 3

Spot Detection using Predictive Tracking Model

Pradyumn Agarwal
5PME - 2140327

***Abstract*—I propose a spot tracking algorithm with reduced time complexity and recognition based on movement. The algorithm reduces the search area and hence minimizes the data set to be analyzed.**

I. Introduction

Object recognition and tracking usually requires application of filters and convolutions on the complete frame. The time complexity of such an algorithm would be $O(n^2)$, since the filter must act on at-least $N \times N$ pixels of each frame. Smoothing and noise reduction also requires preliminary analysis of the data and results in a compromise of either time or memory. In high-res videos, both time and memory resources are limited. A simple spot detection based on the intensity of the spot would require vast resources if implemented using a brute force algorithm.

In the proposed algorithm we constrain the search region to a fixed offset region from the previous location of the spot. The value of the offset also keeps updating based on the speed of the spot, ie. the magnitude of deviation in the previous iteration. Here the time complexity would be almost linear with subtle variations. A fail-safe must be implemented incase the tracking point jumps out of the searching area. In such a situation and also to establish the initial point, the spot is found at the global maxima.

It is to be noted that if several points are to be tracked using an idea of local maxima of the filter, the algorithm proves much more beneficial than brute forcing. Although the time complexity in brute force remains the same, in the algorithm suggested, the time is only multiplied by the number of points, hence still time friendly.

II. Goals

The algorithm serves the following purposes-

- Detection(complete search) for spot is only required initially.
- Is extremely fast than running the complete search.
- Re-detects the spot when out of bounds
- Establish an identity for each spot and hence implement multiple simultaneous tracking.
- Reduce loss of data across frames by preserving the location.

III. Methodology

The main goal of this algorithm is to reduce the time complexity of evaluating every pixel. The idea of locality in the objects and a sense of prediction in their possible position is implemented. We track the point location in the preceding frame and using calculated thresholds, we minimize the required search area. The size of this area is determined by the trends in the movement of the spot. If there is a sudden change in position, the search limits must be increased. Similarly, if the point appears to be near stationary, the area can be minimized. Hence, the minimum area to be searched is given as the magnitude of the displacement vector of the point. We increase this by a factor of 2.5 to account for accelerating behavior. The brute force method is a error free mechanism, but requires a time complexity of $O(n^2)$, where n would be an estimation of the length of the image. The algorithm has a linear time complexity, $O(n)$ and is also independent of the input size. Here, n is a scale of movement in the spot. This algorithm also allows distinction of spots, since they are independently tracked and hence their recognition criteria can be different. For example, the specific colors can be tracked simultaneously, which would not be possible through the brute force algorithm without making separate comparisons(in the case of similar criteria, top two values can be tracked and hence can be executed in a single cycle).

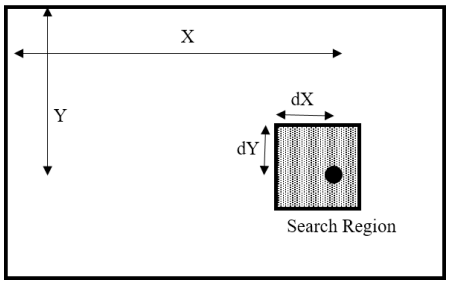


Fig. 1: Spot and Search area marked on a frame

IV. Algorithm

Finding the brightest spot.

Algorithm 1 Image Processing

1: Get frame from Video
2: Find search limit as- factor $* \sqrt{dX^2 + dY^2}$
Ensure: Min Limit $<$ search limit $<$ Max Limit
3: **if** searchLimit $>$ threshold **then**
4: Search complete frame
5: **end if**
6: Get area to be searched
7: Find position of maximum intensity with respect to center of the search area
8: Find position relative to complete frame
Ensure: $0 <$ Position $<$ Maximums

Execution of the program by capturing live video.

Algorithm 2 Video Capture

1: Initialize the webcam for Video Capture
2: **repeat**
3: Read single frame from camera
4: Analyses frame based on [Algorithm 1] and get position of spot
5: Draw circle marking the spot along with any necessary details on the frame
6: Display the new frame
7: **until** Halt command is given
8: Close camera and terminate program

Handling interrupts and input instructions.

Algorithm 3 Event Handling

1: **function** ON KEY PRESS(Key)
2: **if** Key equals Q **then**
3: Halt the program
4: **else if** Key equals R **then**
5: Reset spot using brute search
6: **else if** Key equals W **then**
7: Pause the program and wait for restart
8: **end if**
9: **end function**
10: Create listener for keyboard
11: Start the listener

The above algorithms are implemented in code to create a working model.

V. IMPLEMENTATION

The algorithm is implemented using Python programming language(see Appendix). We use the following modules-

OpenCV-	Short for Open Source Computer Vision, used as cv2 provides utilities to access live video from webcam, writing on frames and displaying the video.
pynput.keyboard-	Implementation of keyboard input and interrupt using listeners.
numpy-	Short for Numerical Python provides tools for analysis and operating on variables and structures.

A. Finding the Spot Location

The brightest spot is evaluated by finding the pixel with maximum value in a gray scale frame of the video [2]. We use *numpy.argmax([iterator])* which gives the index of maximum element in the argument passed. The iterator is a row wise flattened array. The index is unraveled to the 2-D frame using *numpy.unravel_index*. The following describes the working logic.

B. Finding Absolute Position

To find the position with respect to the complete frame, first we calculate the deviation of the spot loaction from the ceter of the search area. This is done by subtracting X and Y values with half the value of width and height respectively. This gives us dX and dY which can be added to the previous position to find the new position. The values of this position must be costrained between 0 and width of the frame for X, and 0 and height of the frame for Y. Further, this spot can be marked along with the search area using OpenCV.

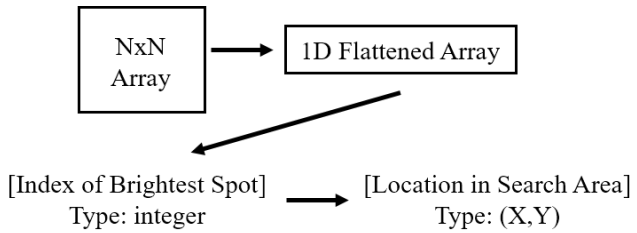
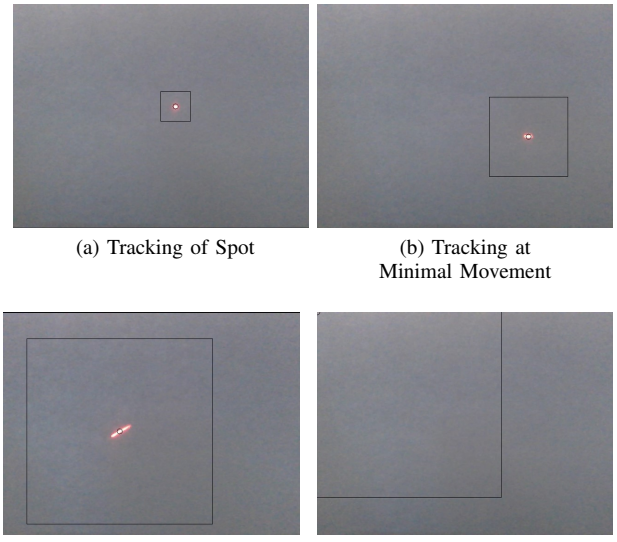


Fig. 2: Finding the Spot Location



(a) Tracking of Spot

(b) Tracking at Minimal Movement

(c) Tracking at Fast Movement

(d) No spot detected(Spot out of bounds)

Fig. 3: Output frame under different situations

VI. RESULTS

We find the spot being marked on consecutive frames with adequate accuracy and precision. It is seen that when the spot stabilizes, the search area is minimized [3a]. On increasing the resolution and frame rate, this area can be further reduced. On movement, the change in search area can be recognized. In [3c] the remnant of the laser spot, gives an idea of how fast it is moving. Hence the substantial search area. [3d] shows a frame when no prominent spot is present. On reappearance, an output similar to [3a] is obtained.

VII. CONCLUSION

The algorithm provides a fast alternative and is efficient in tracking a point spot on plane background. If there is noise in the image, it must be reduced such that the algorithm does not fixate at that point on overlap. The algorithm also allows implementation of spot distinction and simultaneous tracking using spot IDs. The functionality of this program can be comfortably increased while preserving the basic idea.

Acknowledgment

I would like to thank my professor, Dr. Vineeth V. for allowing me to research and develop on the topic of "Image Processing - Spot Tracking Algorithm." It has helped me extend my knowledge of the subject and understand its uses and applications. Creating this article has been a great learning experience and I am grateful for it.

References

[1] Itseez, *Open source computer vision library*, https://github.com/itseez/opencv, 2015.

[2] *The opencv reference manual*, 2.4.9.0, Itseez, Apr. 2014.

[3] M. Palmér, *Pynput package documentation*. [Online]. Available: https://pynput.readthedocs.io/en/latest/.

[4] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

Appendix

Code

```
import cv2 #Handles Graphics- Reading, Writing, Drawing
import numpy as np #General Computational Tools
from pynput import keyboard #Keyboard Control

#Setting up Camera
cam = cv2.VideoCapture(0)
result, frame = cam.read();

#Initializing Variables for Frame and Search
(HEIGHT,WIDTH) = frame.shape[0:2]
X,Y = int(WIDTH/2),int(HEIGHT/2);
searchLimit = int(0.01*WIDTH);
dX,dY = searchLimit,searchLimit

#Function to search complete Frame
def resetSpot():
    global X,Y,searchLimit
    Y,X = np.unravel_index(np.argmax(frame[:,:,0]),frame[:,:,0].shape)
    searchLimit = int(0.01*WIDTH)

#Event Handler
def on_press(key):
    print("Pressed: ",type(key));
    key = key.char
    if(key == 'q'):
        cam.release()
        print("PROGRAM COMPLETE");
        exit();
    if(key == 'r'):
        resetSpot();
    if(key == 'w'):
        input();
        return;

listener = keyboard.Listener(on_press=on_press)
listener.start()


print("STARTING")
resetSpot()
while(True):
    #Reading Frame
    result, frame = cam.read()
    frame = cv2.flip(frame,1);

    #Finding search limit and getting area to search
    searchLimit = int(np.clip(2.5*(dX**2 + dY**2)**0.5,0.02*WIDTH,HEIGHT))
    if(searchLimit > 0.9*HEIGHT):
        resetSpot();
    intensity = frame[max(0,Y-searchLimit):min(Y+searchLimit,HEIGHT),
                      max(0,X-searchLimit):min(X+searchLimit,WIDTH),
                      0]

    #Finding the Brightest Spot
    dY,dX = np.unravel_index(np.argmax(intensity),intensity.shape)
    dX -= 0.5*intensity.shape[0];
    dY -= 0.5*intensity.shape[1];
    X = int(np.clip((X + dX),0,WIDTH))
    Y = int(np.clip((Y + dY),0,HEIGHT))

    #Drawing and displaying
    cv2.rectangle(frame,(X - searchLimit,Y - searchLimit),(X+searchLimit,Y+searchLimit),1)
    cv2.circle(frame,(X,Y),int(0.01*WIDTH),(0,0,0))
    cv2.imshow("myImg",frame);

    cv2.waitKey(1)
    print("RUNNING");

print("OUT OF LOOP");
```