Mean-shift algorithm: seeking & tracking method

Erazem Pušnik, 63170246

I. Introduction

This report provides the experiments of the Mean-shift algorithm. The algorithm iteratively shifts a point in the selected image and converges in the average of the data points in its neighbourhood. It is used in tracking moving objects where the tracking is based on the color of the tracked object. In the report we test the algorithm using different parameters and address its performances.

II. Experiments

The methods used in this report were implemented in *Pyhton* programming language. In this report we experiment with Mean-shift seeking and tracking methods . For testing the methods and displaying results we used the provided source code.

1) ***Mean-shift seeking:***

The mean-shift seeking method uses iteration to find the location of the maxima of an image. In mathematical way it is similar to a gradient ascent. The algorithm starts at a predefined initial point and uses a kernel in which it calculates the new *starting* location. The method calculates the distances to all point inside of the kernel from its starting point and shifts the starting point closer to the middle of the cluster. This process is repeated until convergence.

We use a predefined threshold - termination value, which tells us when the algorithm has converged. The initial value of the threshold is 0.01 and can be changed with the kernel size. Larger threshold values will result in fewer iterations which reduces the time but will in contrast reduce the accuracy of the algorithm.

The table II-1 shows the results of the algorithm with different input parameters.

parameters	I	II	III	IV	V
start	50, 50	32, 32	53, 44	58, 50	51, 48
kernel size	5×5	5×5	7×7	15×15	5×5
threshold	0.01	0.01	0.01	0.01	0.08
iterations	230	1	203	45	119
end location	51, 70	32, 32	69, 52	51, 70	53, 66

If we compare the first and the last columns of the table we notice that the threshold was raised. As expected this lowered the number of iterations the algorithm needed to apparently converge. It is important to notice that the actual maxima of the *generated response* is located at x = 70 and y = 50. The threshold missed the actual convergence point by 3 and 4 in each of the coordinates. The effect of the error depends on the requirements of the system. If time to convergence is more important than the pinpoint accuracy we can adjust the threshold accordingly.

In the rows 1, 3 and 4 in the table II-1 we compare the results based on kernel size. By increasing the kernel to size 15×15 we notice significant difference in the number of iterations compared to the 5×5 but the same end result. This approach works well when our starting point is located somewhat in the middle of our generated *response* since the size of the kernel cannot be a problem. In contrast if our starting point is located at the border of the *response* we will not be able to extract a correct patch size since it will be out of bounds of the matrix.

Additionally if our starting point is set in the region of the *response* where all values are equal to zero the algorithm will converge immediately. This is shown in the second row of the table II-1. In this case the algorithm cannot find the increase of the values inside of its kernel and cannot shift its starting point. This is somewhat solvable with the size increase of the kernel if we can avoid previously mentioned problems with the kernel size.

For our specific *response* if the starting point is the same as in the table II-1 in second row, 32, 32 the smallest kernel size which converges to the actual result is 18×18.

2) ***MS mode-seeking:***

The method *generate responses 2* is presented with the table below.

parameters	I	II	III	IV	V
start	53, 34	41, 45	30, 53	44, 59	51, 48
kernel size	5×5	5×5	7×7	15×15	10×10
threshold	0.01	0.01	0.01	0.08	0.01
iterations	197	1	197	94	10000
end location	80, 40	41, 45	80, 41	80, 41	80, 41

We notice that the behaviour is similar to the already explained results at section II-1 and will therefore not be additionally described. The actual maxima of the generated image is located at x=80 and y=40. What is noticeable is that with the kernel size 10×10 with the termination threshold is never reached. This means that the algorithm does not converge and is stopped by the limitations of iterations. What is interesting from the table II-2 we see that the result is almost right even though it did not converge.

3) ***Mean-shift tracking:***

We ran the tracker on 5 different sequences form the given database. The table below II-3 shows the performances of these sequences in terms of number of failures of our tracker.

sequence name	number of failures of the tracker
woman	3
drunk	0
hand1	5
basketball	3
gymnastics	1

4) ***Identifying failure cases::***

The table II-3 shows that the tracker works well. It has almost no problems tracking if the object is not surrounded with other objects of similar colour. This is well shown in the second example of the table II-3 named *drunk*. The car is the only object in the majority of the sequence and is isolated. The speed of the object also does not affect the performance of a tracker if the object is clearly isolated. This statement is supported by the *gymnastics* case in the table II-3 where the gymnast is moving fast compared to other cases except the *hand* which moves at similar speed but fails drastically more times due to the colour similarities of the background as well as speed of motion.

In contrast to the *gymnast* in the *woman* case the object is moving a lot slower but still results in 3 failures.

The picture 1 shows the frame after which the failure was detected. Since this algorithm is colour based and weighted

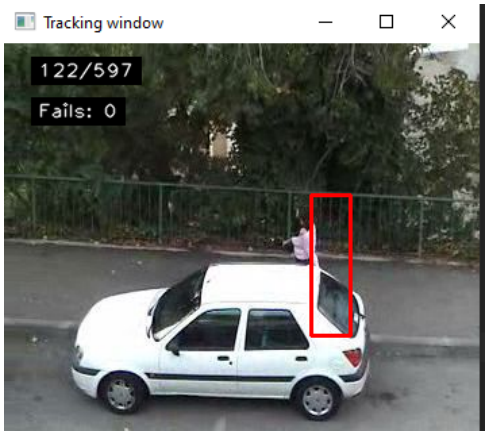

Figure 1. Failure case 1.

towards the middle of the object the reason for this failure seems to be the back windshield of the car. The woman is wearing black jeans and has black hair. At the beginning the tracker sees the whole object and in this frame the object is behind the car. The back windshield is the same colour as the objects jeans and the background is the same colour as objects hair. It looks like the objects maxima was determined somewhere at the jeans instead of the shirt which lead to the tracker choosing the wrong object to track.

The tracker then fails again at the front windshield where the same behaviour occurs. This is shown in the figure II-1 below.

Figure 2. Failure case 2.

A very similar failure case to the *woman* is case *basketball*. This case shows the failure even clearer. The basketball player moves at a decent speed and is easily tracked until he is crossed by another player from the same team. Since they both have the same jersey the tracker starts to follow the wrong player.

The tracker in figure 3 gets two players in its box and proceeds to track the top player which is wrong. In a couple of the next frames even more of the top player comes into the tracker box and it seems that the green color of the jersey and the color of players skin outweighs the bottom player which has more white color in these frames and therefore the kernel of the center finds the maxima at the wrong player.

5) ***Finding good setting of the parameters****:* The parameters which influence the performance of the tracker are number of histogram bins, alpha parameter to update the model and

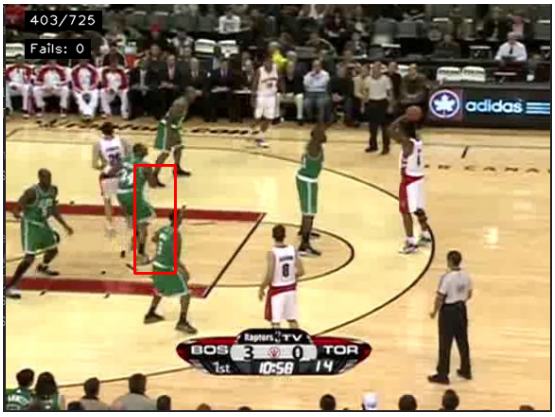

Figure 3. Basketball failure case 1.

number of iterations. The previously mentioned failure case *basketball* is resolved if the number of iterations is increased from 20 to 50 so it can satisfy the threshold mistake value of 1 pixel but the tracking speed is decreased as showed in the figure 4. Adding or reducing the number of histogram bins also impacts the tracking speed. Depending on the case more or less colour works better. For example the *woman* case works significantly better with 8 bins compared to 16.

Figure 4. In the picture on the right the algorithm has increased number of iterations to 50. Even though the frames of the sequences are not exactly the same we can clearly see that the position of the bounding box of the tracker has improved significantly in this case and the tracker does not fail. This also causes in the increase of the time consumption and only resolves this specific failure in this case. This means that we can increase the accuracy of the tracker at the price of time.

We also introduced the alpha parameter which is used to correct the model. In the selected cases it turned out that the best results were obtained when the alpha value was at 0. This means we do not update the model at all. The last parameter which we tested was termination criteria. The majority of the best results were obtained with the value 1. When increased to bigger values the results weren't as affected but when the value is low the tracking time increases significantly.

III. Conclusion

The seeking method successfully finds the maxima of the image and is later used for tracking. The tracker successfully tracks the objects and works better than expected. The failures of the tracker only appear at highly irregular movements and where the colour of multiple objects or background is almost identical to that of the tracked object. The increase of the number of iterations of the algorithm results in slightly better tracking but consumes a lot more time thus probably not making it worth doing. The performance is on average reduced by 1 failure.