Lab 2: Object Tracking in Videos

Thomas Wimmer

October 2022

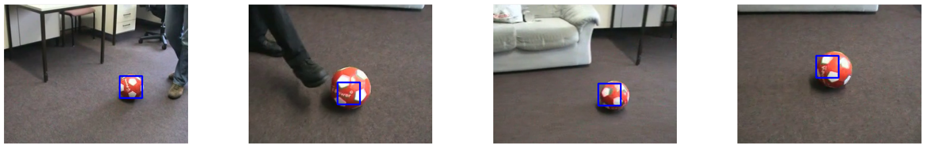
1 Mean Shift

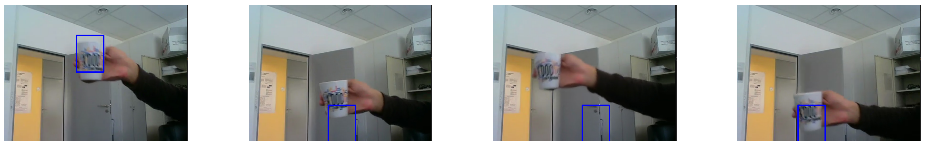
1.1 Q1: Basic Experiments

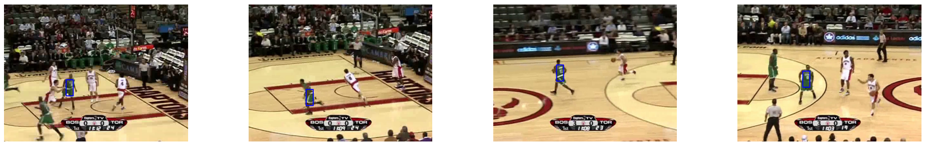
The Mean-Shift algorithm first calculates a histogram from the region of interest (ROI) defined by the user at the beginning of the video. In the given code framework, this histogram is calculated over the marginal distribution only over the Hue values of the HSV image. In general, however, it can also be calculated for other values or across multiple dimensions. This option will be discussed in the next section. For the individual frames in the video, a backprojection is then performed on the specified histogram. On the resulting weight map, the classical mean shift algorithm can be performed: Starting at the position of the last tracking window, the window is iteratively shifted in the direction of the weight centroid (the calculation is weighted with an isotrope kernel function, usually Gaussian-based) until either a predefined number of iterations is reached or the algorithm has converged.

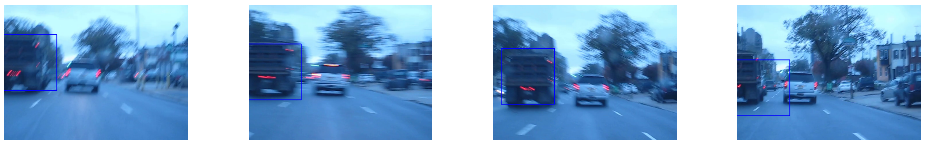
Clearly, one of the advantages of mean-shift tracking is that an ROI only needs to be defined at the beginning of the video and the algorithm does not need to do any offline or online learning to track the object. This also makes the algorithm more flexible than, for example, a neural network that has only been trained to track humans. It is thus application independent. Since the algorithm is based on the histogram (which can be calculated freely according to use case over only Hue, Saturation or only over a specific color channel in RGB) it is also more flexible in terms of the appearance of the tracked objects. The algorithm is relatively robust to rotations or small deformations of the tracked object because the tracking is only based on the histogram and thus not dependent on the spatial orientation of an object as long as it stays within the tracking window size.

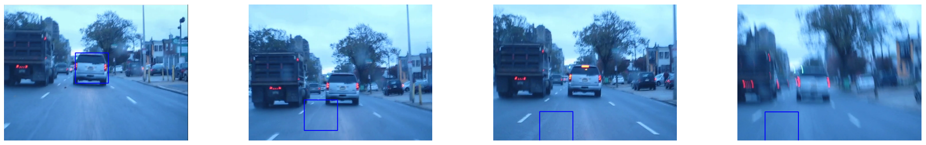
On the other hand, mean-shift tracking has the disadvantage that the initial window size is not updated (although the size of the tracked object may increase/decrease over time in the video) and that the algorithm usually has problems finding a "lost" object again. Also, when the lighting and thus the histogram of the tracked object changes, the algorithm in its basic form is not adaptive to these changes. Occlusions, sudden movements, and complex or changing backgrounds can pose significant challenges to the algorithm.


(a) Tracking the ball in front of the grey background works very well as the tracking is based on the hue component. We can however observe the problem of non-changing tracking window sizes in the second image.


(b) The cup is lost in the second image but the algorithm managed to partially recover in the last image.


(c) Tracking the player in the green shirt can be achieved by only selecting the green shirt and not too many surroundings in the initial ROI. Using this trick to set a clearly distinguishable histogram from the surrounding area, the tracking works well.


(d) Tracking the transporter on the left side works well, even in case of occlusions as in the last image.


(e) However, tracking the car in the center is difficult due to the general blue shift in the image and the many camera movements and motion blurs. The car is ”lost” and the algorithm cannot find it again.

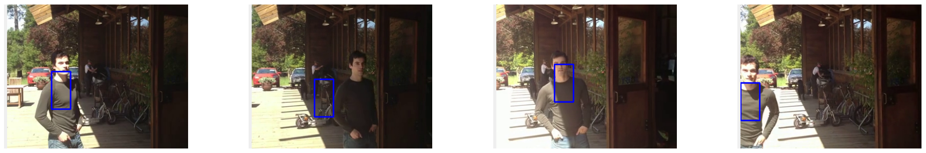

(f) Tracking the head and chest of the person doesn’t work when the brightness changes too much. The algorithm is able to find the person again as he moves back out of the shadow.

Figure 1: Analysis of tracking results of the Mean-Shift Tracking algorithm.

1.2 Q2: In-depth Analysis and Possible Improvements

The analysis of the different components of the HSV image can be performed by additionally outputting the Hue and Saturation components during tracking. Furthermore, the weight map (backprojection of the ROI histogram) can be output to get even more information about the internal processes in the algorithm. Sample Outputs of what this looks like are given in Figure 2. Based

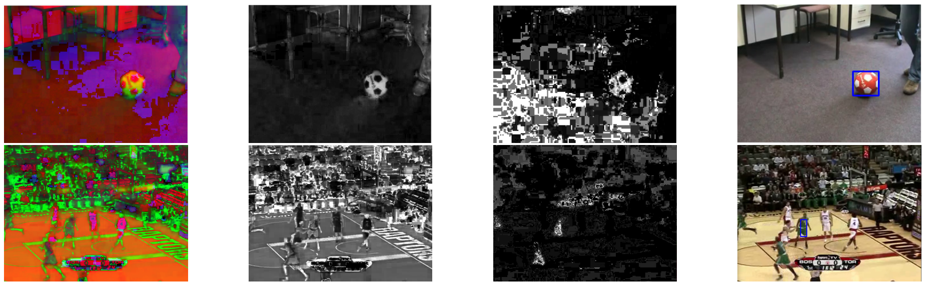


Figure 2: The visualization of the Hue (1) and Saturation (2) component, as well as the weight map for the mean-shift algorithm (3) can help understanding the results of the tracking algorithm (4) with (1-4) from left to right.

on the finding that using the hue component alone does not make sense in most cases, since it provides more or less arbitrary values for light or dark areas in the image, and the information about saturation is not considered at all in the tracking, it only makes sense to try to consider also the saturation component in a multivariate distribution instead of the marginal distribution over only the hue component.

Using this method to create the histogram gives much better results in some cases as is shown in Figure 3. A parameter that is also important to consider here and that can be tuned not only for saturation but also for the hue channel is the number of bins for the histogram.

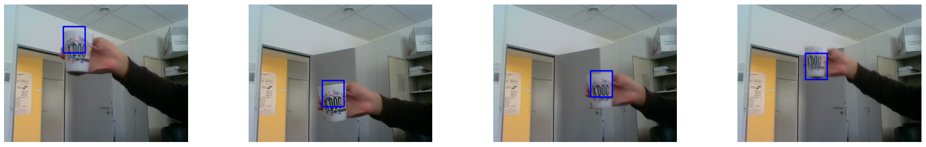


Figure 3: The cup that could not be tracked in the initial approach can now be tracked of the mean-shift algorithm after considering also the saturation values.

Besides these improvements, iteratively updating the histogram of the ROI might also be interesting to deal with illumination changes / color shifts in the image. This can be done by an operation like this one:

```
roi_hist = roi_hist * (1 - alpha) + roi_hist_new * alpha
```

The parameter `alpha` becomes an additional hyperparameter in this case. As can be seen in Figure 4, this can actually improve the results in some cases. However, it also makes the algorithm more unstable in finding a ”lost” object in the image, since the histogram is updated with the histograms of the false detections. So if the algorithm does not find back to the ROI within a few frames, it is often quite impossible to recover from it.

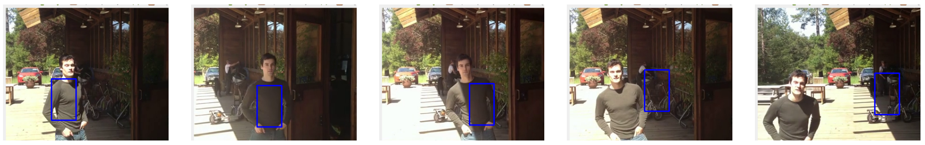


Figure 4: By updating the histogram we back-project during tracking, we can even track the person’s shirt in the shadows. However, when the person abruptly jumps out of the shadow in frame 4, the algorithm loses the ROI and stays in the shadow. It cannot recover from this mistake in the rest of the video.

2 Hough Transform

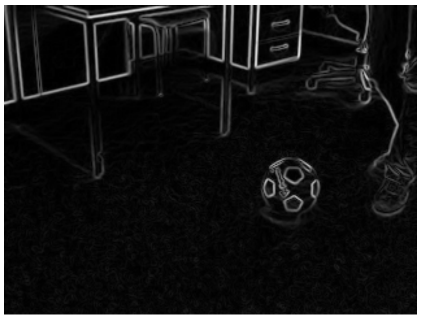
2.1 Q3: Basic Setup for Hough Transform

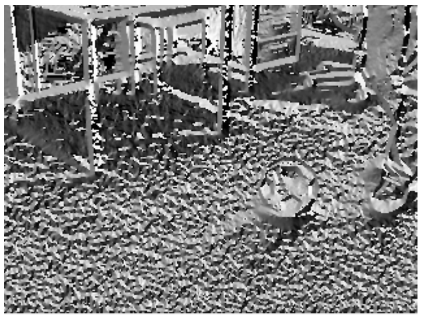
The Generalized Hough Transform is another Object Tracking method that is The first step in setting up the Generalized Hough Transform is to find a way to identify the pixels that are relevant to the algorithm. In this assignment, we select the points based on their gradient magnitude. If the magnitude of the gradient is greater than a certain threshold, the pixels are relevant. All relevant pixels in the ROI which is defined by the user at the beginning of the video are used to create the R-table. The R-table consists of the vectors of the relevant points to the center of the ROI and they are indexed by their gradient orientation.

It is thus necessary to compute the gradient magnitudes and orientations for all pixels in the image. OpenCV provides the user with a method to compute the gradient in x- and y-direction using the Sobel kernel to approximate the derivatives. From the so found gradients g_x, g_y we can then calculate the magnitude and orientation by

$$|g| = \sqrt{g_x^2 + g_y^2},$$
$$\theta(g) = \tan^{-1}\left(\frac{g_y}{g_x}\right).$$

We can visualize the gradient magnitude and orientation alongside the normal images, as well as the orientations of only pixels with significant magnitude as shown in Figure 5.


(a) Gradient magnitude


(b) Gradient orientations

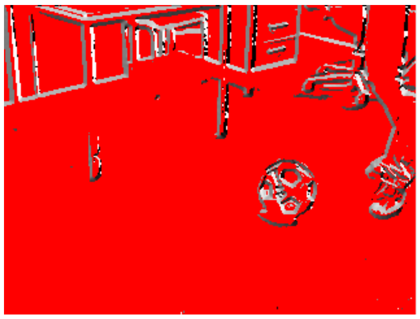

(c) Gradient orientations of the relevant pixels with masked pixels in red

Figure 5: The relevant pixels for the Generalized Hough Transform can be computed by selecting all points with significant gradient magnitude in the frame. The gradient orientation is used to index the R-table.

2.2 Q4: Calculating the Generalized Hough Transform

2.2.1 Implementation

To perform the object tracking using the Generalized Hough Transform, we have to build the R-table of our ROI using the gradient orientation as index and the vectors of relevant points to some reference point (can be chosen arbitrarily but needs to be the same for all points in the ROI, i.e. the center of the selected ROI frame) as values. We can use the computation of the gradients as described above. One important hyperparameter in this case is the number of bins that we use in our R-table. If we set this number to high (e.g. 360 rows in the R-table, one row corresponding to one gradient angle in degrees), the resulting voting map in the end will be too sparse. Making it too small on the other hand, can lead to a significant slowdown at runtime as there will be many vectors for each row in the R-table and can also lead to a decrease in accuracy of the tracking.

After creating the R-table from the ROI, we can perform the tracking. For each image, we identify all the relevant points using the same method as before (selecting points with significant gradient magnitude) and look up the list of vectors in the R-table that correspond to each gradient orientation. We can then compute the voting map by giving a vote to each point that can be reached from the significant points combined with one of the corresponding vectors in the R-table. Finally, the point in the image that received the most votes can be selected as the reference point (i.e., the central point of the tracking frame).

2.2.2 Comments

The first notable difference between the Meanshift tracking and the tracking via the Generalized Hough Transform is the difference in the run-time of the two algorithms as the latter takes more time and memory to be computed. This behavior could however possibly be improved by a more efficient implementation (possibly also at a lower level as the `C++` based OpenCV implementations).

When setting fitting gradient magnitude thresholds for the different scenes, the Generalized Hough Transform often outperforms the tracking using the Meanshift algorithm, as it improves on some of the issues mentioned in section 1.2. An example is shown in Figure 6.

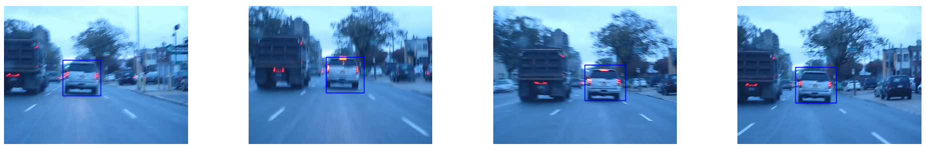


Figure 6: The tracking of the car works very well, even in situations with lots of camera movement in between frames or slight motion blur. The tracking with the Meanshift algorithm performed much worse in this example.

Two major problems can be identified with the generalized Hough transform as we currently use it for object tracking: Tracking is not continuous across different frames in the video. For each frame, we compute the votemap and simply choose the argmax as the most likely point where our ROI is currently located. This decision does not take into account the previous position at all. This makes it very likely that we will get inconsistent (i.e., "jumping") estimates of where the object is. This behavior can for example be observed when the object of interest is distorted through motion blur in one frame and sharp in the next frame again, as can be seen in Figure 7. However, this property can also have a positive side, meaning that when the object of interest leaves the image for a moment or is occluded, the algorithm is still likely to find it as soon as it is fully visible again in the image.

The second major drawback of the generalized Hough transform is that its performance depends very much on the method we use to decide whether a point is relevant to the algorithm or not and the method to index the R-table. For example, if the object to be tracked (e.g., the ROI) contains only a few points with high gradient magnitude, but other objects in the background have more relevant points (i.e., more points with high gradient magnitude and similar

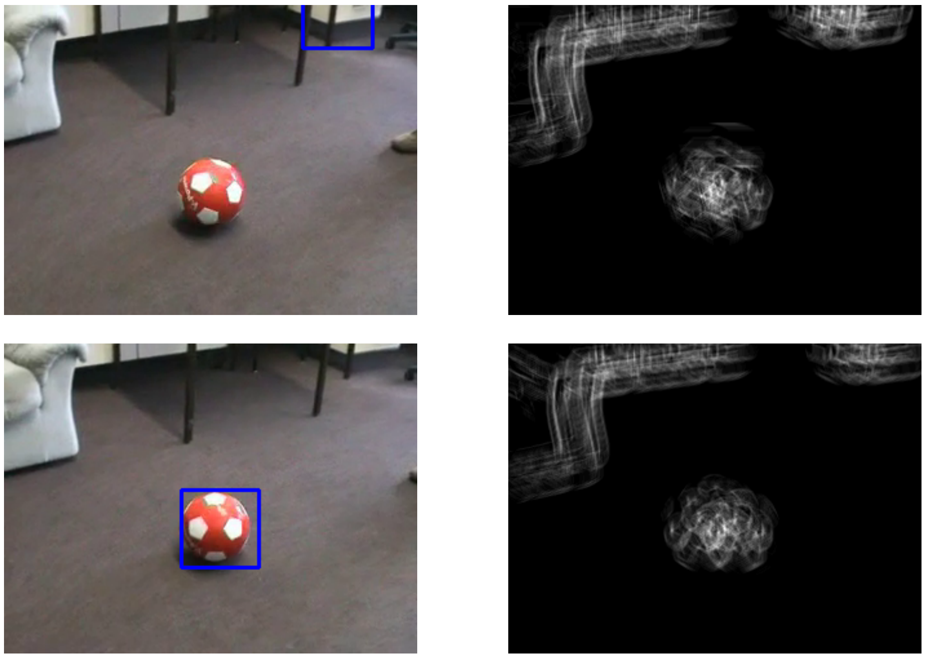


Figure 7: Since the basic Generalized Hough Transform algorithm simply takes the argmax of the votemap as the most likely position of the ROI, it is prone to locally inconsistent predictions from frame to frame.

gradient orientation), it is very likely that the algorithm will have difficulty finding the desired object in the video. Indexing the R-table by the gradient orientations makes the algorithm also sensitive to rotations (or deformations) of the object that should be tracked as the indexing is not rotation-invariant (only to the degree of binning used for the gradient orientations). This can lead to undesired behavior. An example of the effect of high gradient magnitudes in the background is displayed in Figure 8, while an example of the effect of ROI deformations is visualized in the example in Figure 9.

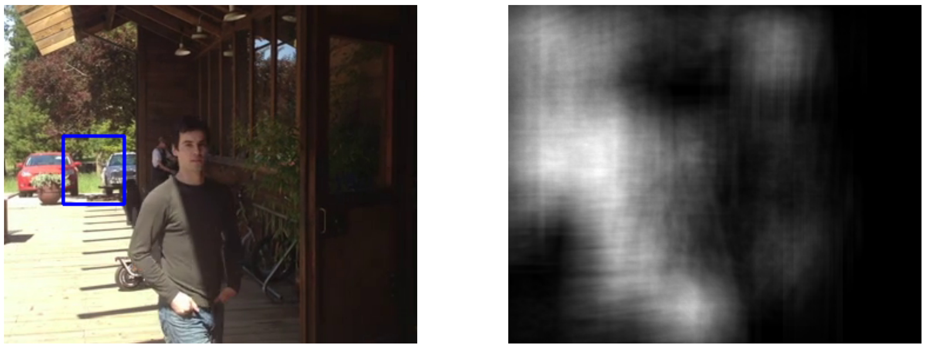


Figure 8: The left part of the image has much more points with high gradient magnitude and thus the points in this half get much more votes than the points in the right half where the ROI actually is in the shadow.

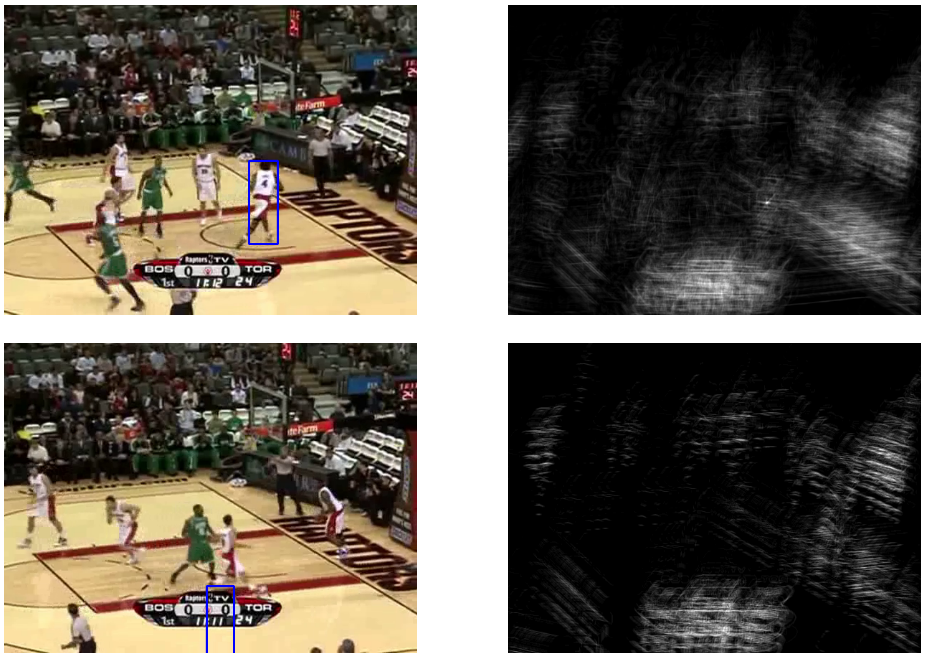


Figure 9: The player movement from the first to the second frame changes the gradient orientations of the pixels so much that the Generalized Hough Transform can not find the player anymore.

3 Combination of the two Approaches (Q5)

The first possible improvement to the Generalized Hough Transform is to apply the meanshift algorithm to the votemap instead of just selecting the argmax. This simple trick can prevent the inconsistency between predictions in successive frames (i.e., the ”jumping” of the tracking window). The results are shown in Figure 10.

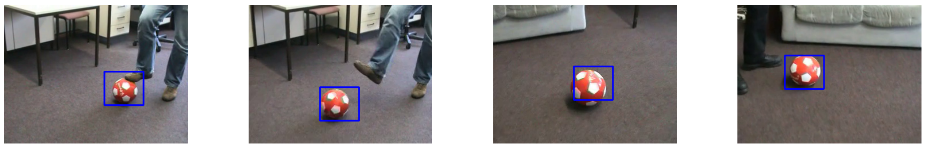


Figure 10: Where the previous basic implementation of the Generalized Hough Transform still had problems with tracking the ball position, the adapted algorithm using the meanshift algorithm works well.

The next intuitive idea is to combine the weight maps of the two different methods (i.e., the backprojection map of the histogram and the votemap of the Generalized Hough Transform). When combining them, we can make use of all the improvements that we mentioned before, meaning iteratively updating the histogram of the ROI, using different components of the HSV values and using the meanshift algorithm to track the object in the combined weight map. For combining the two different weight maps, there are different possibilities. It first makes sense to normalize both maps to the interval $[0, 1]$. Simply adding up the two weight maps would result in a bias towards the histogram-based method in most cases as the weight map of it is more dense than the often sparse voting map of the Hough Transform. A simple weighted calculation of the combined weight map could look like this:

```
weights = hough_votes * beta + backprojection_weights * (1 - beta)
```

This method of combining the two weight maps can work with a `beta` parameter that is large enough. Another method that was used in the experiments is adding the term

```
weights += hough_votes * backprojection_weights
```

to the previously computed weight map and thus weighting points with both, high votes and high backprojection weights even more. The resulting algorithm can track objects that were previosly hard to detect (in a continuous manner). An example is shown in Figure 11.

The combination leads to an improvement that comes with slightly more computational effort and some loss of stability, as the optimal initial ROI looks different for the two methods. While it is important to include the boundaries of the object being tracked (i.e., the edges with significant slope) for the generalized Hough transform, histogram-based tracking works better with narrow ROIs that do not include too much of the background to obtain a histogram that only contains information about the object being tracked and does not include the background.

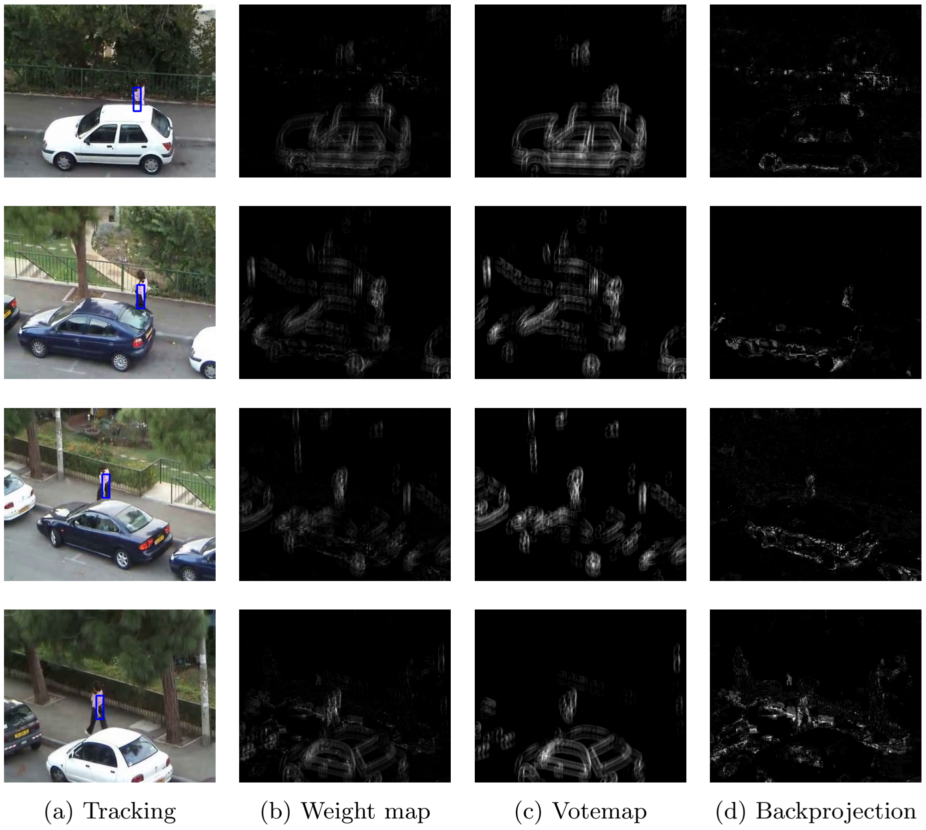


Figure 11: Combining histogram-based tracking (d) and the Generalized Hough Transform (c) using a weighted sum of the two (b) results in consistent tracking of the woman across the entire video (a).

3.1 Robustness to Aspect Changes

One of the short-comings of the tracking algorithms used so far is that they aren't robust to aspect changes of the tracked object. In many real-life use cases, this can, however, be the case (e.g., a moving object that changes in size). Since the algorithms we use are based on the meanshift algorithm on the weight map, we can use a solution that already exists to adapt to different object sizes: [1] proposed a method called Camshift that automatically adapts the size of the tracking window and is thus robust to aspect changes of the object. The method is implemented in the OpenCV library and can easily be used instead of the meanshift algorithm. It uses the basic meanshift algorithm and adapts the tracking window size after convergence of the algorithm at each step. The results are promising as the algorithm can adapt aspect ratio and size as shown in Figure 12.

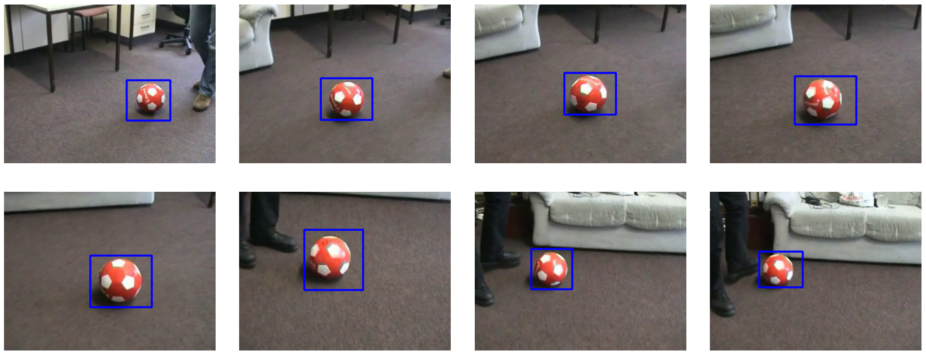


Figure 12: Using the Camshift [1] instead of the meanshift algorithm, we can adapt to different object sizes and aspect ratios.

References

[1] Gary R Bradski. Computer vision face tracking for use in a perceptual user interface. 1998.