PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

UE21EC342AC1 - DIP 1

Report on

Tracking Centroid of Shapes

Submitted by

Divyansh Sharma (PES1UG21EC093)

Erisha Sarin Dias (PES1UG21EC095)

Aug - Dec 2023

COURSE INSTRUCTOR: DR. SHIKHA TRIPATHI

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PROGRAM - B.TECH

ABSTRACT:

The project involves a robust centroid tracking system capable of accurately identifying and tracking the centroids of various shapes within a frame. The project aims to implement image processing techniques to detect and differentiate shapes, and a real-time centroid computation. The goal is to create a reliable algorithm that can handle diverse shapes, varying sizes, and different lighting conditions for applications such as object tracking, motion analysis, and automated security surveillance systems.

INTRODUCTION:

Problem Statement Explanation:

- Real-Time Tracking: The primary goal is to track the movement and changes in the centroids of shapes in a live video stream.
- Shape Recognition: Identify various shapes (e.g., circles, squares, polygons) within the video feed.
- Centroid Calculation: Determine the center points (centroids) of these shapes as they move or change.

Existing Work:

- Computer Vision Algorithms: Techniques such as image segmentation, edge detection, contour analysis, and feature extraction are commonly used.
- Tracking Algorithms: Kalman filters, particle filters, and other predictive algorithms might be utilized for smoother and more accurate tracking.
- Machine Learning Models: Deep learning networks like CNNs (Convolutional Neural Networks) for object detection and tracking.
- Software Packages: Similar efforts might have used MATLAB, OpenCV, or custom libraries for image processing and analysis.

Introduction to Approach:

- Image Acquisition: Utilizing the IP cam package to capture a live video feed.
- Preprocessing: Filtering, noise reduction, and thresholding to enhance the quality of the video feed for better shape recognition.
- Shape Detection: Employing algorithms to detect and identify shapes within the frames.
- Centroid Calculation: Once shapes are identified, calculating their centroids.
- Tracking Algorithm: Continuously updating and tracking the centroids across subsequent frames for real-time monitoring.

THEORY AND ALGORITHM:

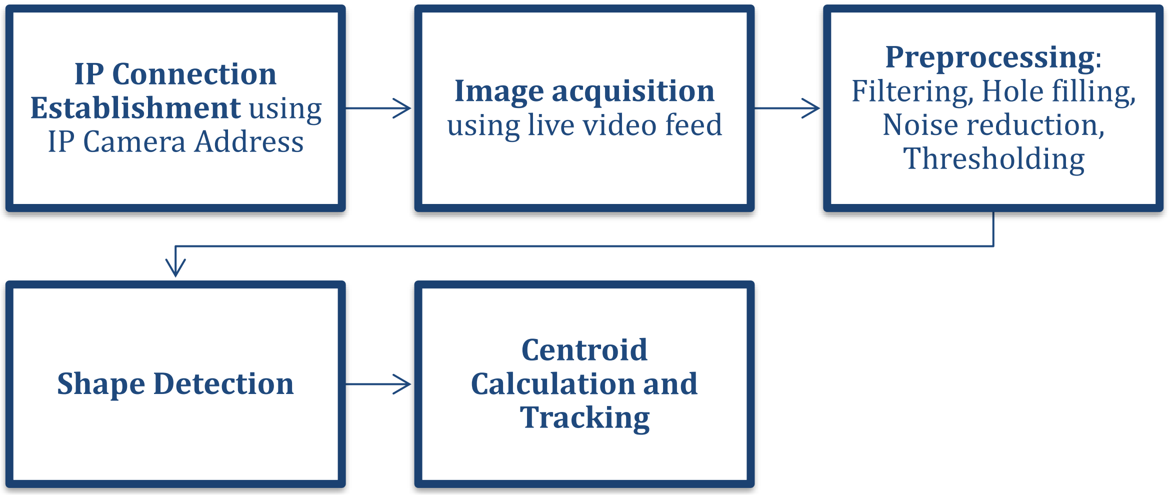
In the project for real-time centroid tracking of shapes, several Digital Image Processing concepts and respective algorithms have been employed:

1. Image Representation:
   - Grayscale Conversion (rgb2gray): Converts the RGB image to a grayscale image. This simplifies subsequent processing by reducing the image to a single channel representing intensity.
2. Image Enhancement and Thresholding:
   - Binarization (imbinarize): Converts the grayscale image to a binary image based on a threshold value. This simplifies shape identification by turning pixels into pure black or white, separating shapes from the background.
3. Morphological Operations:
   - Hole Filling (imfill): Fills the holes within shapes, ensuring complete, closed regions for accurate shape detection.
   - Noise Reduction (bwareaopen): Eliminates small objects (less than a specified area) that may not correspond to significant shapes. This reduces noise and irrelevant information in the image.
4. Connected Component Analysis:
   - Connected Component Labeling (bwlabel): Identifies connected regions in the binary image and assigns unique labels to each distinct shape or region. This is crucial for differentiating and tracking individual shapes.
5. Shape Analysis and Centroid Calculation:
   - Shape Area Calculation: Iterates through the labeled regions to compute the area of each shape. This area calculation helps in sorting and identifying the largest shapes.
   - Centroid Calculation: Determines the centroid of each identified shape by computing the mean of the row and column coordinates of the shape's pixels. This helps in tracking the position of shapes in the image.

Algorithm Challenges and Considerations:

- Assumptions: The algorithm assumes a minimum of four distinct shapes in the frame and focuses on tracking the largest ones.
- Sensitivity to Noise and Lighting: Variations in lighting and noise in the environment might affect the accuracy of shape detection and centroid calculation.

FLOWCHART:


IP Connection Establishment using IP Camera Address
Image acquisition using live video feed
Preprocessing: Filtering, Hole filling, Noise reduction, Thresholding
Shape Detection
Centroid Calculation and Tracking

CODE:

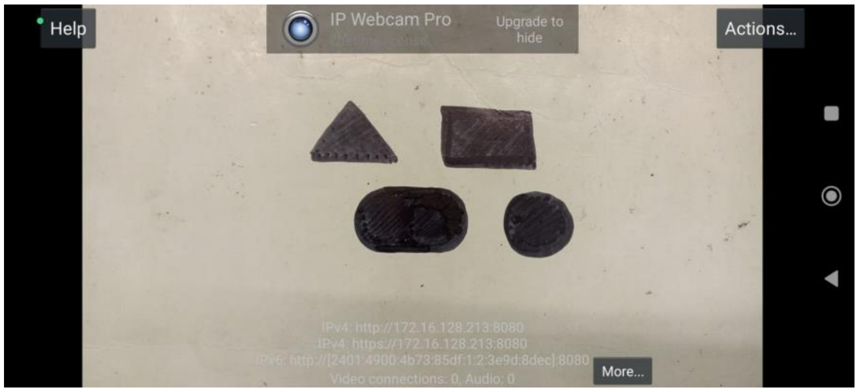
```
clc
clear all
close all
warning off

% connecting to phone IP camera using its IP address
mycam = ipcam('Give IP address..'); % type down ip address of ip cam device

% Infinite loop to capture and process frames
while true
    % capture a snapshot
    e = mycam.snapshot;

    % RGB to grayscale
    ks = rgb2gray(e);

    % grayscale to a binary
    mg = ~imbinarize(ks);

    % Fill holes in the binary image(noise reductions)
    mg = imfill(mg, 'holes');

    % Remove small connected objects that have lesser than 10 pixels
    mg = bwareaopen(mg, 10);

    %number of connected components in the binary image(bc), L-> matrix
    [a, bc] = bwlabel(mg);
```

```
% Initializing variables for counting pixels in each recognised component
b = [];
temp = 0;
gk = 1;

% Get the size of the labeled matrix
[f, g] = size(a);

% Count pixels in each recognised object
for i = 1:bc
    for wk = 1:f
        for tk = 1:g
            if (a(wk, tk) == gk)
                % count number of pixels in each object(area)
                temp = temp + 1;
            end
        end
    end
    %append to array the number of pixels in each connected component
    b = [b temp];
    temp = 0;
    gk = gk + 1;
end

% vector of levels corresponding to number of objects
level = 1:bc;

% Sort the connected components based on the pixel count
for i = 1:length(b)
    for j = 1:(length(b) - i)
        if (b(j) > b(j + 1))
            temp = b(j);
            b(j) = b(j + 1);
            b(j + 1) = temp;
            gemp = level(j);
            level(j) = level(j + 1);
            level(j + 1) = gemp;
        end
    end
end

% If there are at least 4 connected components
if (bc >= 4)

    b_heighest_5 = [];
    level_heighest_5 = [];

    % Select the top 4 connected components
    for i = length(b):-1:length(b) - 3
        b_heighest_5 = [b(i) b_heighest_5];
        level_heighest_5 = [level(i) level_heighest_5];
        end

        % Initialize vectors to store the centre coordinates
        center_x = [];
        center_y = [];

        % Calculate the centre coordinates for each
        for n = 1:4
            bmap = a == level_heighest_5(n);
            [rows, cols] = find(bmap == 1);
            rc = mean(rows);
            cc = mean(cols);
            center_x = [center_x (rc)];
            center_y = [center_y (cc)];
        end

        % Display the original image and the centre coordinates
        imshow(e);
        hold on;
        plot(center_y, center_x, 'w*', 'linewidth', 10);
        drawnow;
        clf;  % Clear figure
    else
        % display original image otherwise
        imshow(e);
    end
end
```

RESULTS AND OUTPUT SCREENSHOTS:

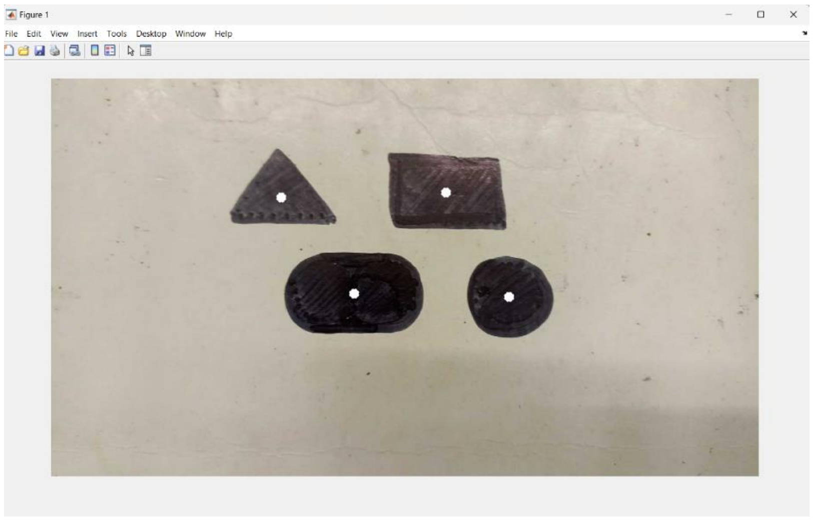
- The code continuously captures snapshots from the specified IP camera and converts the RGB image to grayscale, creates a binary image, fills holes, and removes small connected components.
- As shapes move or change position within the video stream, the centroid tracking algorithm dynamically adjusts and accurately follows their movements.
- The process repeats in an infinite loop, continuously capturing and processing new snapshots from the IP camera.

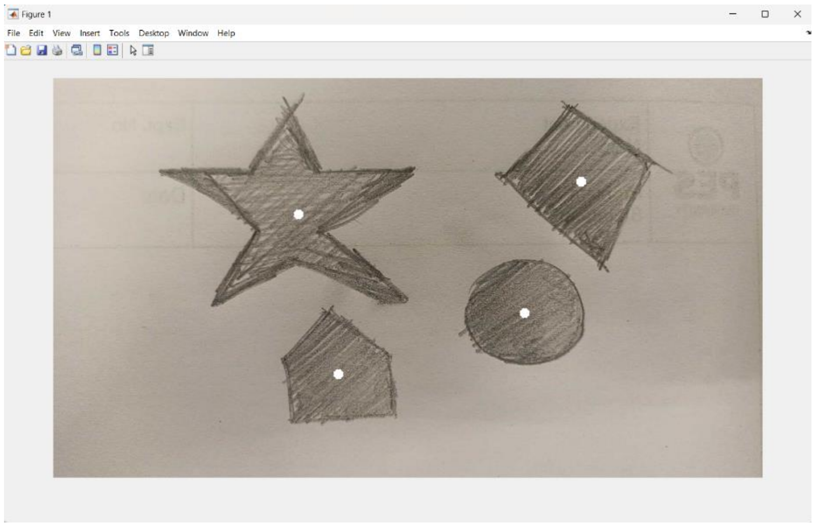
1. IP Cam App in Mobile Phone capturing the live video feed, The IP Address shown in the bottom is to be used by the code.



```
IPv4: http://172.16.128.213:8080
IPv4: https://172.16.128.213:8080
IPv6: http://[2401:4900:4b73:85df:1:2:3e9d:8dec]:8080
Video connections: 0, Audio: 0
```

2. The code when run, shows the live video feed captured by the IP Cam app, calculates the centroids, and tracks them in real time.





OBSERVATIONS AND CONCLUSIONS:

The project successfully achieves real-time centroid tracking of shapes using MATLAB and the IP cam package. Through a series of digital image processing steps including grayscale conversion, binarization, morphological operations, and connected component labeling, the algorithm accurately identifies shapes and computes their centroids within the live video feed. By plotting the centroids on the video frames, the tracking mechanism showcases the effective real-time visualization of shape movements. The implementation of this project helps us understand DIP concepts and apply them practically for basic shape detection and tracking, demonstrating the viability of these techniques in real time applications in surveillance, object detection, robotics, etc.

REFERENCES:

1. MathWorks:
   - https://in.mathworks.com/discovery/digital-image-processing.html?requestedDomain=
   - https://in.mathworks.com/help/supportpkg/ipcamera/ug/ipcam.html
   - https://www.mathworks.com/solutions/image-video-processing/object-recognition.html
   - https://in.mathworks.com/matlabcentral/answers/328060-program-for-tracking-the-moving-object-in-an-video-by-centroid-method
2. Research Gate:
   - https://www.researchgate.net/publication/283856859_Tracking_multiple_moving_object_based_on_combined_color_and_centroid_feature_in_video_sequence