Mean-Shift Tracking

Biljana Vitanova

I. Introduction

In this project, we implement a Mean Shift-based object tracker, starting with the implementation of the core mode-seeking algorithm. The tracker is evaluated on the VOT14 dataset, where we analyze its performance under different parameter settings and identify common failure cases. Additionally, we explore improvements by accounting for background information and experimenting with different color spaces.

II. Experiments

A. Mean Shift Mode Seeking

In our implementation of mode seeking, we experimented with two kernels: the Gaussian kernel with its derivative (also Gaussian), and the Epanechnikov kernel with its derivative (the uniform kernel).

We tested different initial points and kernel sizes. From Figure 1, we can see the convergence is primarily influenced by the initial position. Regardless of the bandwidth, points that are too far from the mode often fail to converge, as the gradient of the density estimate is too weak.

Smaller kernel sizes tend to converge to local maxima due to their limited scope, while increasing the kernel size improves the chance of reaching the global maximum at the cost of reduced estimation precision.

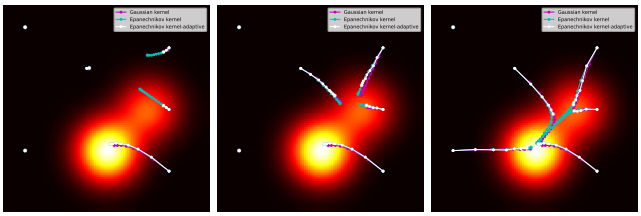

Figure 1: Convergence of the Mean Shift algorithm for different bandwidths. The left plot uses a bandwidth of 7 pixels, the middle 17 pixels, and the right 27 pixels.

To balance this trade-off, we applied an adaptive kernel approach: starting with a large bandwidth for faster convergence and gradually reducing it to improve accuracy near the maximum. However, this method is still influenced by the initial distance and kernel size, as starting with a small kernel can lead to premature convergence.

B. Mean Shift Tracking

In our tracker implementation, we used the Epanechnikov kernel and its corresponding derivative (uniform kernel). We tested the tracker on the VOT14 dataset. Running it on all video sequences resulted in a total of 30 failures.

Table I: Failure counts per sequence.

Sequence	Failures	Sequence	Failures
ball	0	skating	4
car	0	hand1	2
sphere	0	tunnel	2
fish2	2	bolt	1
woman	0	motocross	3

C. Mean Sift custom function

To further test the behavior of the Mean Shift algorithm, we created a custom function with four symmetric peaks in a 100×100 grid. We used three position types: a) close to the center, b) points between two peaks, and c) points near a single peak.

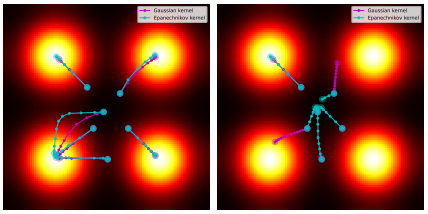

Figure 2: Convergence of the Mean Shift algorithm on a symmetric four-peak function. The left plot uses a bandwidth of 27 pixels, and the right 77 pixels.

With a bandwidth of 27, all points converged correctly to nearby maxima. However, increasing the bandwidth to 77 caused the algorithm to get confused and fail. This is because a larger kernel covers a wider neighborhood, including multiple peaks. Mean Shift then computes a weighted average of conflicting directions, pulling the estimate toward the center of the region.

D. Failure Analysis

We analyze the behavior of the tracker by looking at the failure cases observed during testing. The Mean Sift tracker has trouble in several situations that affect how well it works.

One common issue is fast object movement, like in the hand sequence videos Figure 3. The quick motion causes the tracker to lose the object because it changes position too quickly between frames.

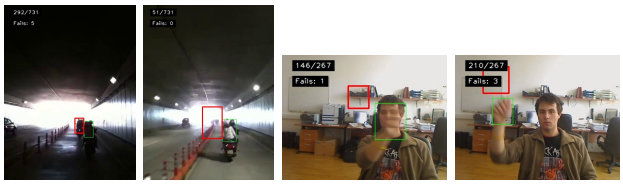

Figure 3: Failure cases: Tunnel sequence on the left, Hand2 sequence on the right.

In the tunnel video, the tracker fails for a few reasons. At first, the object is well-lit and easy to see. However, as it moves into a shadowed region, the contrast between the object and background decreases, making detection harder. At the same time, the object appears smaller, which further complicates tracking.

In the skating sequence Figure 4, the tracker fails due to strong color similarity between the object and the background, resulting in nearly identical histograms. This confuses the tracker and causes it to lose the target. In the fish sequence

Figure 4, a significant change in appearance, from a full side view to a thin, head-on profile leads to tracking failure

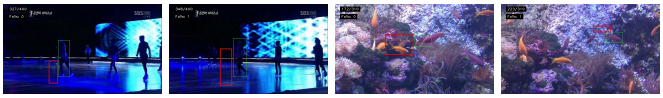

Figure 4: Failure cases: Skating sequence on the left, fish2 sequence on the right.

On the other hand, the tracker works well when the object moves slowly and remains visually distinct from the background, as seen from Figure 5. In the polarbear video, the object moves at a steady speed and is easy to separate from its surroundings. Similarly, in the surfing video, where the surfer moves smoothly and stays visually clear.

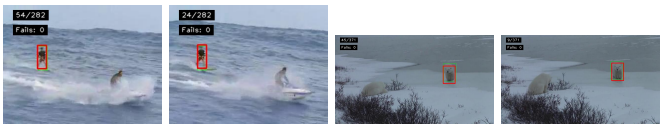

Figure 5: Success cases: Surfing sequence on the left, polar bear sequence on the right.

E. Parameter Analysis

Further, we test the performance of the tracker using different parameters. For that, we evaluate five parameters: the maximum number of iterations (*max_iter*), number of bins (*nbins*) used for histogram representation, the update rate (*alpha*) updating the tracking template over time, the stopping criterion (*eps*) which defines whether the shift is small enough to stop, and sigma (σ), which defines the kernel size.

Table II: Dependence of failure counts on the number of iterations.

eps	Total Failures	*max_iter*	Total Failures
0.01	**30**	5	38
0.1	**30**	10	**30**
1	43	20	**30**

Using a larger *eps* value (1), the tracker converged quickly, often in under 5 iterations, but with increased number of failures. A smaller *eps* (0.01) improved accuracy and reduced failures but slowed down tracking due to longer convergence time. A good trade-off between speed and accuracy was achieved with *eps* (0.1), which was used in the final setup. For *max_iter*, we set the value to 20 to ensure convergence in difficult cases. Increasing it further had no significant impact on performance.

Table III: Dependence of failure counts on the number of bins.

Number of bins	4	8	16	30	64
Total Failures	47	39	**30**	36	55

Using a smaller number of bins (4 or 8) in the histogram led to more failures, especially when the object and background had similar colors (e.g., skating sequence). Low bin counts reduced color discrimination. On the other hand, using too many bins increased sensitivity to noise and small color variations, making the tracker less stable. It also added computational cost due to higher histogram dimensionality, significantly lowering the FPS. In the final implementation we used 16 bins.

Table IV: Dependence of failure counts on σ value.

Sigma value (σ)	0.5	1	1.5	2
Total Failures	**30**	**30**	49	48

Using a smaller kernel size ($\sigma = 0.5$ or 1) resulted in fewer failures overall and performed best on complex sequences. It was effective under illumination changes and partial occlusions (e.g., *tunnel*: 2 , *fish1*: 1, *fish2*: 2 failures). In most cases, it struggled with fast motion and rapid scale changes. However, increasing the σ value did not mitigate these issues and instead introduced more failures. Therefore, we used $\sigma = 1$ in the final setup.

Table V: Dependence of failure counts on α value.

Update rate (α)	0	0.001	0.01
Total Failures	**30**	46	50

We tested the effect of updating the tracking template over time using different values of the update rate parameter (*alpha*). However, this did not lead to any improvement. In our case, the tracker performed best without updating the template.

F. Background Correction

We applied background correction to reduce the influence of background colors that overlap with the target. This improved the tracker's ability to focus on discriminative features, reducing total failures from 30 to 21 in the best setup.

Table VI: Effect of background patch size (neighborhood ratio) on tracking failures.

Neighborhood ratio	3	5	7
Total Failures	**21**	**21**	23

To test the impact of background information, we increased the background patch size, but this did not lead to better performance. Therefore, we used a background patch scaled by a factor of 3 relative to the target size.

G. Color Spaces Modeling

Additionally, we experimented with different histogram representations based on color space. However, this did not reduce the number of failures, BGR and RGB performed best overall.

Table VII: Effect of Color Space on tracking failures.

Color Space	BGR	HSV	YCbCr	RGB	Lab
Total Failures	**21**	28	26	**21**	26

III. Conclusion

In this project, we successfully implemented the Mean Shift tracker, analyzed its performance on sequences of varying complexity, improved its robustness by incorporating background information, and experimented with different color spaces.