Gabriel Pellegrino da Silva
Damoun Ayman
Object Tracking github repository
Master M2 IA

OPT10
Image Mining
November 2, 2020

Lab 3: Object Tracking in Videos

0.1 Preliminaries

The aim of this work is to performe objet object tracking in videos and understand the challenges and difficulties of object tracking this form of data. The solutions are based on basic mean shift and hough transform, both well known algorithms for this task.

In video object tracking, the goal is to locate one or multiple objects of interest, known as targets, in each frame of a video. These targets are usually located by drawing the smallest possible rectangle, object known as bounding box around the found target, note that the bounding box contains the found target. Video object tracking applications are a wide open subject, for example, it can be used in autonomous driving, smart video surveillance, human-computer interaction, sport analytics and much more.

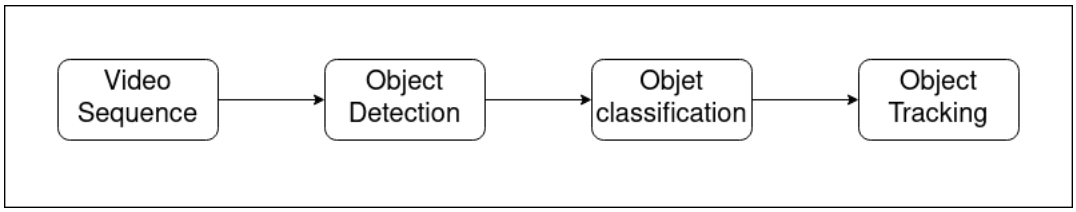


Figure 1: common object tracking flow diagram. [1]

Object detection and tracking can be subdivided as shown in figures 2 and 3

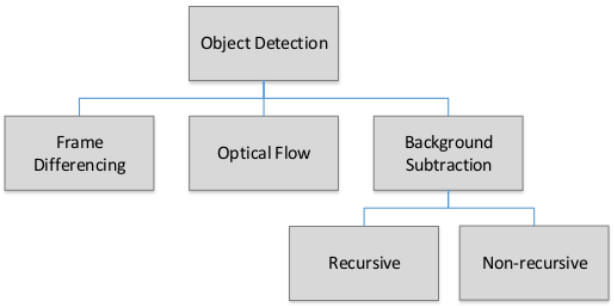


Figure 2: object detection methods [2]

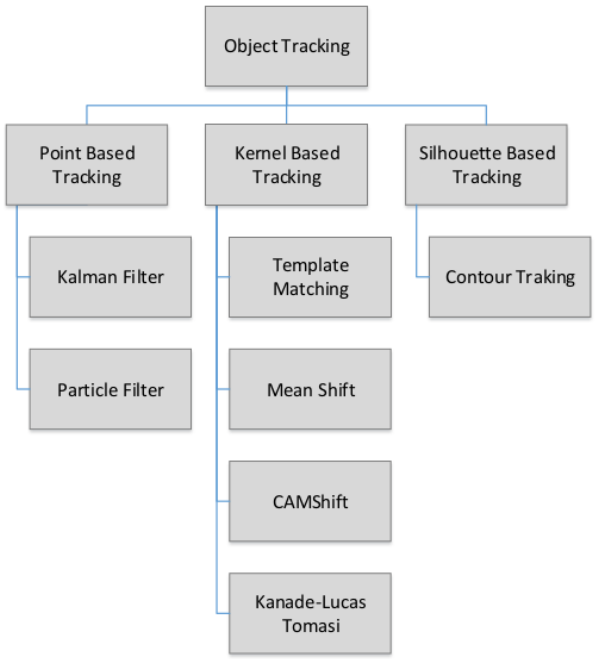


Figure 3: object tracking sub categories [2]

0.2 Mean Shift

Question 1
Experiment the tracking performed by the provided code Tracking_MeanShift.py that uses the basic Mean Shift algorithm, on the marginal density of the hue component H. Explain the principle of this algorithm, and illustrate its advantages and limits by your experiments.

solution:
The meanshift tracking method iteratively finds the area of a video frame that is most similar to the previously initialized model. So, let's define our problem as from a video you select an area from it and design it as your target and the goal here is to find the target in the sequence of frames of this given video, basically the target will be tracked along the video. On this selected region, we select a bunch of points based on the color histogram and compute the centroid of spatial points. If the centroid lies at the center of this region, we know that the object hasn't moved. But if the centroid is not at the center of this region, then we know that the object is moving in some direction. The movement of the centroid controls the direction in which the object is moving. So, we move the bounding box of the object to a new location so that the new centroid becomes the center of this bounding box. Hence, this algorithm is called Meanshift because the mean, i.e the centroid, is shifting. This way, the location of the object is kept evolutively frame to frame.

Algorithm 1: Mean shift tracking pseudocode

Data: $\{\mathrm{y}, h_R(u)\}$ (previous frame)
Result: Final position x and a new reference histogram (model) $h_R(u) = h_{\mathbf{x}}(u)$
Calculate the histogram $h_{\mathrm{y}}(u)$ with the current frame (Step 1)
Calculate the weights ω_i for each point of the support
Mean-shift: calculate the new position x:

$$\mathbf{x} = \frac{\sum_S \omega_i g\left(\|\mathbf{y} - \mathbf{x}_i\|\right) \mathbf{x}_i}{\sum_S \omega_i g\left(\|\mathbf{y} - \mathbf{x}_i\|\right)}$$

where $g(x) = -k'(x)$
if $\|\mathrm{x} - \mathrm{y}\| < \varepsilon$ **then**
  stop
else
  y $\leftarrow$ x, and go to **Step 1**
end

The following is the step-by-step implementation for the meanshift method for object tracking in a video using the well used script language Python, the first part of the code is used to determine the region of interest. Note that *ROI* is the object that we want to track, ROI stands for region of interest.

```
roi_defined = False
def define_ROI(event, x, y, flags, param):
    global r, c, w, h, roi_defined
    # if the left mouse button was clicked,
    # record the starting ROI coordinates
    if event == cv2.EVENT_LBUTTONDOWN:
        r, c = x, y
        roi_defined = False
    # if the left mouse button was released,
    # record the ROI coordinates and dimensions
    elif event == cv2.EVENT_LBUTTONUP:
        r2, c2 = x, y
        h = abs(r2 - r)
        w = abs(c2 - c)
        r = min(r, r2)
        c = min(c, c2)
        roi_defined = True
```

The next step is to define the termination criteria that will be used. Here, the main idea is to make this process a finite process, thus the criteria either can be a fixed number of iterations or it can be set that when the centroid is not moving anymore through the iterative process, then we stop.

```
# Setup the termination criteria: either 10 iterations,
# or move by less than 1 pixel
term_crit = ( cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1 )
```

Following the code, the steps are : iterate through each frame, calculate the histogram back projection, apply the meanshift method to get the new location and draw it on the window. These steps are iterated until the condition terminates.

```
while(1):
    ret ,frame = cap.read()
    if ret == True:
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        # Backproject the model histogram roi_hist onto the
        # current image hsv, i.e. dst(x,y) = roi_hist(hsv(0,x,y))
        dst = cv2.calcBackProject([hsv],[0],roi_hist,[0,180],1)

        # apply meanshift to dst to get the new location
        ret, track_window = cv2.meanShift(dst, track_window, term_crit)

        # Draw a blue rectangle on the current image
        r,c,h,w = track_window
        frame_tracked = cv2.rectangle(frame, (r,c), (r+h,c+w), (255,0,0) ,2)
        cv2.imshow('Sequence',frame_tracked)

        k = cv2.waitKey(60) & 0xff
        if k == 27:
            break
        elif k == ord('s'):
            cv2.imwrite('Frame_%04d.png'%cpt,frame_tracked)
        cpt += 1
    else:
        break
```

Experiments

Test Sequence 1: Women

In the first test sequence a women is tracked. After selecting the object of interest (ROI), the algorithm is started. First of all, be careful to select only the object you want to follow. This is because if you select a bigger area than the object you want to follow, since it's moving, its background is probably changing, so the object it's being tracked is composed of a real object and some noise that is the background, this is a much harder task than to track just the object. The figure set 4 shows the object tracking in the case of a women being tracked. In this case, we can notice that the algorithm can't track well the car. By selecting the pink pullover of the woman, we can easily follow the woman's position in the video. This is because the pink color is not present in the background. In frame (4k) we can one can clearly notice that mean shift algormithm is sensitive to occlusions.

Test Sequence 2: Mug

The figure set 5 shows the object tracking in the case of the mug. At the beginning of the video the object is tracked successfully but from the frame the algorithm can't follow anymore. the most apparent cause is motion blur and sudden movements.

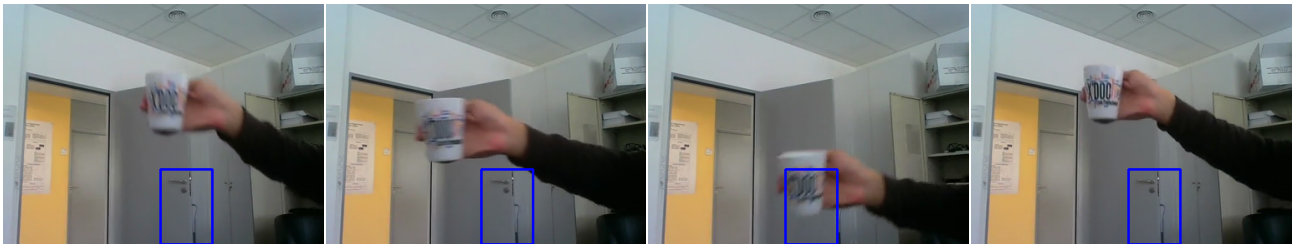
(a) Frame 4 (b) Frame 57 (c) Frame 90 (d) Frame 135

(e) Frame 184 (f) Frame 236 (g) Frma 350 (h) Frame 360

(i) Frame 366 (j) Frame 376 (k) Frame 425 (l) Frame 561

Figure 4: This set of figures represents the result of the tracking of a women by the mean shift algorithm.

(a) Frame 3 (b) Frame 17 (c) Frame 36 (d) Frame 39



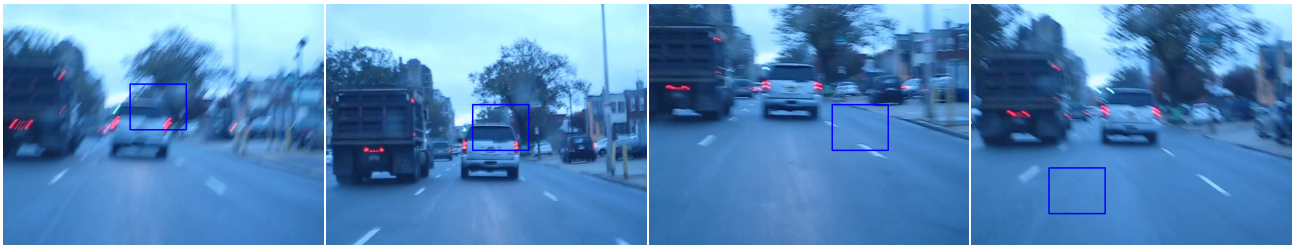
(e) Frame 73 (f) Frame 91 (g) Frma 110 (h) Frame 157

Figure 5: This set of figures represents the result of the tracking of a mug by the mean shift algorithm.

Test Sequence 3: Car

The figure set 6 shows the object tracking in the case of a car being tracked. In this case, we can notice that the algorithm can't track well the car. We transform the image from RGB to HSV to make the work easier. We obtain the probability that the model matches with different parts of a particular video frame. On this case, since the color of the car and the color of the road are similar, we get wrong results. Motion

blur and sudden movements greatly affect tracking.

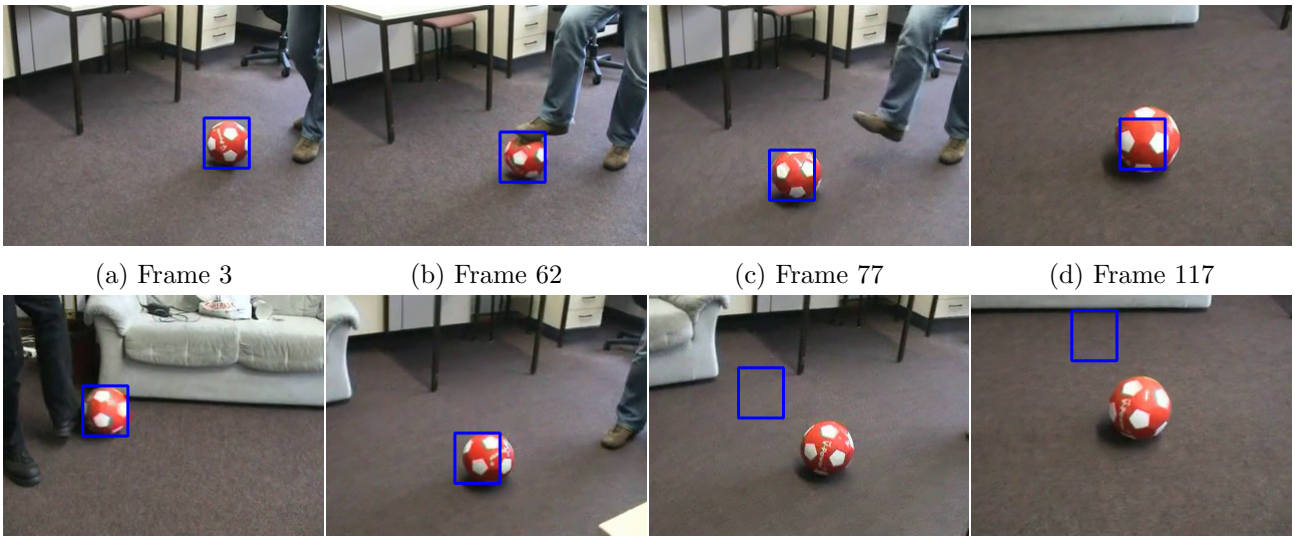


(a) Frame 3 (b) Frame 24 (c) Frame 36 (d) Frame 102

Figure 6: This set of figures represents the result of the tracking of a car by the mean shift algorithm.

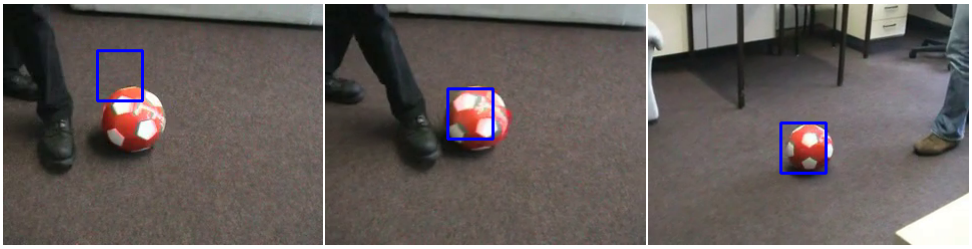
Test Sequence 4: Soccer ball

The experience carried out here is the tracking of the soccer ball. This experience is very interesting because it highlights the sensitivity of the mean shift algorithm to object scaling. For example the code doesn't succeed in following the ball in (7g) (7h).



(a) Frame 3 (b) Frame 62 (c) Frame 77 (d) Frame 117

(e) Frame 181 (f) Frame 261 (g) Frame 266 (h) Frame 299



(i) Frame 330 (j) Frame 335 (k) Frame 376

Figure 7: This set of figures represents the result of the tracking of a ball by the mean shift algorithm.

Test Sequence 5: Basketball player

The mean shift algorithm perfectly follows the player of the selected green team. In this example we have taken care to select only the green team shirt of the player. Even if the background is complex and variable, the results are very satisfying.

(a) Frame 4 (b) Frame 23 (c) Frame 58 (d) Frame 100

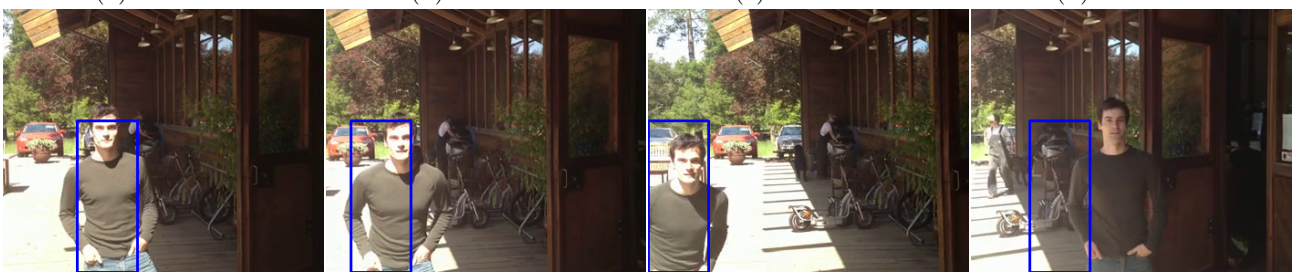
(e) Frame 150 (f) Frame 413 (g) Frame 487 (h) Frame 593

Figure 8: This set of figures represents the result of the tracking of the basketball player by the mean shift algorithm.

Test Sequence 6: Sun

In this experiment, the effect of illumination changes is examined.

(a) Frame 3 (b) Frame 12 (c) Frame 22 (d) Frame 38



(e) Frame 66 (f) Frame 104 (g) Frma 118 (h) Frame 167

Figure 9: This set of figures represents the result of the tracking of a person by the mean shift algorithm.

To summarize, the best results can be achieved if the following conditions are fulfilled:

- The target object is mainly composed of one color.
- The target object does not change its color.
- Illumination does not change drastically.
- There are no other objects in the scene similar to the target object.

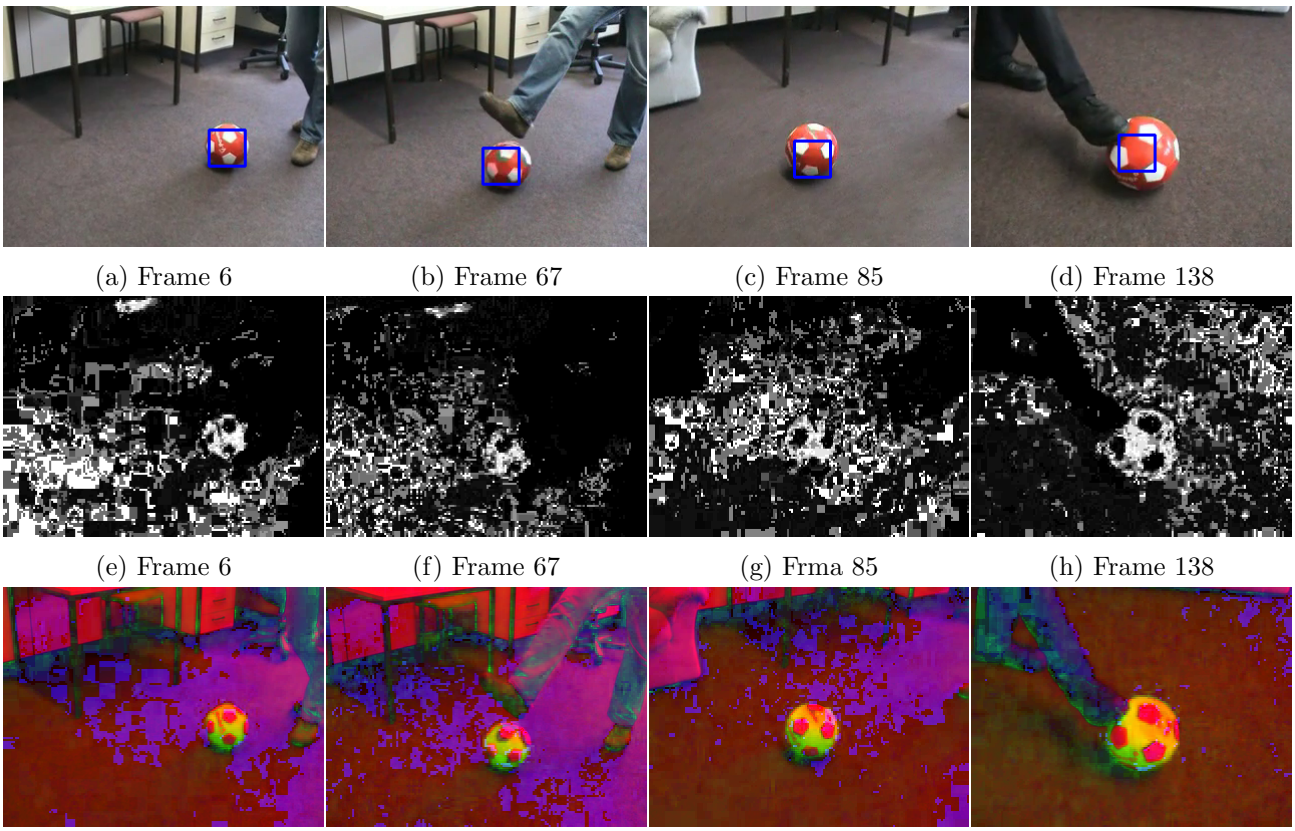
The following table lists the strengths and weaknesses of the mean shift algorithm:

Strengths	Weaknesses
Suitable for real data analysis	Inappropriate window size can cause modes to be merged, or generate additional "shallow" modes
Does not assume any prior shape (e.g. elliptical) on data clusters	The window size is not trivial
Can handle arbitrary feature spaces	Sensitive to the transformation of objects.
Only one parameter to choose	Sensitive to object scaling
	Sensitive to the rotation of objects
	sensitive to brightness and contrast

Question 2
Analyse more in-depth the result by displaying the sequences of hue images, and also the weight images corresponding to the back-projection of the hue histogram. Propose and program improvements, by changing the computed density and/or updating the model histogram.

solution:

As we can see on the figures of experiment of the ball . the first four (10a 10b 10c 10d) figures represent the tracking of the ball and the next four (10e 10f 10g 10h) represent the density function used and finally the last four (10i 10j 10k 10l) present the Hue saturation value of the frame.



(a) Frame 6 (b) Frame 67 (c) Frame 85 (d) Frame 138

(e) Frame 6 (f) Frame 67 (g) Frma 85 (h) Frame 138

(i) Frame 6 (j) Frame 67 (k) Frame 85 (l) Frame 138

Figure 10: This set of figures represents the result of the tracking of the ball by the mean shift algorithm.

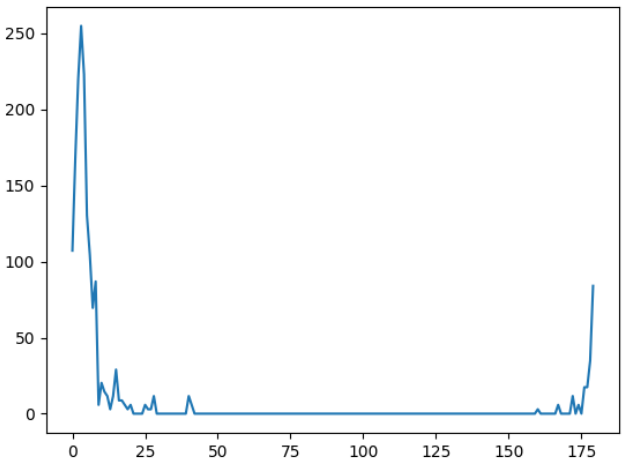


Figure 11: ROI histogram (Ball)

Let remind:

$$\text{Original Mean-Shift:} \quad \text{Find mode of } c\sum_{i=1}^{n} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right) \text{ using } y_1 = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{y_0-x_i}{h}\right\|\right)^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}$$

$$\text{Extended Mean-Shift:} \quad \text{Find mode of } c\sum_{i=1}^{n} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right) \text{ using } y_1 = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{y_0-x_i}{h}\right\|\right)^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{y_0-x_i}{h}\right\|^2\right)}$$

The classic mean shift algorithm has several weaknesses. One way to improve the algorithm is to change density function. By analyzing the Fourier spectrum of natural images we will build a saliency map, which allows us to label certain statistically interesting patches of the image as potential objects. We will then feed the location of all the proto-objects to a mean-shift tracker that will allow us to keep track of where the object move from one frame to the next.

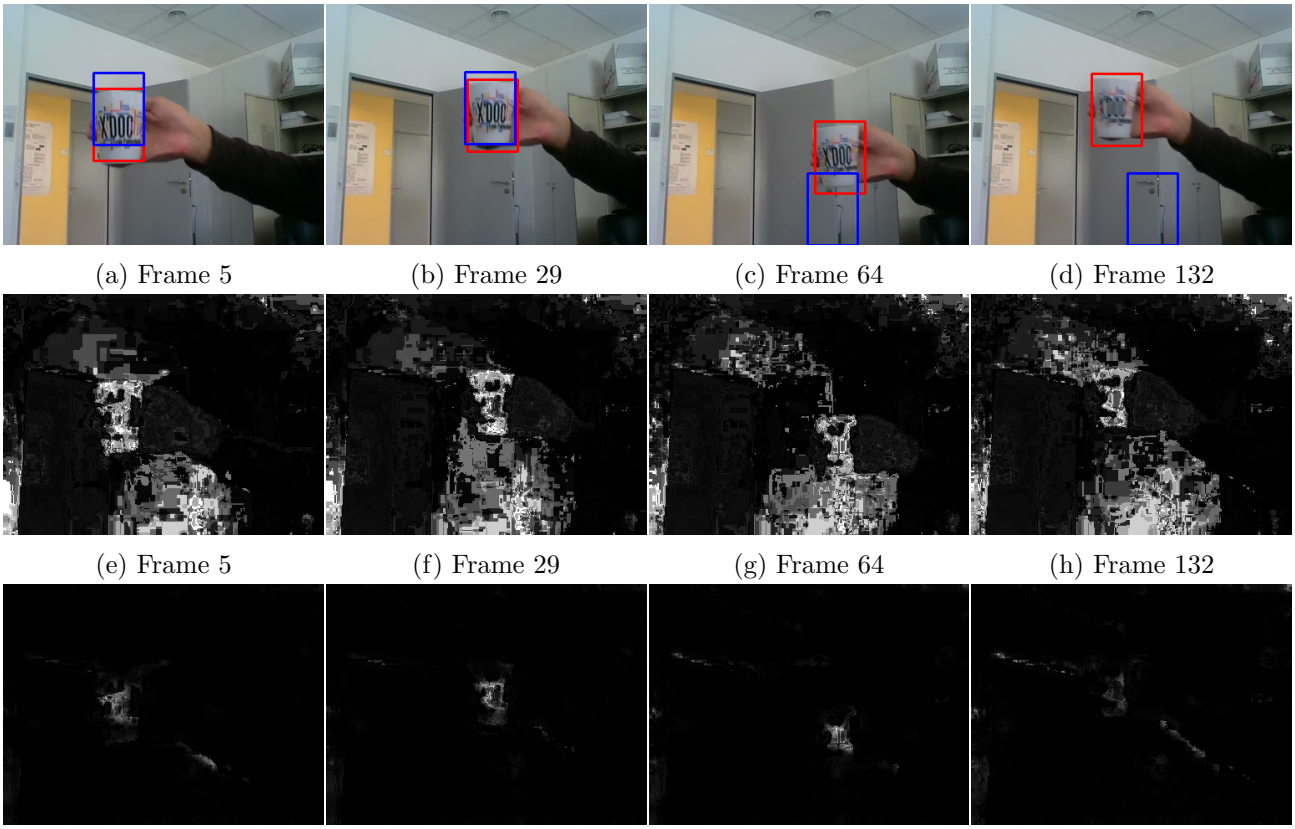
Saliency map: We will use Fourier analysis to get a general understanding of natural image statistics, which will help us build a model of what general image backgrounds look like. By comparing and contrasting the background model to a specific image frame, we can locate sub-regions of the image that pop out of their surroundings.

Once all the potentially interesting patches of an image are located we will combined with the distribution used before.

```
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
# Backproject the model histogram roi_hist onto the
# current image hsv, i.e. dst(x,y) = roi_hist(hsv(0,x,y))
dst = cv2.calcBackProject([hsv],[0],roi_hist,[0,180],1)
cv2.imshow('dst', dst)
cv2.imshow('hsv', hsv)
sal = Saliency(frame, use_numpy_fft=False, gauss_kernel=(3, 3))
cv2.imshow('saliency', sal.get_saliency_map())
cv2.imshow('new_dis', sal.get_saliency_map()*dst)
# apply meanshift to dst to get the new location
ret, track_window = cv2.meanShift(sal.get_saliency_map()*dst, track_window, term_crit)
```

The Saliency will generate a map of all the interesting proto-objects and feed that into the tracker module.The red rectangle it is in is the improved mean shift and the bleu rectangle is the original mean shift. On the set of figure 12 13 14, we can see that the change of the density function and the use of saliency map has greatly improved the tracking of objects. In the figures 12i 12j 12k 12l we can see the

new density.
the algorithm has become more robust against sudden movements this has been proved by the results of the experiment on the car 13 and also robust against the background noise. On the other hand the algoritm still sensitive to brightness and contrast.

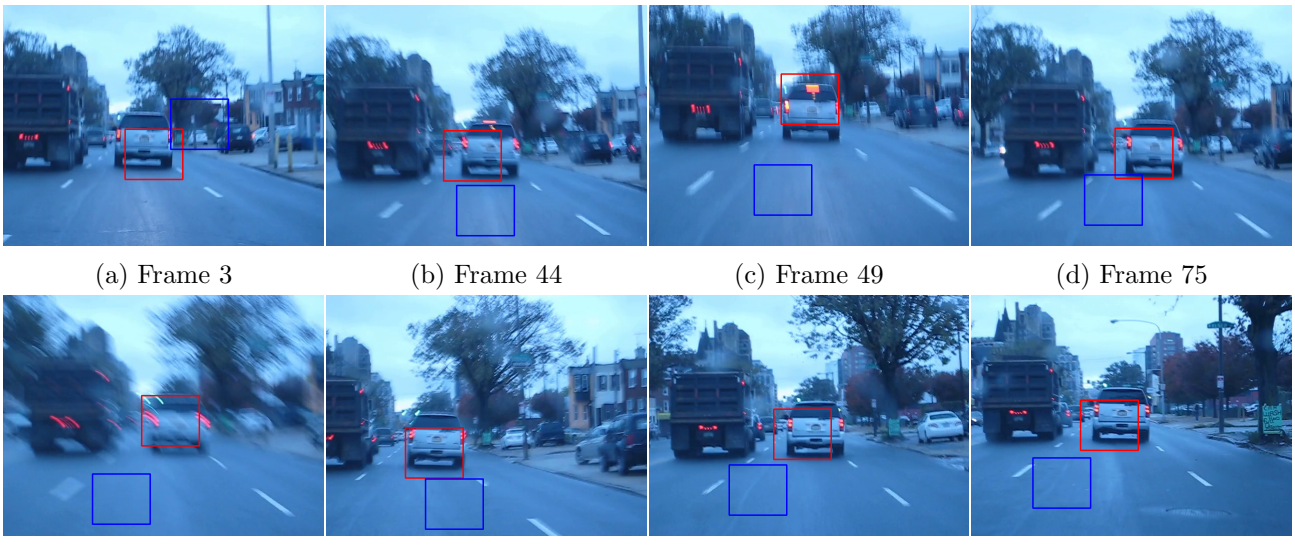


(a) Frame 5 (b) Frame 29 (c) Frame 64 (d) Frame 132

(e) Frame 5 (f) Frame 29 (g) Frame 64 (h) Frame 132

(i) Frame 5 (j) Frame 29 (k) Frame 64 (l) Frame 132

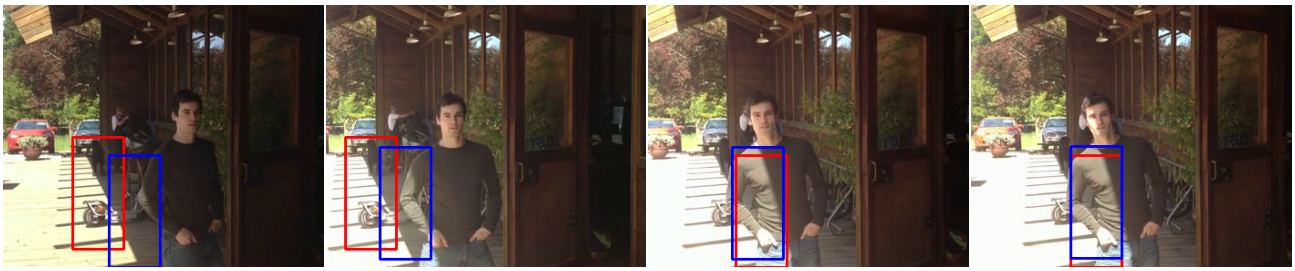
Figure 12: This set of figuress represents the result of the tracking of a mug by original the mean shift (blue) and the improved mean shift algorithm (red).



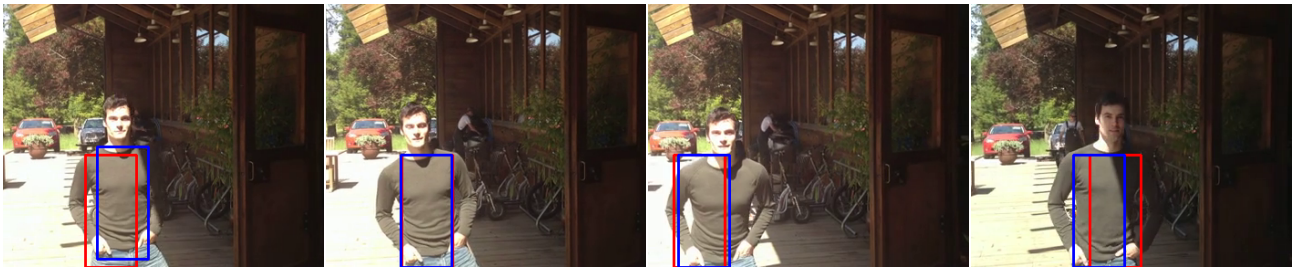
(a) Frame 3 (b) Frame 44 (c) Frame 49 (d) Frame 75

(e) Frame 106 (f) Frame 111 (g) Frma 162 (h) Frame 222

Figure 13: This set of figures represents the result of the tracking of a car by by original the mean shift (blue) and the improved mean shift algorithm (red).



(a) Frame 18 (b) Frame 35 (c) Frame 41 (d) Frame 44



(e) Frame 64 (f) Frame 70 (g) Frma 104 (h) Frame 155

Figure 14: This set of figures represents the result of the tracking of a person by original the mean shift (blue) and the improved mean shift algorithm (red).

0.3 Hough Transform

- The Hough transform was initially developed to detect analytically defined shapes (e.g., lines, circles, ellipses etc.).
- The generalized Hough transform can be used to detect arbitrary shapes (i.e., shapes having no simple analytical form).
- It requires the complete specification of the exact shape of the target object.

Algorithm 2: Generalized Hough Transform pseudocode

Data: edges (x_i, y_i, ϕ_i)
Result: Local Maxima in $A(x_c, y_c)$
Find Object Center (x_c, y_c)
Create Accumulator Array $A(x_c, y_c)$
Mean-shift: calculate the new position x: Initialize: $A(x_c, y_c) = 0 \quad \forall (x_c, y_c)$
For each edge point (x_i, y_i, ϕ_i) For each entry $\bar{r}_k^i$ in table, compute:

$$x_c = x_i + r_k^i \cos \alpha_k^i$$
$$y_c = y_i + r_k^i \sin \alpha_k^i$$

Increment Accumulator: $A(x_c, y_c) = A(x_c, y_c) + 1$
Find Local Maxima in $A(x_c, y_c)$

Question 3
Calculate for each frame, the local orientation, i.e. the gradient argument of pixels, and also the gradient magnitude. Use a threshold on the gradient magnitude to mask pixels whose orientation is not significant. Display the sequence of orientations, where the masked pixels appear in red.

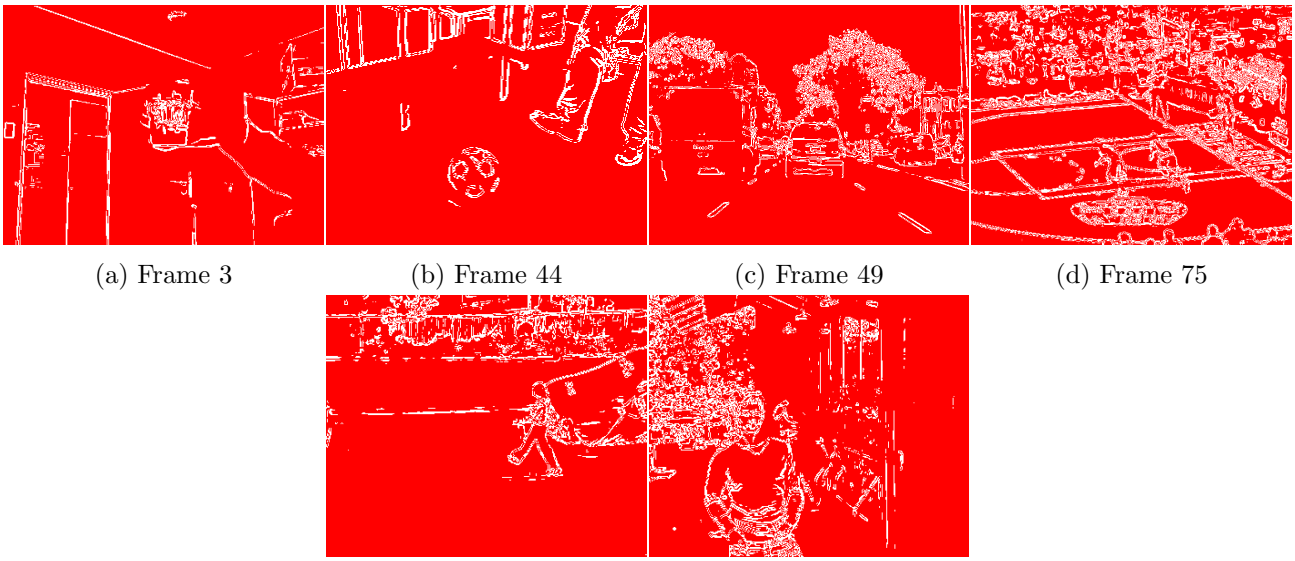
solution:

The orientation of the gradient is calculated and a mask is created based on the value of the gradient modulus.

```
def gradient_magnitude(frame_gray):
        dx = cv2.Sobel(frame_gray,cv2.CV_64F,1,0,ksize=3)
        dy = cv2.Sobel(frame_gray,cv2.CV_64F,0,1,ksize=3)
        # Compute the magnitude of the gradient
        return  np.hypot(dx,dy).astype('uint8')


def gradient_orientation(frame_gray):
        dx = cv2.Sobel(frame_gray,cv2.CV_64F,1,0,ksize=3)
        dy = cv2.Sobel(frame_gray,cv2.CV_64F,0,1,ksize=3)
        # Compute the orientation
        return  (np.arctan2(dy,dx) * 180 / np.pi)
```

To display the sequence of orientations, where the masked pixels appear in red we convert the frame with grey scale to RGB like below:

```
gradient_magnitude = get_gradient_magnitude(frame_g)
_ , filtered = cv2.threshold(gradient_magnitude, threshold, 255, cv2.THRESH_BINARY)
cv2.imshow('filtered_gradient_magnitude', filtered)

backtorgb = cv2.cvtColor(filtered,cv2.COLOR_GRAY2RGB)
backtorgb[np.where((backtorgb==[0,0,0]).all(axis=2))] = [0,0,255]
cv2.imshow('backtorgb', backtorgb)
```



(a) Frame 3 (b) Frame 44 (c) Frame 49 (d) Frame 75

(e) Frame 106 (f) Frame 111

Figure 15: The sequence of orientations, where the masked pixels appear in red.

Question 4
Build a model of the initial object under the form of an implicit model indexed on the orientation (R-Table), calculated on significant (unmasked) pixels. Then, calculate the associated Hough transform on all the images of the sequence. Calculate the straightforward tracking, corresponding to the maximal value of the Hough transform at each image. Comment and criticise the obtained results.

solution:

Two functions have been created to track objects using the generalized Hough transformation. The R−table, as the name suggest, it is a table in which each row corresponds to an edge orientation. Each entry in the table represent an edge pixel location (r,α) with respect to a center point. A single row in the table can have multiple such entries. When inserting a new entry (r,α) representing an edge pixel, we first

find the orientation of that edge pixel. Then the orientation is used to find the row in the table and (r, α) is appended to the entries in that particular row. Since the r−table's index is discrete, you can choose how many types of edge orientations to store since the values of the orientation map is between 0 and 2π, so we can break that into (row_count) values shown in the python code below.

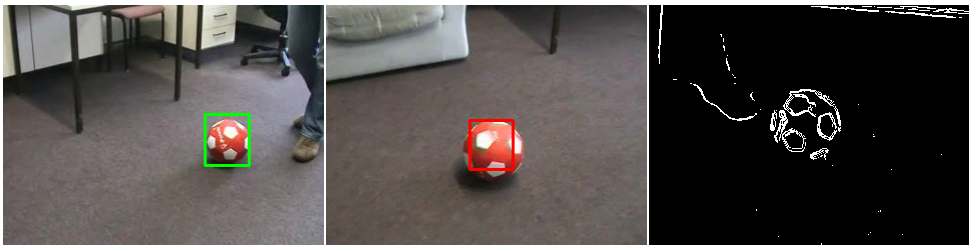
$$r = \sqrt{(x_c - x)^2 + (y_c - y)^2}$$
$$\alpha = \tan^{-1}\left(\frac{y_c - y}{x_c - x}\right)$$

```python
def build_r_table(obj):
    X,Y =  obj.shape
    gradient_magnitude = get_gradient_magnitude(obj)
    _ , filtered = cv2.threshold(gradient_magnitude, threshold, 255, cv2.THRESH_BINARY)
    cv2.imshow('r_table', filtered)
    orientation = get_gradient_orientation(filtered)
    orientation[filtered == 0] = -255
    unique_orientation = np.unique(orientation)

    r_table = dict()
    center = np.array([[int(X/2) ,int(Y/2)]])

    for teta in unique_orientation:
        if teta == -255:
            continue
        r_table[teta] = center - np.argwhere(orientation == teta)
    return r_table


def transform_hough(image, r_table, x,y,w,h):

    X, Y = image.shape
    gradient_magnitude = get_gradient_magnitude(image)
    _ , filtered = cv2.threshold(gradient_magnitude, threshold, 255, cv2.THRESH_BINARY)
    orientation = get_gradient_orientation(filtered)
    orientation[filtered == 0] = -255

    vote = np.zeros(image.shape)

    for teta in r_table:
        tmp = np.argwhere(orientation == teta)
        if tmp.shape[0] == 0 :
            continue
        for r in r_table[teta]:
            ind_for_vote = tmp + r
            ind_for_vote = ind_for_vote[ (ind_for_vote[:,0] < X) & (ind_for_vote[:,0] > 0)
            ↪  &(ind_for_vote[:,1] < Y) & (ind_for_vote[:,1] > 0)  ]
            vote[ind_for_vote[:,0], ind_for_vote[:,1]] += 1
    vote[max(x-w, 0):min(x+2*w, X), max(y - h, 0):min(y+2*h, Y)] += 200
    centers = np.argwhere(vote == np.amax(vote))
    center = centers.mean(axis = 0).astype('int')
    return center[0], center[1]
```

The generalized Hough Transform can be used for object recognition. An advantage is that this algorithm can be used on any arbitrary shape since we only have to parametrize the contour of the shape. Moreover, this algorithm can be extended to compute multiple scales of the template shape by simply storing multiple scales of r of the (r,α) in the r-table.

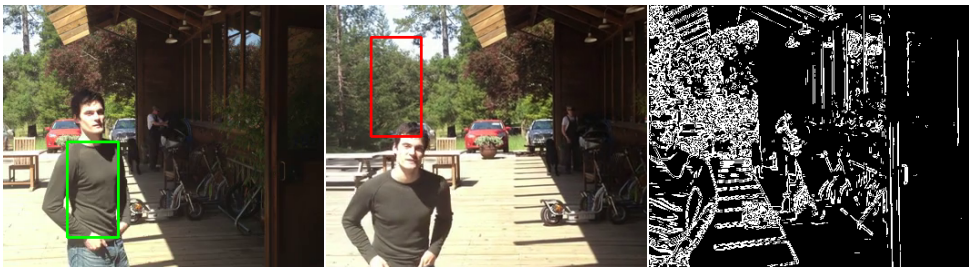
(a) ROI (b) tracking (c) Distribution

(d) R-table Mug



(e) ROI (f) Tracking (g) distribution

(h) R-table Ball



(i) ROI (j) Tracking (k) Distribution

(l) R-table person

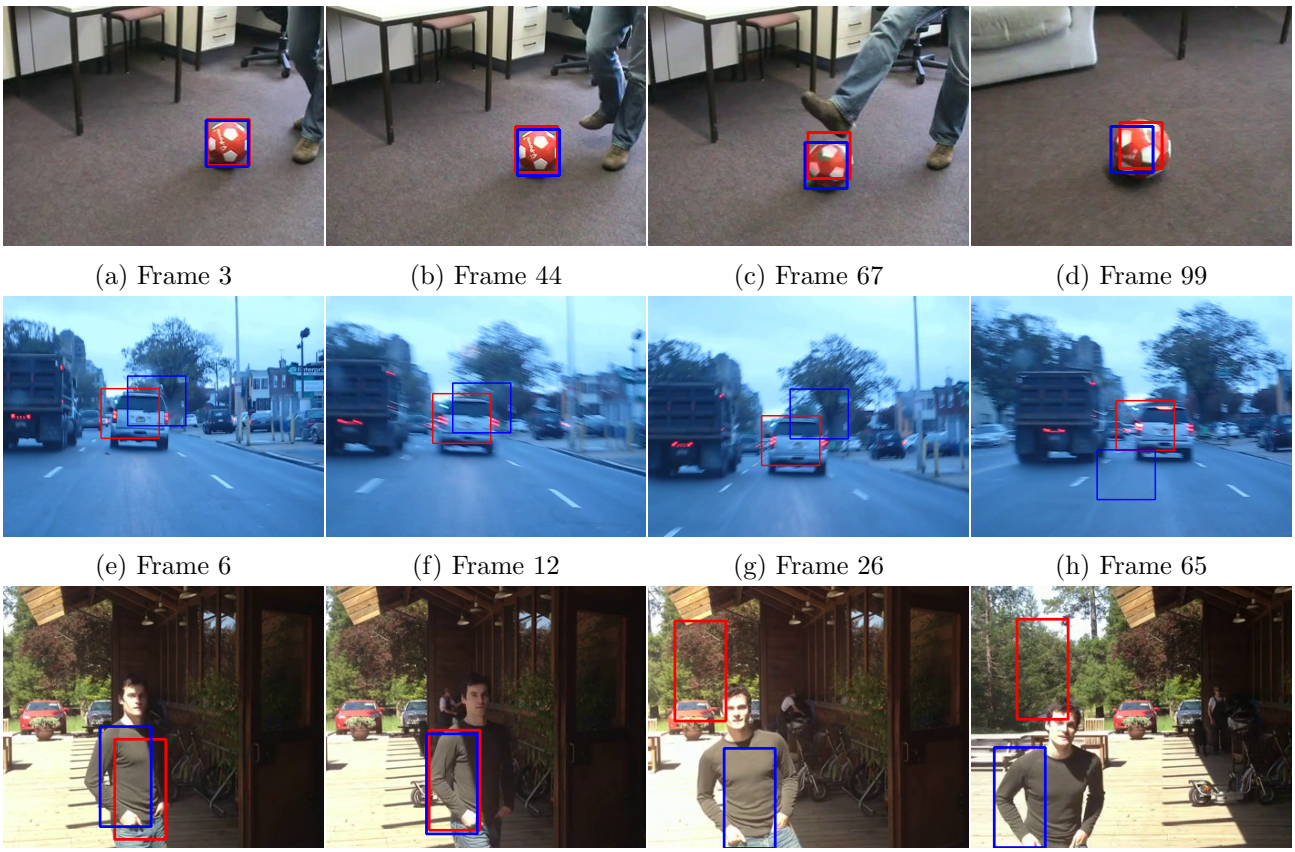
Figure 16: This set of figures represents the result of the tracking using the generalized Hough transform.

Strengths	Weaknesses
The generalized Hough transform is essentially a method for object recognition.	It requires a lot of storage and Extensive computation
It is robust to partial or slightly deformed shape	
It is robust to the presence of additional structures in the image	
It is tolerant to noise	

Question 5
Replace the computation of the maximal value by the application of the Mean Shift on the Hough transform (i.e. by replacing the back-projection of the Hue histogram by the output of the Hough transform in the first argument of the Mean Shift). Interpret the result and compare it with the previous one. Propose an update strategy of the model that would allow to be robust to aspect changes of the object.

solution:

By replacing the computation of the maximal value by the application of the Mean Shift on the Hough transform we get a more efficient tracking than the original approach.



(a) Frame 3 (b) Frame 44 (c) Frame 67 (d) Frame 99

(e) Frame 6 (f) Frame 12 (g) Frame 26 (h) Frame 65

(i) Frame 8 (j) Frame 12 (k) Frame 72 (l) Frame 129

Figure 17: This set of figures represents the result of the tracking of objects by replacing the back-projection of the Hue histogram by the output of the Hough transform in the first argument of the Mean Shift.

Bibliography

[1] Norah Almohaimeed and Master Prince. "A Comparative Study of different Oject Tracking Methods in a Video". In: *International Journal of Computer Applications* 181 (Feb. 2019), pp. 1–8. DOI: 10.5120/ijca2019918470.

[2] Himani S. Parekh, D. Thakore, and U. K. Jaliya. "A Survey on Object Detection and Tracking Methods". In: *International Journal of Innovative Research in Computer and Communication Engineering* 2 (2014), pp. 2970–2978.