Tracking Objects in a Video by Blob Matching

Introduction

There are myriad applications of object tracking in real life. Also, there are many techniques for tracking objects of interest from the stationary cameras. Tracking systems have been developed for many applications like virtual reality, and for those measuring athletic performance need to robustly track specific body parts. The tracking system is used to monitor moving objects and to provide the position information in a prompt order sequence for further processing and is one of the important components of many vision systems. It has numerous applications in traffic control, human-computer interaction, digital forensics, gesture recognition, augmented reality, and visual surveillance. A system that can perform simultaneous tracking on all different scales is highly desirable, but until now, no such system exists. The work described in this project targets the second category of applications [1][2].

This project is based on the study method proposed in the paper [12]. There are three process levels we need to explore. The level I-II needed to implement for the main tracking system, the level III is an advanced learning goal to build a zebrafish moving model using Extended Kalman Filter. The Extended Kalman Filter is the non-linear version of the Kalman filter. The Extended Kalman Filter predicts the future positions based on the object's current position [3]. The Extended Kalman Filter is very popular for tracking objects and predicting future positions.

The approach in this project does not have any restriction on the position of the camera and does not make any assumptions about occlusions nor overlaps. The project uses a single fixed camera mounted at an arbitrary position. We use simple rectangular patches with a certain dynamic behavior to model blobs. Overlaps and occlusions are handled by allowing blobs models to overlap in the image space and by maintaining their existence despite the disappearance of some cues. The cues that we use are blobs obtained by thresholding the result of subtracting the image from the background. Then the blobs are compared with the previous frame to track the blobs. A detailed explanation is given in the proposed method section.

Problem Specifications

Zebrafish have been broadly connected in numerous areas. The exact distinguishing proof and tracking of people are significant for zebrafish shoaling conduct examination. In any case, multi-zebrafish tracking still faces numerous challenges, such as comparative shape, development, and visit impediment. It is troublesome to keep recognized for a long time due to angle covering caused by the intersections. An arrangement of computer vision tracking strategies was proposed to unravel these issues.

These strategies are widely divided into two categories: the tracking strategies based on detection-data affiliation and strategies based on identification. While the issue of discovery and tracking of angle school is related to the multi-target discovery and tracking issue within the field of computer vision and design acknowledgment, it has solid one-of-a-kind characteristics making it challenging and worth exhaustive examination. In detail, we are confronted with the taking after two difficulties.

1. Detection issue:
    a. The fishes’ texture information not enough to detect and its location can’t be detected effectively only by texture features.
    b. The shape of the fish may represent by one or several templates.
    c. When the fish density is large, the targets will frequently occlude each other in the image. So, it is difficult to detect the location of each target.
2. Tracking issue:
    a. Cause the higher degree of similarity between the fish, when the use of a single feature method may not distinguish between different targets.

b. The motion or direction of fish swimming is so complex that an event humans cannot predict.
c. The detecting errors caused by fish occlusion will lead to a fragmentation in trajectory, adding more difficulties in tracking.

Review of Prior Work

While we were reading the paper we had to implement, reading brought up in the abstract that its own objective was to detect and its own approach to track humans in front of other vehicles. we were really into it and got more depth that cause we are now tracking the fishes instead of humans. It was cool because we read article which was interested knowing how the tracking works and everything because there’s so many different way’s right now in the world that revolves around tracking including security issues, cameras and tracking objects.

In the article “Cameras Work Together to Track a Single Person" Futurity, 2014 and the writer Ma Michelle, the article writer talks about is that there are new ways to track humans across other cameras in different way [4]. It connected well of the idea just to go more in depth as simple with the example of new GPS display that was brought up and it adds humans onto the whole view of the picture. So, while reading we were very interested because we worked around many different cameras. We were reconnecting this article to our project because we see same way on tracking. Prior work relates to the general problem because you know we are tracking fishes and the article is tracking humans on the GPS. So, we have that advantage of tracking anything but there are more concerns with the issues overlapping as well. That was one of our disadvantages in our project because the fishes would overlap, and we would have to revisit the code many times to fix this issue.

In the paper “Computer Vision for Tracking” towards data science and the writer Cohen Jeremy illustrates many points on how detection works [3]. In the paper there is picture that evolves box around cars, box, person and traffic lights. There were two algorithms used in this paper but more importantly we wanted to see how Extended Kalman Filter algorithm was effective at all. So, we were confused why we needed the Kalman filter at all? So, we read this paper as well to go more in depth in learning and it says for Kalman filter that it can actually predict the future position based on current position, so we got more acknowledge on why Extended Kalman filter was more useful.

During the project, we contribution equally. We generally get on Zoom meeting to start the coding and help each other on different parts. We divide few parts such as Amit worked on the Image processing, video capture, contours, blobs frame, extract blobs, rectangle patch, resize image, threshold fixing, velocity function, Matplotlib to visualization the graph, overlap area, relating fish to blobs and calculating fish positions. On the other hand, Rajdeep worked on the image processing, extract blobs, optimum graph, bipartite correspondence graph, blob vertex, bounding box, cost function, threshold setting, gaussian blur erosion, prediction, estimation, and refinement in the Extended Kalman filter.

Describe the solution methodology

The proposed solution consists of three main levels, the image level, the Blobs, and the Extended Kalman Filter (EKF). They all contribute together for accurately tracking the detected objects in need.

During the Image level, the frames of the video are being captured and being passed for processing one by one in the correct sequence. Hence, the background of the images is needed to be removed assisting the blob method that follows. To avoid having unwanted blobs due to unwanted background noise, a threshold value is initially applied. This threshold is empirically found using a blank background for a certain period, examining the pixel value variations during the time of testing. Then, a Gaussian filter is applied so further noise will be eliminated from the background.

Following on, since background noise is eliminated, using the Contours algorithm, the blobs of the current frames are obtained. Hence, area, density, bounding boxes and locations of the blobs are calculated. These blobs are then compared with the previously obtained blobs from the previous frames to create a relationship between them. Upon correlating the blobs from previous frames with the current ones, the vertices of their position are calculated helping to develop an undirected bipartite graph, Gi(Vi,Ei), where Vi represents the blobs in each i frame and Ei represents the sets of vertices associated with the blobs in frame i. The blob graph is also used to assist in splitting blobs between consecutive frames. Also, to compare different graphs, a cost function C(Gi) is then defined to find the optimum Gi. Calculating the cost generates a parent set Pi, which is a set of all detections in the previous frame, and descendent set Di, which is a set of detections in the current frame related to the previous one. The 1,2,3 equations represent the calculations used. Equation 1 is the Descendent set Di, where Ni represents the neighbors set. Equation 2 represents the total area of descendants for each parent separately (Si(u)), where A represents the area function, and finally, 3 is the cost equation.

$$(1)\quad D_i = \bigcup_{u \in Pi} N_i(u) \qquad (2)\quad S_i(u) = \sum_{v \in N_i(u)} A(v) \qquad (3)\quad C(G_i) = \sum_{u \in P_i} \frac{|A(u) - S_i(u)|}{\max(A(u), S_i(u))}$$

Furthermore, the velocities of the blobs are calculated or correlated to each blob between frames as follows. If a blob is using splitting, the velocity of the parent blob is inherited in the current frame. In the case of a blob created by merging operation, then the parent's blob the maximum velocity is inherited in the current frame. Otherwise, the velocity is calculated using equation 4, where b is 0.5, dt is the time difference, bu and bv is the bounding box's center in the prior frame.

$$(4)\quad V(u) = \beta \frac{(b_v - b_u)}{\delta t} + (1 - \beta)V(u)$$

The final level of the proposed methodology is the Extended Kalman Filter [9]. For each blob, the EKF is initialized with its initial position of the blob. Then, in succeeding frames, a prediction of the state vector and state error covariance matrix is calculated using the prior frame information. This results in predicting the next position of the blob. Equations 5,6,7 depict the EKF calculations where x is the predicted state vector, P is the state error covariance matrix, Q is the Gaussian process noise and q represents the acceleration variance.

$$(5)\quad \begin{aligned} x_{t+1} &= F x_t \\ P_{t+1} &= F P_t F^T + Q \end{aligned} \qquad (6)\quad Q = \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix} q \qquad (7)\quad \mathrm{A} = \begin{bmatrix} (\delta t)^3/_3 & (\delta t)^2/_2 \\ (\delta t)^2/_2 & \delta t \end{bmatrix}$$

Upon calculating the predicted positions, the overlap area of the blobs with the predictions is calculated as well. It is then used as a measurement error standard deviation σ and finally the EKF is updated using equations 8,9, where h is a measurement function and w/t is the sequence of zero-mean.

$$z_t = h(x_t) + w_t \quad (8)$$

$$\begin{aligned} K_{t+1} &= \hat{P}_{t+1}H^T(H\hat{P}_{t+1}H^T + R_t)^{-1} \\ x_{t+1} &= \hat{x}_{t+1} + K_{t+1}(z_{t+1} - h(\hat{x}_{t+1})) \\ P_{t+1} &= (I - K_{t+1}H)\hat{P}_{t+1}. \end{aligned} \quad (9)$$

Finally, some rules are being applied to refine the blob-object relationships. Threshold of 10% overlap area to split a blob and assign a new object. A blob-object match is assigned if the overlap area is above the threshold of 10%. New blobs are assigned as a new object and finally if blobs are merged it is then assigned as a new object.

Proposed Method (300+ words)

The proposed method of the author in paper using blobs obtained after background subtraction is motivated by the efficiency of this preprocessing step even though some information is permanently lost. In particular, a blob obtained this way does not always correspond to a single fish. The relation between blob and fish was allowed maximum flexibility to be many to many.
There are three levels of abstraction used. The first level deals with raw images which have been done by a dataset, which is the grayscale images and binary images. In the second level blobs, we collect all blobs in each frame by the morphology processing with binary image, such as closing, opening, erosion, and dilation [8]. Blobs extracted are computed and subsequently tracked. Tracked blobs are passed on to the zebrafish level where relations between zebrafishes and blobs as well as information about fishes are inferred using previous information about zebrafishes in that level.

- The first at the images level, we perform background subtraction and thresholding to produce different images. Choosing a threshold is an important thing. The images dataset we have could be thresholder contains two categories of pixel values and we will try to separate two. That is the fish and the background. Several measures were taken to further reduce the effect of noise. A single step of closing to fill the hole, remove noise with opening, erosion followed by a step dilation if performed on the resulting image and a small cluster are totally removed [8].

- The second level was called Blobs level. This level performer the motion of blobs and by allowing blobs to merge, split, appear, and vanish. The following parameters are computed for each blob b.
    - area - A(b): the number of pixel inside the blobs
    - bounding box - the smallest rectangle surrounding the blobs
    - density - D(b): A(b)/bounding box area
    - velocity - V(b), calculated in pixels per second in horizontal and vertical directions.

- The third level was called Fish level. The Extended Kalman Filter used to predict the future positions based on the current position [9]. This method includes five sections which describe the tracking cycle. The five main parts are relating fish to blobs, prediction, calculating fish positions, estimation and refinement.

Innovations outside project guidelines

In order to get an image enhanced or if useful information is needed to be extracted, image processing is used. It is a signal processing method, in which the input is an image, and the output may be the image's features of that image. Hence, image processing includes three steps. The import of the image is the first step, followed by the analyzing and manipulation of the image and finally the output which results in image analysis or an altered image. In this work, in order to execute image processing functions in python, OpenCV library is being used. OpenCV (Open-Source Computer Vision Library) is an open-source library that consists of over 2500 optimized computer vision algorithms and state-of-the-art machine learning algorithms [5]. It is widely used and really well contributed by the community that's why it was chosen. Also, OpenCV library offers tools for optimized frame by frame video capture and functions that are able to resize or change color of each frame if needed. Some of the algorithms used in this project are Gaussian Blur and finding Contours. Gaussian Blur is an image filtering process which helps in removing the noise in an image [11]. The difference of Gaussian blur from other filtering processes is that it uses Gaussian kernel instead of a box filter. Hence, it helps remove unnecessary background noise from our images, such as the lines of the boxes, and lets us detect only the objects in need. In this project OpenCV used to read video and file from local files. We do several image processing with data. We performance morphological transform, finding contours to obtain the area, bounding box, resident or calculated velocity.

Furthermore, the contour function is used to get the location of objects (detection of the objects) [6]. Contours can be described as a curve that joins all the continuous points which have identical color or intensity. It is used mostly for refining objects and identifying them as unique objects as blobs. Before applying Contours, the image must be initially in grayscale and apply a threshold for a better refinement of the edges of the objects. Both of the methods are applied in this work for assisting Contours, but a threshold for the images seems to output better results during some of the experiments conducted.

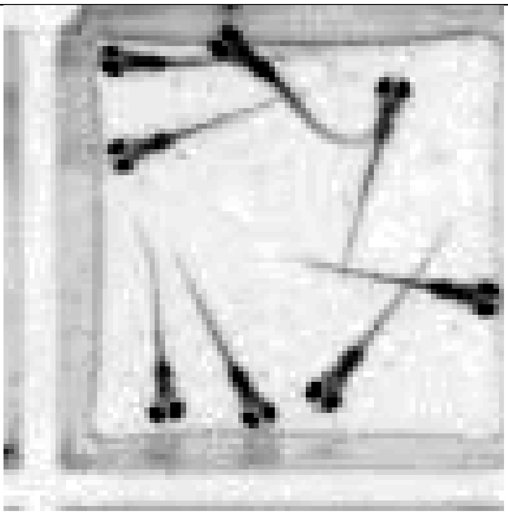
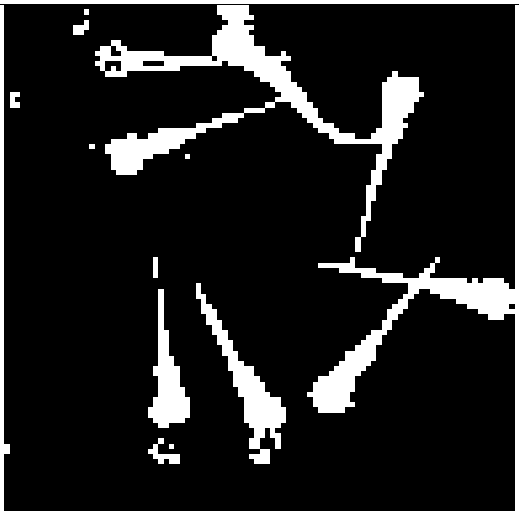
Graphs are also implemented and used in this project. They are used to relate the blobs extracted from the Contours, with the vertices of the position of the blobs in previous frames. It is represented by an undirected bipartite graph, Gi(Vi,Ei), where Vi is Bi U Bi-1. Bi are the sets of vertices of x, y positions of the blobs from the previous frames. For implementing graphs in python, Matplotlib is used. Matplotlib is a data visualization and graphical plotting library for python [10]. It offers an open-source alternative of MATLAB and provides an object-oriented API for plots.

In this project, NumPy [7] was used calculus matrix array complexity which focus on calculated Extend Kamal Filter and do basic calculation in the blob level [9]. NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, and sorting.

Data and output

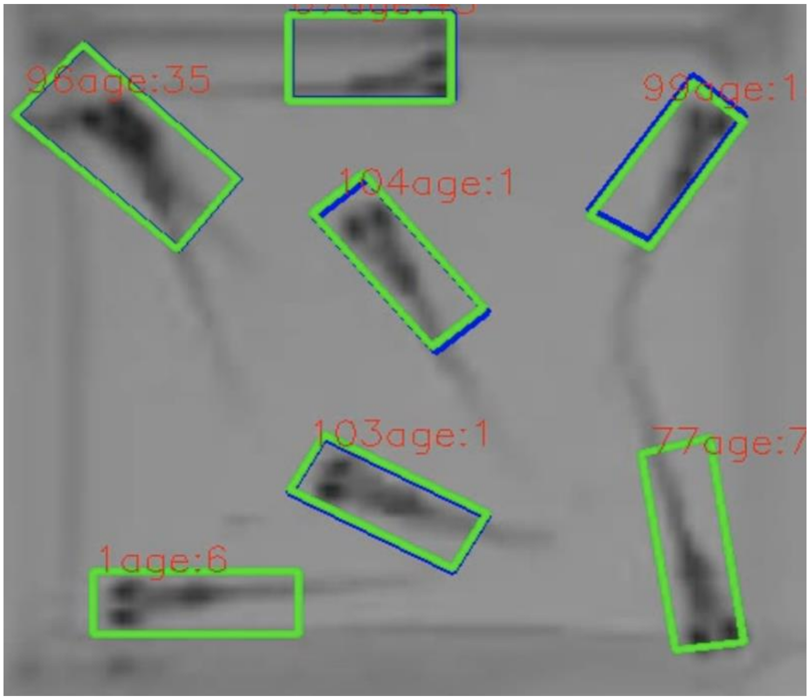
Zebrafish dataset contains two folders of grayscale, binary images, and one 13 min video. The binary images are processed with background subtraction, each pixel of it has value 0 or 1. There are 300 frames in each folder, each frame has 8 to 10 zebra fishes placed in a small box with size of each frame is 96 x 98 (width x height). The area avenger of each fish is about 25 pixels, this area measured of the blobs when did processing before. There are many scenarios moving with complexity motion in this data example, the place is smaller than the fish and has less place to swim, the trajectory of each fish is quite difficult to predict. They are usually occluded and overlapped to each other. Also, there are some fish that do not move the whole frames.

Example Grayscale image and Binary image from folder data.

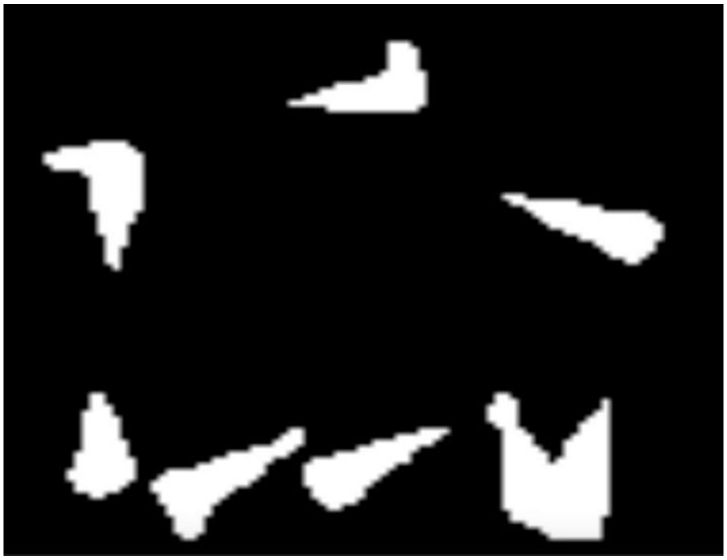
Grayscale Image	Binary Image
	

Program output

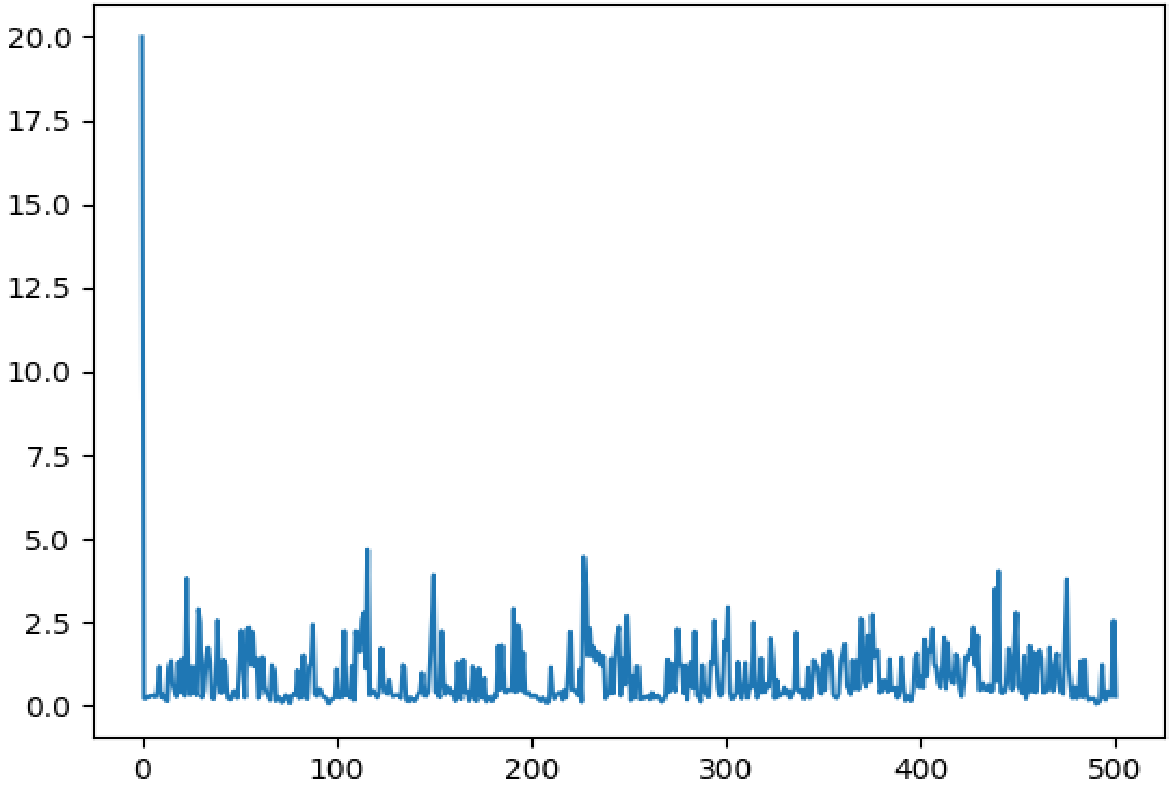
The final project output has three different results. The three main results include tracking video output from the video data, correspondence graph, and blob graph.



In this output, we see two different box colors. Green box is prediction position and blue box is the current or original position.



The second output contain the correspondence graph which is binary image. In each frame background is removed to get the blobs. In the “Experiments and Analysis” section, I will explain more about this part and the main issue which we are getting according to the program.

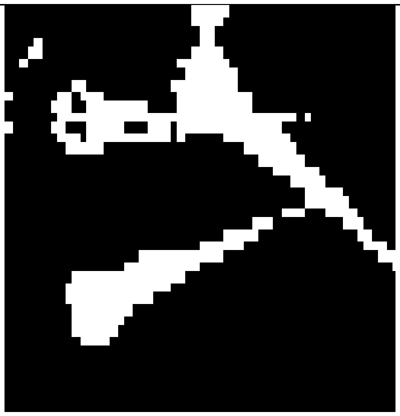
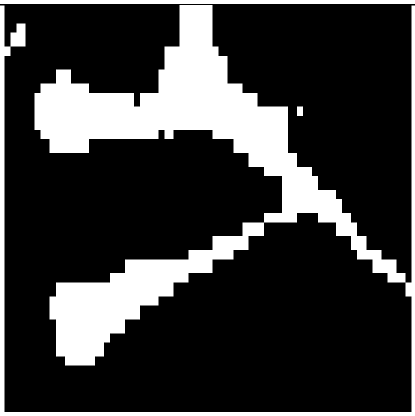


This blob graph show the (x,y) graph the blob. x is the cost value and y are the frame index. Use the blob graph, the cost of current frame is calculated. Cost value is the C(Gi). Left side is x value and bottom line contain y value.

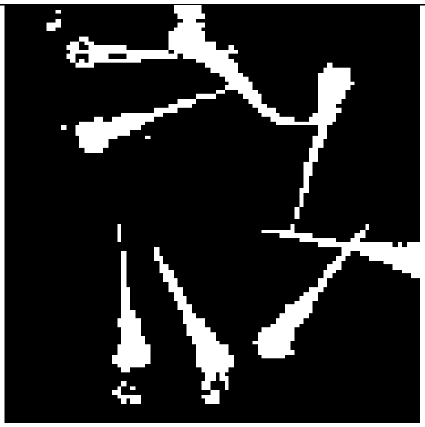
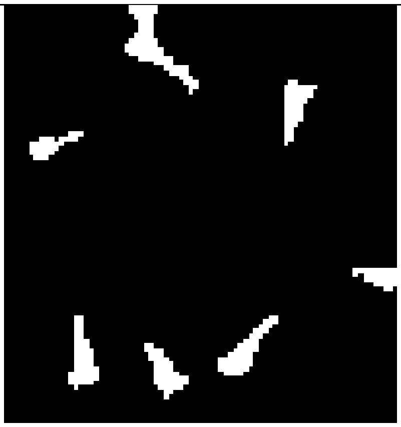
Experiments and Analysis

In this project, the preprocessing image to get all possible blobs in the frame binary is very important. Because we need to satisfy the requirement to reduce noise and separate as much as blobs which have area contain the fish to each other. Then we can build a graph quickly with low cost. There are four processing using morphology that will be useful [8].

- Closing: It is obtained by the dilation of an image followed by an erosion [8]. It is useful in closing small holes or dark regions inside the foreground objects, or small black points on the object.

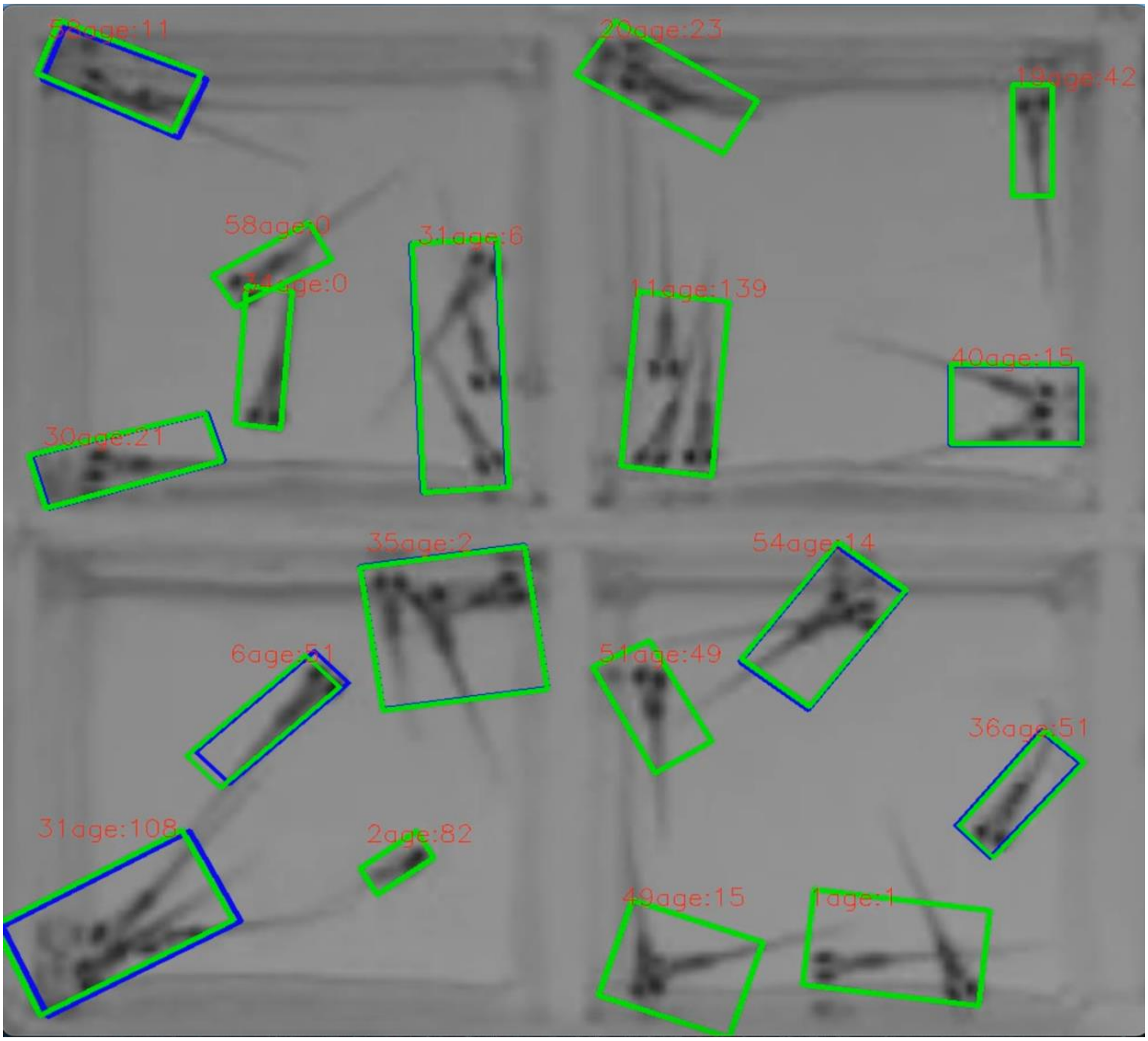
Before	After
	

- Erosion: The basic idea of erosion that erodes away the boundaries of foreground (keep foreground in white) objects [8]. The kernel slides through the frame. A pixel in the original frame (either 0 or 1) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to 0). All the pixels near the boundary will be discarded depending upon the size of the kernel. So the thickness or simply white region decreases in the frame. It is good for removing small white noises, interspace two connected objects etc.

Before	After
	

After doing some morphological, then using the function findContours() of OpenCV to detect blobs in images. Setup blobs parameters and could be added to the graph. We found out about the contours

The main issue we are getting is about the merging. In the below output image, it shows that the multiple fishes have one rectangle patch. The reason for having one single rectangle patch for some fishes because two or more fishes walking together or overlap each other rise to one huge blob. The contour function is used to get the location of objects but, contour areas will not overlap [6]. Our understanding about this issue and solution is that contour find the location of object and then use the rectangle patch on the white area. When multiple blobs come together, it rises to one big white area and contour draw the rectangle patch on it. In order to fix these issues, we need to use different approach in which we train the data which come under the deep learning. I found the solution of this problem but, it takes so much time to train the data. This video has around 20,000 frame and around 9600 fishes in frame. We need to train all fishes in each frame which is the solution but again training the data only work on this video. This contour work on any type of objects and conditions. This issue explained in paper section 3 but, not well explanation how to fix this issue. The paper just talks and explain how we can handle the issue but no implement information.



Zebrafish blobs which do not overlap with each other in each frame have areas from 20 to 30 pixels. This can be used to analyze the size and shape of fish. They're set the threshold to determine valid blobs in a frame. The other method used to get blobs is changing detection or by comparing two consecutive frames. But it will show unwanted blobs of the background being uncovered by moving objects. So, a threshold is applied on the change of image to get binary image or blobs image. The threshold value was obtained by examining an empty background for a short while and measuring the maximum difference in pixel values during this training period. Then the threshold is set to be slightly above that maximum difference value.

Method 1

$$C(G_i) = \sum_{u \in P_i} \frac{|A(u) - S_i(u)|}{\max(A(u), S_i(u))}. \qquad (1)$$

The first equation was used to determine which graph is most consistent to represent the relations between previous blobs and current blobs. We use this equation after collecting all possible blobs in the present frame and with the previous blobs forwarding to the current stage. We may build more than one valid graph and need to choose one of them. Frame by frame were extracted from the video and will have a specific graph.

(Assume all the terms, symbols we are reading explain detail from the papers) The equation (1) calculated the cost (C) graph (G) at frame i. This function is a summation of ratios of size change over all parent blobs. The cost function that we use penalizes graphs in which blobs change significantly in the size. To calculate it, first we need to set up the set (P) parent vertex in the graph G. We travel all the vertex in P, each vertex (blob) we calculated its ratios of size change with numerator is absolute subtraction between area of its box (A) and the area of total area (S) occupied by the neighbors of that vertex and denominator is max value area between A and S. To implement this, we will structure the graph with each vertex is a blob, blobs have their own parameters including the area of it, also the vertex will determine it is the parent vertex or descendent, or both."

We generate the graph based on the two constraints which is parent and locality constraint. After that, we find the optimum within these graphs using the cost function. The cost function represents the penalization. We get high cost when the blobs change significantly between consecutive frames. We got hard time constructing the set of graphs between consecutive frames. The paper skipped basic information because this project is based on expert programmer but for beginner it gets hard when we require basic background information.

Method 2

$$\mathbf{V}(v) = \beta\frac{(\mathbf{b}_v - \mathbf{b}_u)}{\delta t} + (1-\beta)\mathbf{V}(u) \qquad (2)$$

At the end of blob level, we use the second equation to calculate the velocity of each blob v based on the velocities of the blobs at the previous stage and the computed blob graph. But not at all, if the blob outcome of a splitting operation, it will be assigned the same velocity as the parent blob. If the blob outcome of a merging operation, it will be assigned the velocity of the largest child blob. This equation was used only when one blob related to it, then the velocity is computer based on the second equation. The equation was used at the end of the blob level, blob velocity will be used to initialize zebrafish models later

This equation was used only if there is only one blob, u related to v. To calculated it, we need implement the argument have in the equation, first the weight factor β is set to 0.5. bv and bu is centers of bounding boxes v and u respectively, they can be easily had by the coordinates of bounding boxes each blob. The delta t time is a sampling interval since the last stage that is subtraction between start time of this level and end time when finish the optimum graph to finding the best graph before.

Note that the equation does not need to be used when v is the outcome of a splitting operation, then v assigned to the same velocity as the parent blob. Or if v is the outcome of a merging operation, it will be assigned the velocity of the largest child box.

Method 3

$$(8) \quad \mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{w}_t$$

$$(9) \quad \begin{aligned} \mathbf{K}_{t+1} &= \hat{\mathbf{P}}_{t+1}\mathbf{H}^T(\mathbf{H}\hat{\mathbf{P}}_{t+1}\mathbf{H}^T + \mathbf{R}_t)^{-1} \\ \mathbf{x}_{t+1} &= \hat{\mathbf{x}}_{t+1} + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{h}(\hat{\mathbf{x}}_{t+1})) \\ \mathbf{P}_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})\hat{\mathbf{P}}_{t+1}. \end{aligned}$$

In the fish level which is advanced learning goal of this project. This is based on Extended Kalman Filter [9]. Some parts of Zebrafish were transparent which is difficult to predict. Predict the pedestrian is easy because of certain dynamic behavior. In our case, zebrafish fish move so fast and predict the future position is very hard to do. We estimate the fish parameters in use of extended Kalman filtering. The Extended Kalman Filter is initial position of the blob. The Extended Kalman Filter used to predict the future positions based on the current position. This method includes five sections which describe the tracking cycle. The five main parts are relating fish to blobs, prediction, calculating fish positions, estimation and refinement. This equation helps to initial the location of the object and then predict the next location. I am using the update (using equation 8 and 9) function which update its values and new coordinates.

Conclusion

The main purpose of this project is to implement the algorithm that presented in the paper [12]. In this project we have studied how to use OpenCV in several methods of computer vision to preprocess image or video. The result from tracking blob was run as expected because following build graph relationship between blobs previous and next frame in a sequence timestamp in the paper reference. But with challenge from the dataset exists. Many objects occlusion or overlap to each other. The tracking method based on Extended Kalman filter have less power to have the best performance.

The goal of this project is to study, deep understand a tracking system, therefore we can implement the object tracking method based on establishing correspondences between segment (binary) regions across frames. With a number of factors, the semantic object may not correspond to a single binary region, or a single binary region may contain more than one semantic object. Zebrafish dataset was used to practice in this project. Dataset includes grayscale, binary images and a video. We followed the paper which track pedestrians in the scene. This mean fish and objects in the scene tracked as a pedestrian. If I made a replacement of contour with fish detection, then this program track all the objects in the scene as a fish. This project helps us learned lots about python, tracking, detection, image processing, etc. Also, it helps us to manage the time better. We did best we could, went through stressful times but still managed to get our project done. We were very interested to know about tracking system and that was the main reason we picked this project. We are planning to work more on this project and make more advancement in the program.

References

[1] N. Johnson and D. Hogg, “Learning the distribution of object trajectories for event recognition,” Image Vision Computing, vol. 14, no. 8, pp. 609–615, Aug. 1996.

[2] R. Hosie, S. Venkatesh, and G. West, “Detecting deviations from known paths and speeds in a surveillance situation,” in Proc. 4th Int. Conf. Control, Automation, Robotics, and Vision, Singapore, Dec. 1996, pp. 3–6.

[3] C. Jeremy, “Computer Vision for Tracking”, towards data science, 2019

[4] M. Michelle, "Cameras Work Together to Track A Single Person”, February 2014

[5] Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. " O'Reilly Media, Inc.".

[6] R. Sovit, “Contour Detection using OpenCV (python/C++), March 2021

[7] H. Arman “Top Python Libraries: Numpy & Pandas”, November 2019.

[8] S. Ravjot, “Morphological Transformations of Images using OpenCV | Image Processing Part-2” August, 2020

[9] S. Harveen, “Extended Kalmen Filter: why do we need an Extended Version?” April 2018.

[10] S. Brad, “Python Plotting with Matplotlib (Guide)” March 2019

[11] F. Tony, “Gaussian Blurring with Python and OpenCV” March 2019

[12] M. Osama, Nikolaos, “A Novel Method for Tracking and Counting Pedestrians in Real-Time Using a Single Camera” September 2001