ETH Zürich
Eidgenössische Technische Hochschule Zürich

Computer Vision
Department of Computer Science

Lab 6:
Condensation Tracker

Christos Dimopoulos - 23-941-834
cdimopoulos@student.ethz.ch

November, 2023

Contents

CONDENSATION Tracker Based On Color Histograms

In the first part of the lab exercise, we implement a CONDENSATION tracker based on color histograms using python. In this exercise we will focus on tracking just one object and consider just two prediction models (modeled by a matrix A): (i) no motion at all i.e. just noise; and (ii) constant velocity motion model. We also assume that the width and height describing the object bounding box do not change over time. The initial bounding box containing the object to be tracked will be provided by the user. The tracker should prefer samples (which we also equivalently call particles or bounding boxes) with color histograms (CH) similar to the target model.

Color histrogram

Fist of all, we implement the function *hist = color_histogram(xmin, ymin, xmax, ymax, frame, hist_bin)* that calculates the normalized histogram of RGB colors occurring within the bounding box defined by (xmin, xmax) (ymin, ymax) within the current video frame. For this purpose, we use the *cv2.calcHist* function. In the end, we normalize the histogram for it to be numerically stable and scalable.

Derive matrix A

Up next, we derive the dynamic matrix A for two different prediction models. We remind that the general prediction model, is defined by the following stochastic differential equation:

$$s_t^{(n)} = A s_{t-1}^{'(n)} + w_{t-1}^{(n)} \tag{1}$$

where $w_{t-1}^{(n)}$ is the stochastic factor of noise and $s_t = [x, y, \dot{x}, \dot{y}]^T$ is the state vector at timestep t, where x, y represent the location of center of the bounding-box, and $\dot{x}, \dot{y}$ the velocities in the x and y direction.

- **No motion at all:** in this case we completely ignore the velocities and the state vector s reduces to its first two elements as $s_t = [x, y]^T$ and A becomes a 2x2 identity matrix:

$$A = I_{2\times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2}$$

- **Constant Velocity Motion Model:** In this case, we also have to take into consideration the velocities of the bounding box. The new state vector is predicted as $x_t = x_{t-1} + \Delta t \dot{x}_{t-1}$ and $y_t = y_{t-1} + \Delta t \dot{y}_{t-1}$ where for simplicity we take as time step $\Delta t = 1$. Hence the dynamic matrix A becomes:

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Propagation

Next, we implement the function *particles = propagate(particles, frame_height, frame_width, params)*, which propagates the particles given the system prediction model (matrix A) and the system model noise represented by params.model, params.sigma position and params.sigma velocity, according to Equation 1. It is worth mentioning, that for the no motion model the state vector consists of only two elements, the position of the center of the bounding box, while for the case of the constant velocity motion model, we have two more elements that refer to the two velocities. Lastly, we use the parameter frame height and frame width to make sure that the center of the particle lies inside the frame, otherwise we move to the borders.

Observation

After that, we implement the *observe* function that makes observations i.e. compute for all particles its color histogram describing the bounding box defined by the center of the particle and bbox height and bbox width. More specifically, for each particle, we compute the new histogram using the *color_histogram* function, then we compute the χ^2 distance of the computed histogram and the target color histogram, using the provided function *chi2cost.py*. and compute the weights as a Gaussian with variance σ. Lastly, we normalize the weights by dividing with the sum of particles weights.

Estimation

Simply enough, we implement the function *mean_state* that estimates the mean state given the particles and their weights, as a simple matrix multiplication, according to the formula:

$$E[s_t] = \sum_{n=1}^{N} \pi_t^n s_t^n \tag{4}$$

Resampling

Finally, we implement the *resample* function. We want to choose more weighted particles more often. Therefore, we used *np.random.choice* to choose indices at random with the given particle weights as a probability distribution. We replace both the particles and their weights by their respective counterparts at those random indices. Through this method, we shift the average even more towards the highly weighted region where the object to be tracked most probably is to be found.

Experiments

We are provides in the release three videos: video1.wmv, video2.wmv, and video3.wmv. Each video shows a moving object in different conditions:

- video1.wmv shows a moving hand with a uniform background
- video2.wmv shows a moving hand with some clutter and occlusions.
- video3.wmv shows a ball bouncing.

In the second part of this lab, we track the objects in these videos using the tracker code you wrote. We experiment with the three videos with various parameters (params), and determine their effect on tracking performance.

video1.wmv

Firstly, we experiment with **video1.wmv**. The hand in this video is relatively easy to track, so we do not conduct too many experiments, just simple debugging. After several trials, we conclude that optimal results are given for this set of parameters: $hist_bin = 16$, $a = 0.8$, $\sigma_{\text{observe}} = 0.1$, $num\_particles = 300$, $\sigma_{\text{position}} = 15$ and no motion model just noise.

The above combination of parameters seems to give an almost perfect results, since the hand gets tracked very well, while a posteriori mean is a bit more accurate and 'ahead' of a priori one. In general, the hand can be tracked very well just with minor tuning of the parameters. Therefore, the code seems to be working without bugs.

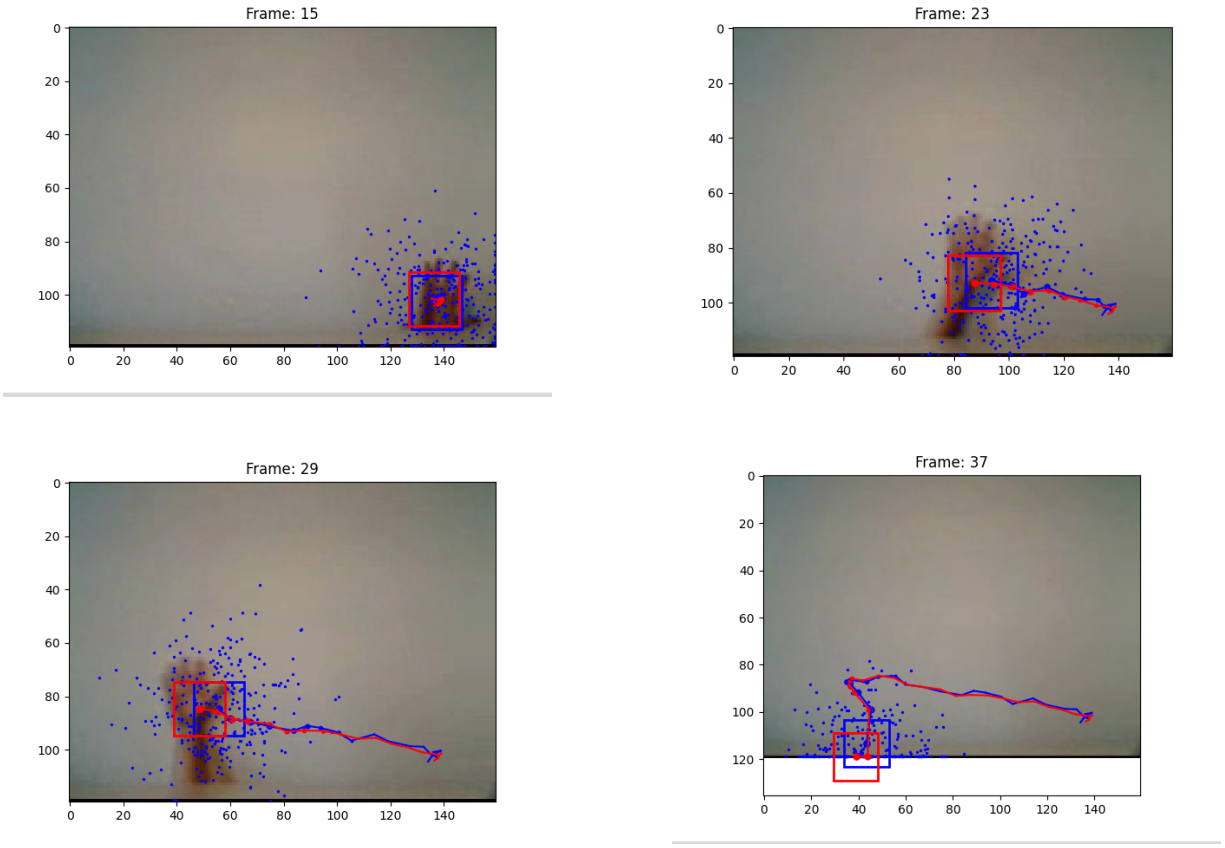

Figure 1: Tracking for video1.

video2.wmv

Next, we experiment with video2.wmv, which shows a moving hand with some clutter and occlusions. We conduct several experiments by varying the different parameters, as showcased in the following table.

Exp. #	histbin	α	σ_{observe}	model	#particles	σ_{position}	σ_{velocity}	initial vel. x	initial vel. y	Observation
1	16	0.8	0.1	0	300	15	-	-	-	Hand gets tracked in the beginning, but lost after the occlusion behind the milk carton
2	16	0.2	0.1	0	300	15	-	-	-	Better, medium successful, finds it after milk carton but still loses the hand eventually
3	16	0.2	1	0	300	15	-	-	-	No improvement, still can't overcome occlusion and background
4	16	0.2	0.1	0	600	15	-	-	-	Successfully overcomes the occlusion and background and tracks hand to the end
5	16	0.2	0.1	0	150	15	-	-	-	No improvement, can't fully overcome occlusion and background
6	**16**	**0.2**	**0.1**	**0**	**600**	**20**	-	-	-	**Also successful, a bit smoother**
7	16	0.2	0.1	0	600	10	-	-	-	No improvement, can't fully overcome occlusion and background
8	64	0.2	0.1	0	600	20	-	-	-	Not successful, gets distracted with background and loses hand
9	8	0.2	0.1	0	600	20	-	-	-	Not successful, gets distracted with background and loses hand
10	16	0.2	0.1	1	600	20	1	5	0	Also successful, but more 'shaky'
11	16	0.2	0.1	1	600	20	10	5	0	Also successful, but even more 'shaky'
12	16	0.2	0.1	1	600	20	0.1	5	0	Also successful, but more 'shaky'

Table 1: Experiment Results for video2

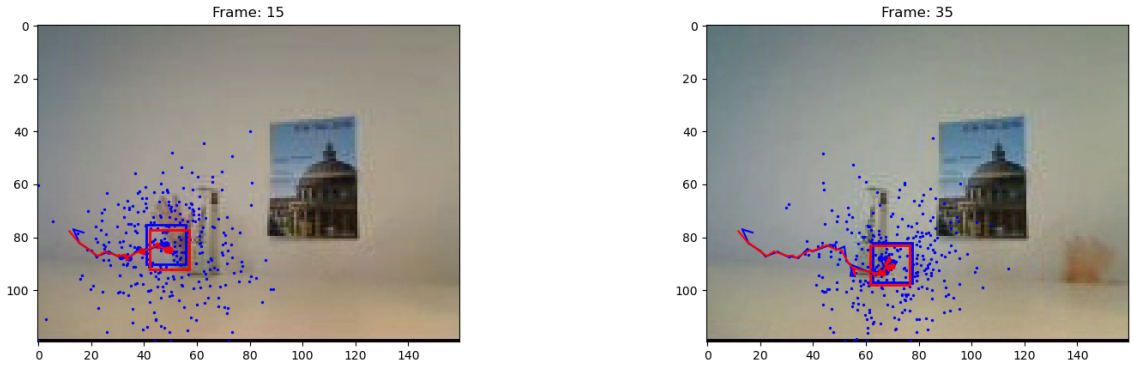

Figure 2: Experiment 1

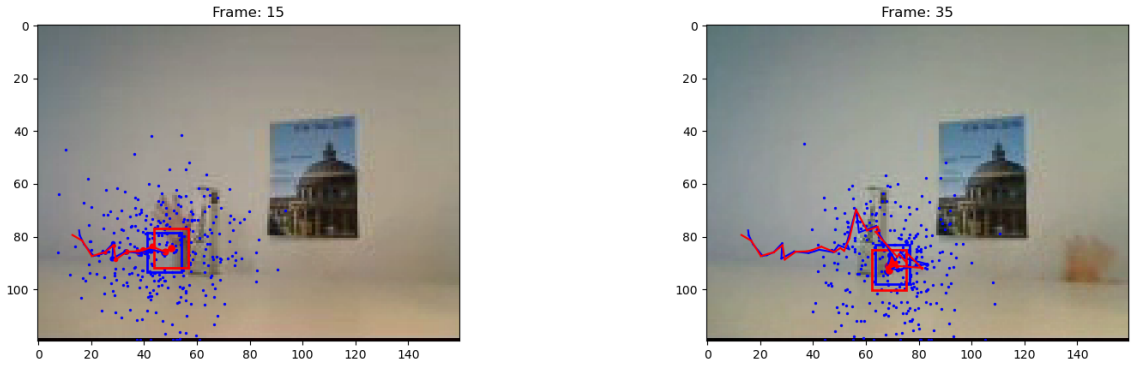

Figure 3: Experiment 2

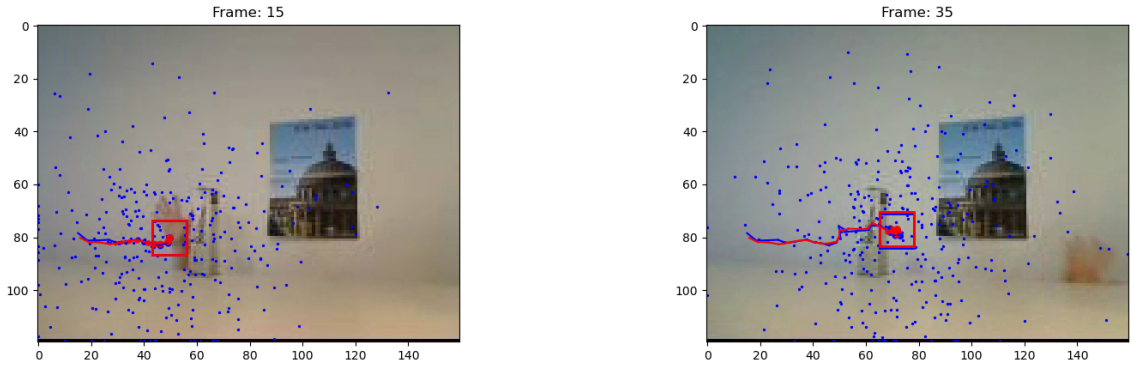

Figure 4: Experiment 3

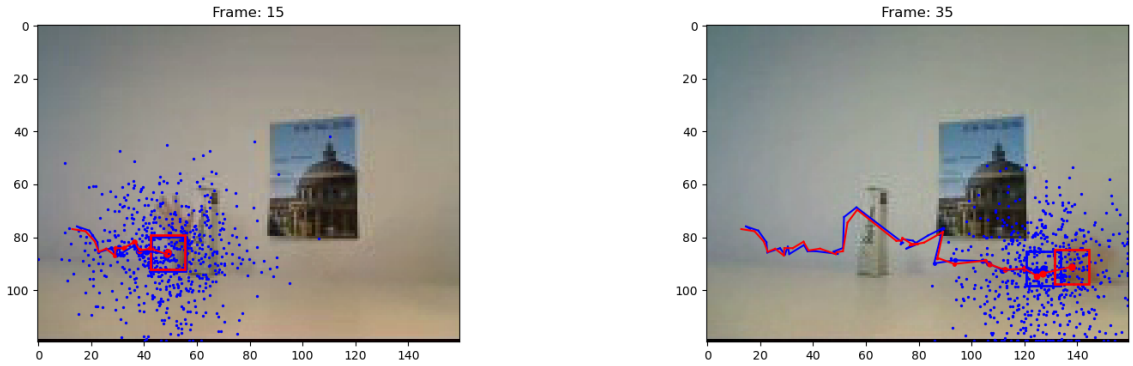

Figure 5: Experiment 4

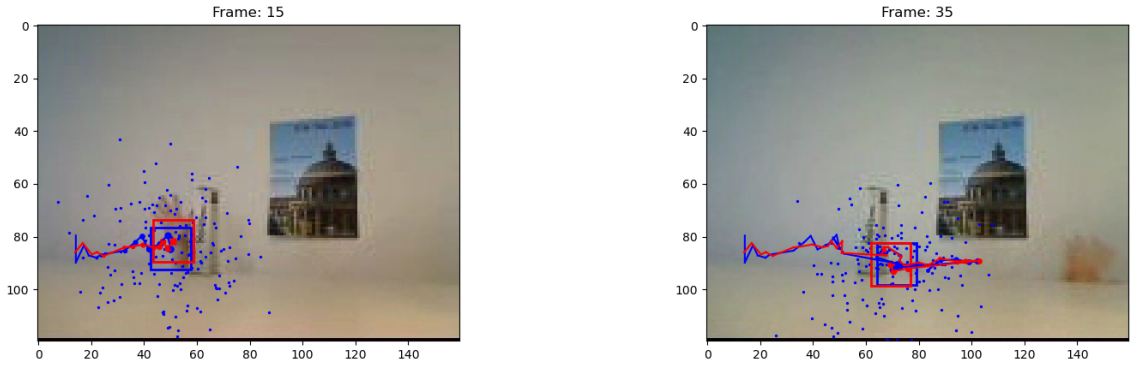

Figure 6: Experiment 5

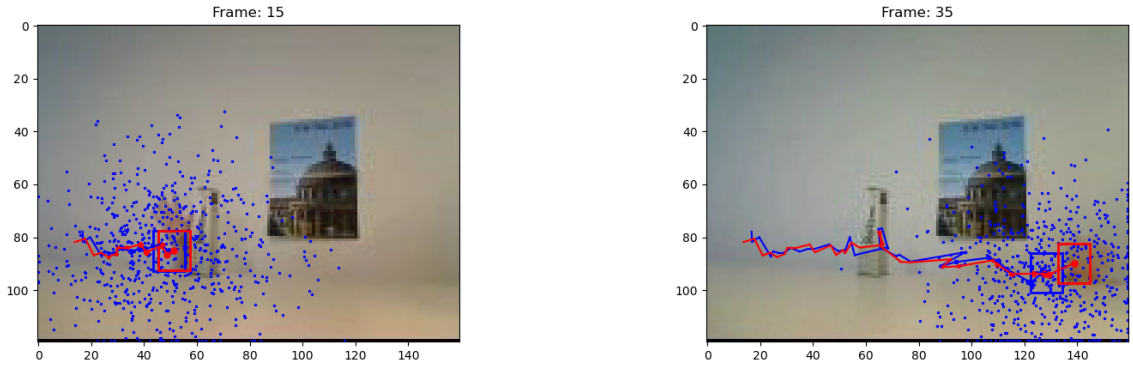

Figure 7: Experiment 6

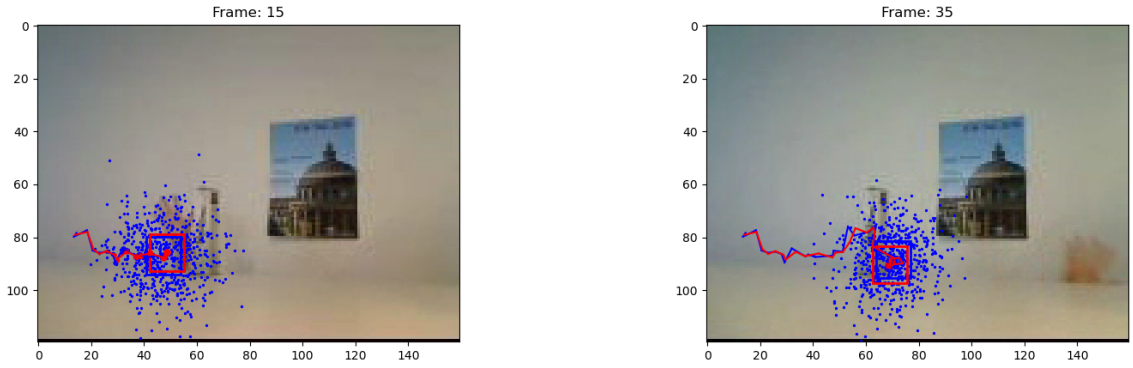

Figure 8: Experiment 7

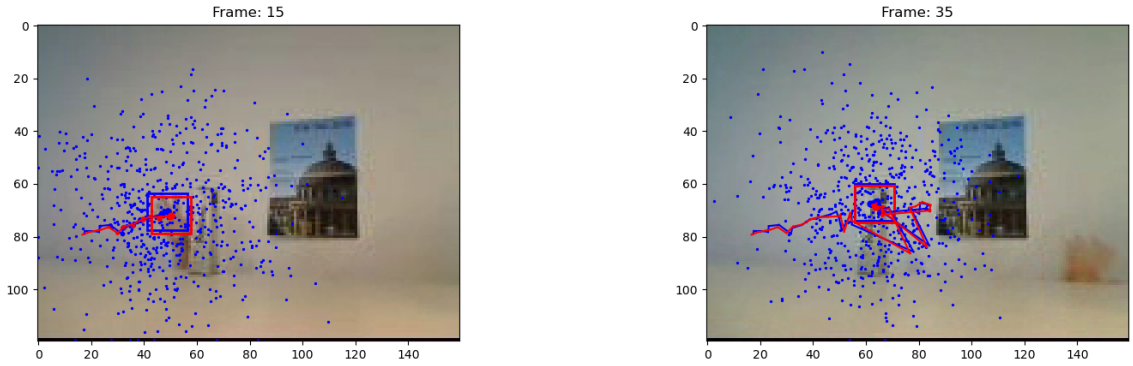

Figure 9: Experiment 8

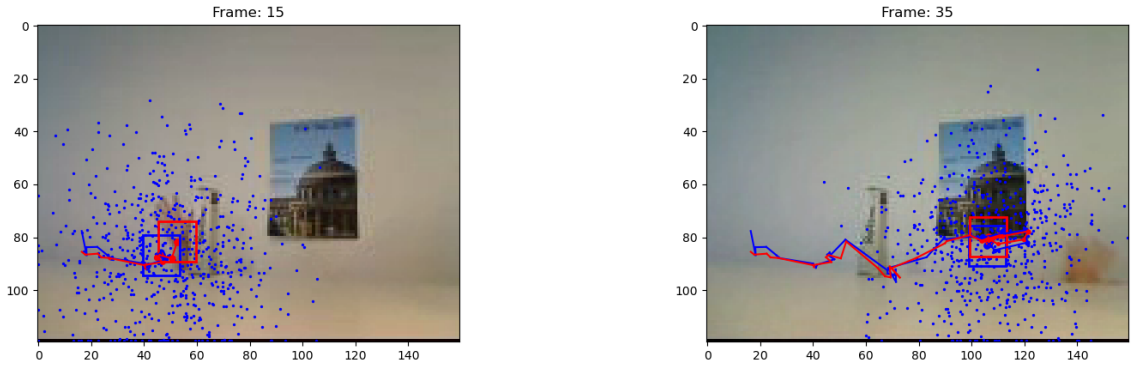

Figure 10: Experiment 9

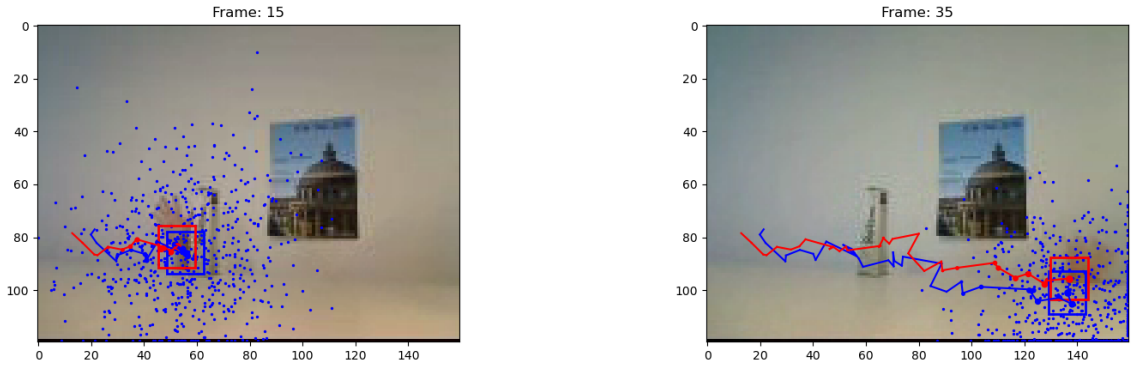

Figure 11: Experiment 10

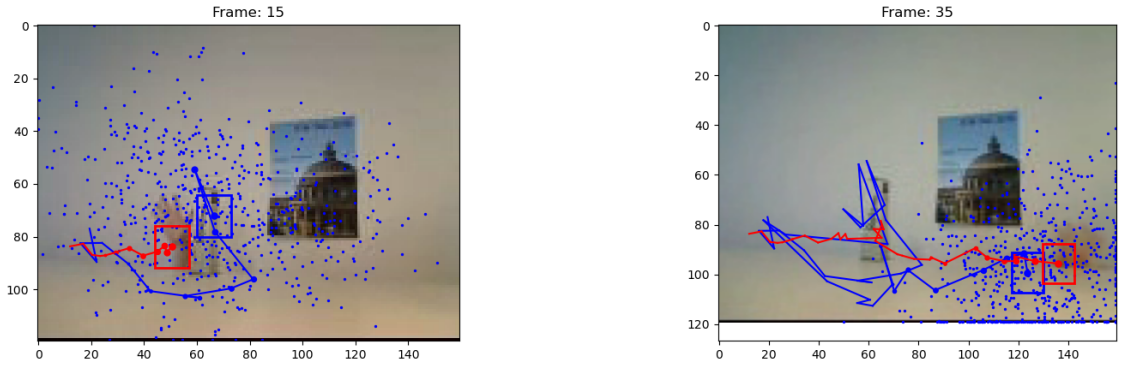

Figure 12: Experiment 11

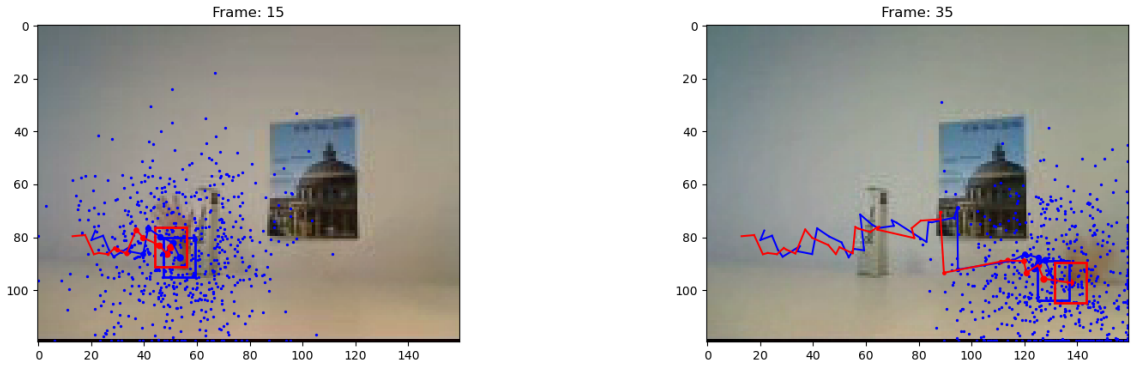

Figure 13: Experiment 12

What is the effect of using a constant velocity motion model?

As demonstrated in Experiments 10-12, incorporating the constant velocity model into the system does not appear to yield significant improvements in my particular scenario. Notably, the a priori estimates

become more unstable and shaky, especially when the $\sigma_{velocity}$ is high, leading to less smooth and accurate tracking. This phenomenon is likely attributed to the hand being part of a human, and consequently, it does not adhere strictly to a constant velocity motion. The model is thus required to continually adjust and correct for subtle changes in velocity. However, further refinement of all parameters while utilizing the constant velocity model could potentially result in outcomes that are comparable or even superior to those observed in Experiment 6.

What is the effect of assuming decreased/increased system noise?

The impact of adjusting the position noise assumption, indicated by the parameter σposition, is evident in experiments 6 and 7. Notably, increasing this parameter in my context resulted in a smoother overall system performance. This adjustment appears to enable the estimation to incorporate sparser point clouds more effectively, facilitating the detection of the hand behind occlusions or the entire arm against the background. Consequently, it enhances the model's performance. Conversely, decreasing the σposition makes the entire model less stable and more prone to challenges after occlusions, employing the opposite rationale as mentioned earlier.

The influence of altering the assumption on velocity noise, denoted by the parameter σvelocity in the constant velocity model, is explored in experiments 11 and 12. The findings reveal that an increase in velocity noise leads to more jittery predictions, resulting in a less smooth model. Consequently, it is advisable to maintain this parameter reasonably low, allowing for some variation in velocity as expected in the natural movement of a human hand.

What is the effect of assuming decreased/increased measurement noise?

Now, let's delve into experiment 3, where I manipulate the assumption regarding measurement noise using the parameter σobserve. In my scenario, increasing this parameter has minimal impact on the outcomes. Surprisingly, even at low levels, satisfactory results are attained in the later phases of tuning. However, it can be asserted that, akin to σposition, elevating σobserve creates a somewhat sparser observation area. Consequently, this adjustment eventually facilitates overcoming certain occlusions and backgrounds. Notably, similar effects can already be achieved by fine-tuning only the parameter σposition.

video3.wmv

Last but not least, we experiment with video3.wmv, which shows a a ball bouncing. We conduct the following experiments, starting with the best parameters for video2 and then varying them, as showcased in the following table.

Exp. #	histbin	α	σ_{observe}	model	#particles	σ_{position}	σ_{velocity}	initial vel. x	initial vel. y	Observation
1	16	0.2	0.1	0	600	20	-	-	-	Very successful
2	16	0.8	0.1	0	600	20	-	-	-	Also successful, but slightly more 'shaky'
3	16	0.2	1	0	600	20	-	-	-	Not successful, gets distracted too much by the border
4	16	0.2	0.1	0	1200	20	-	-	-	Successful, but no improvement
5	16	0.2	0.1	0	300	20	-	-	-	Also successful, but slightly more 'shaky'
6	16	0.2	0.1	0	600	30	-	-	-	Successful, but no improvement
7	16	0.2	0.1	0	600	10	-	-	-	Also successful, a bit smoother
8	64	0.2	0.1	0	600	10	-	-	-	Successful, but no improvement, slightly drifts off to the bottom
9	8	0.2	0.1	0	600	10	-	-	-	Also successful, a bit smoother
10	16	0.2	0.1	1	600	10	1	5	0	Very similarly successful
11	16	0.2	0.1	1	600	10	10	5	0	Quite successful, but more 'shaky', loses it in the end
12	16	0.2	0.1	1	600	10	0.1	5	0	Very successful and smooth too

Table 2: Experiment Results for video3

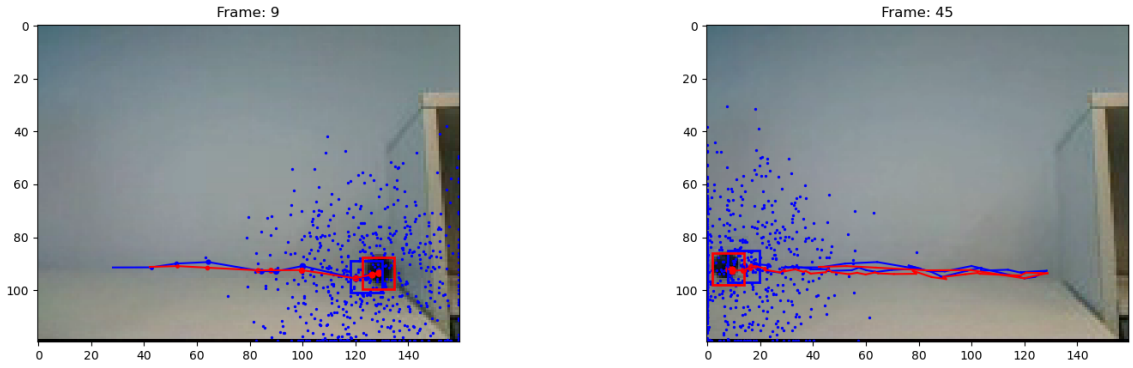

Figure 14: Experiment 1

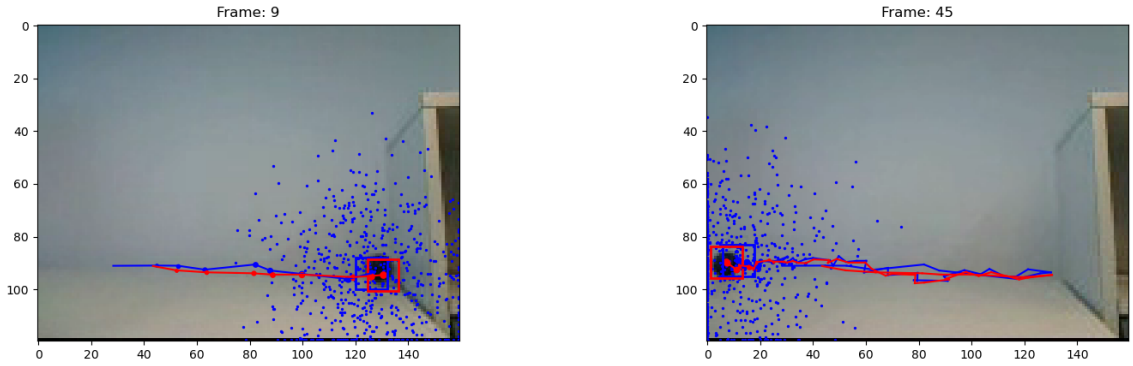

Figure 15: Experiment 2

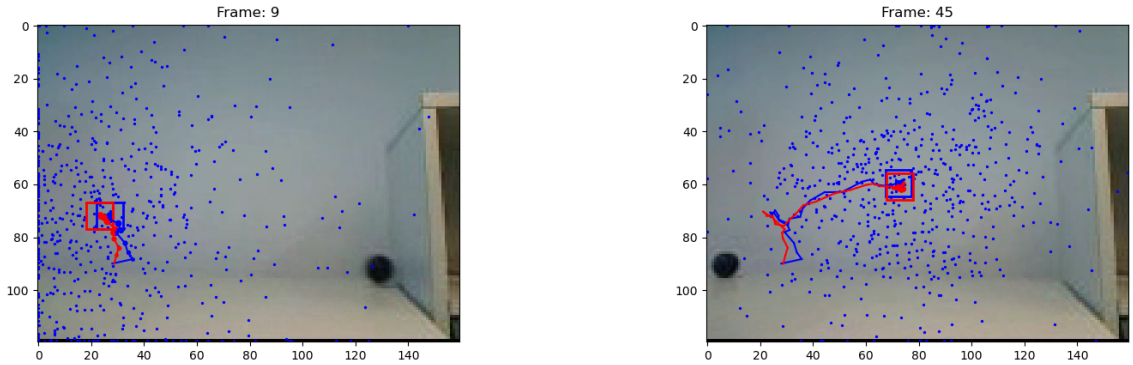

Figure 16: Experiment 3

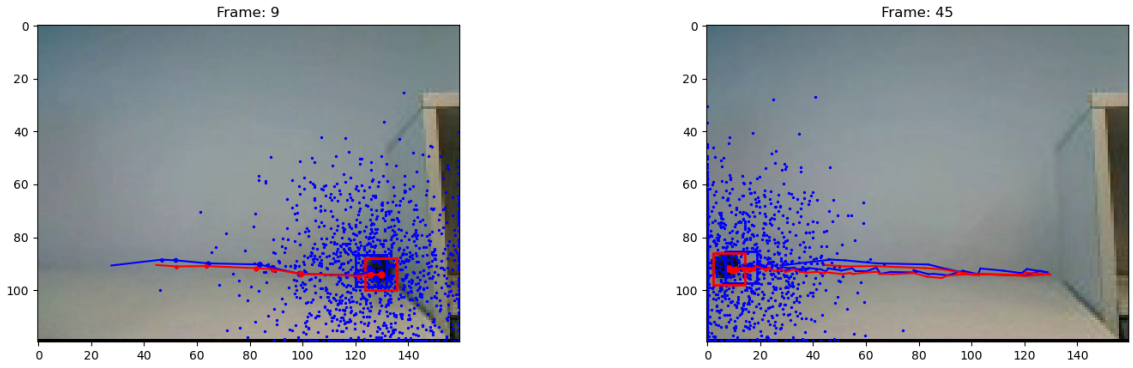

Figure 17: Experiment 4

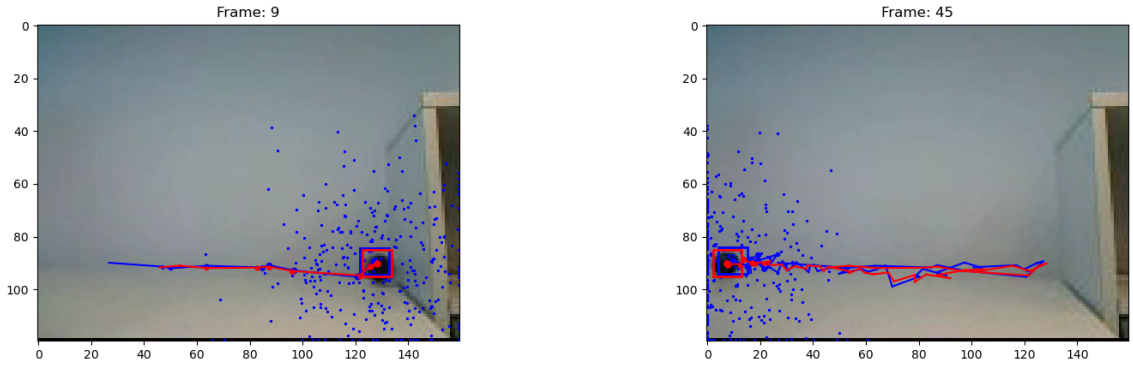

Figure 18: Experiment 5

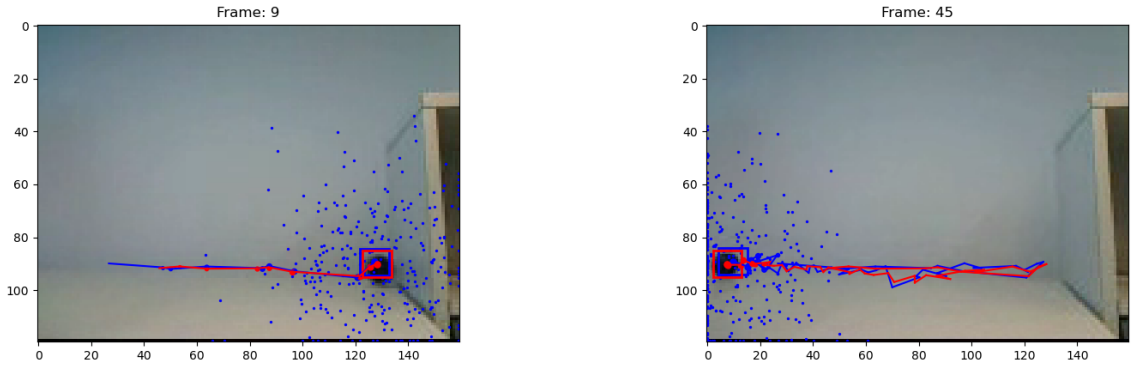

Figure 19: Experiment 6

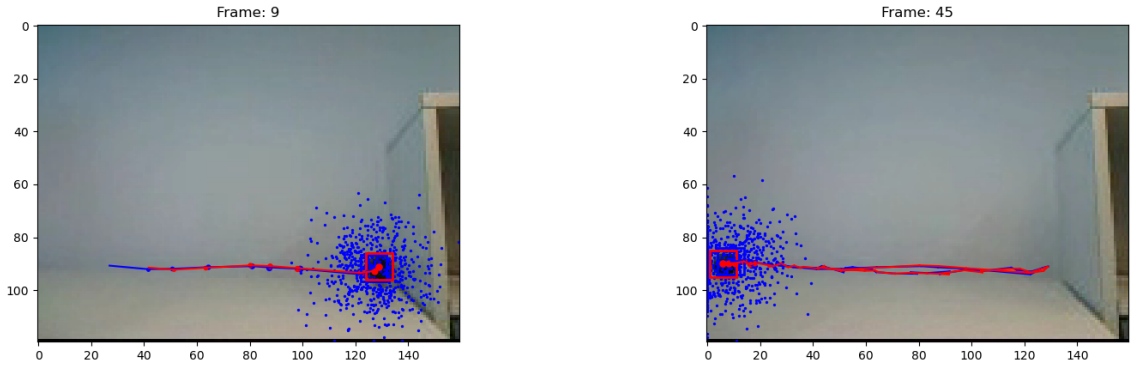

Figure 20: Experiment 7

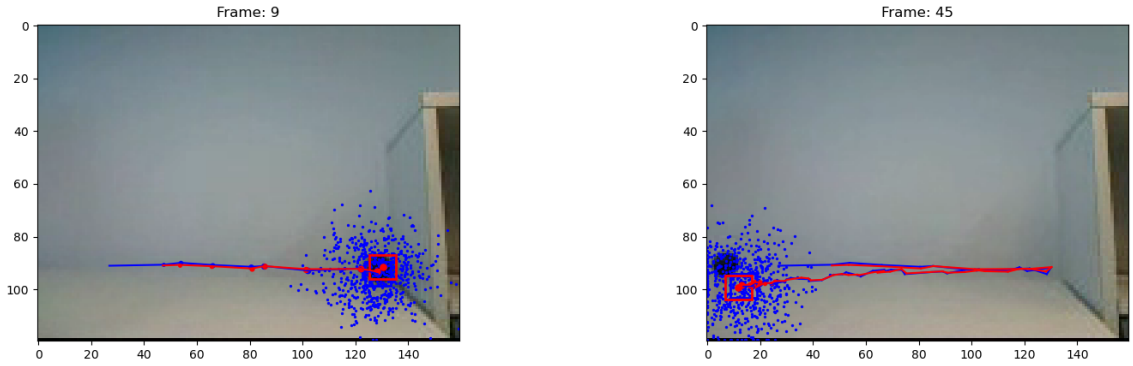

Figure 21: Experiment 8

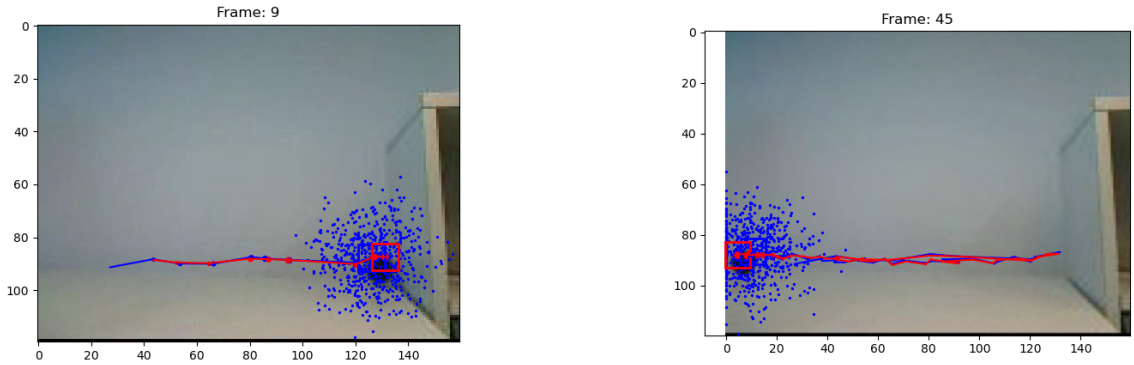

Figure 22: Experiment 9

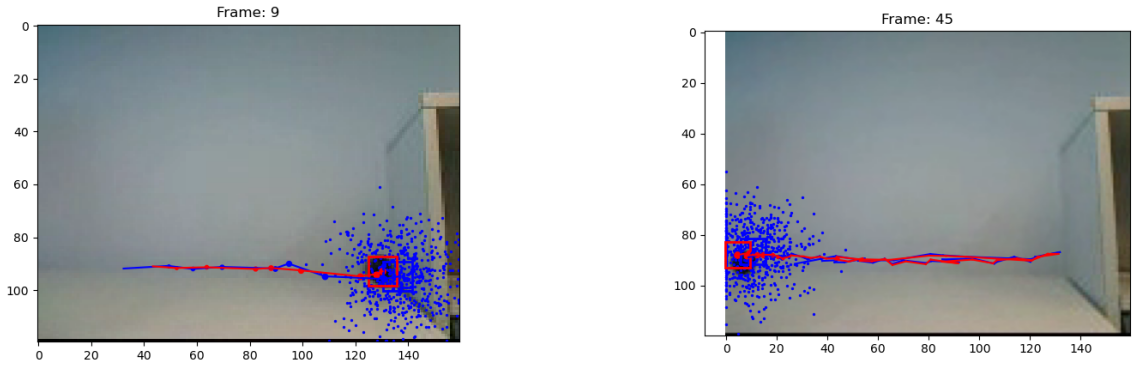

Figure 23: Experiment 10

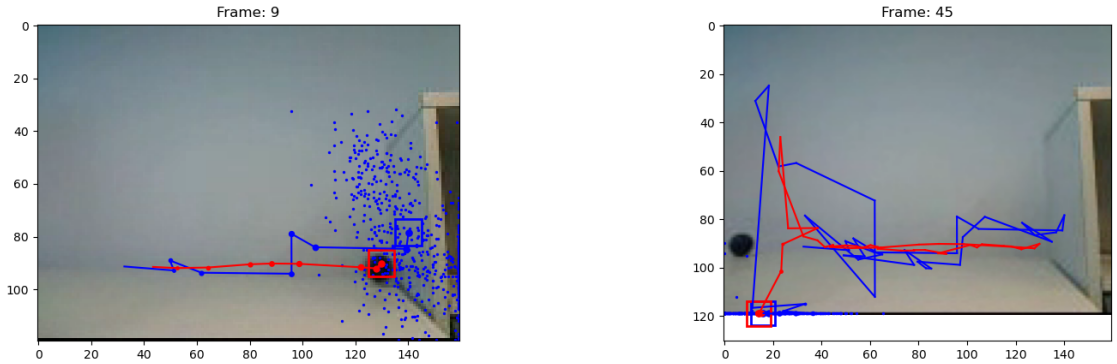

Figure 24: Experiment 11

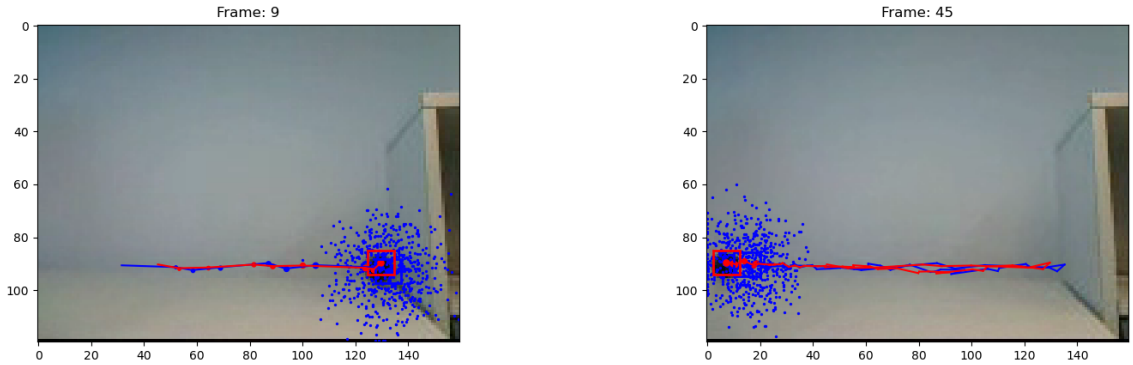

Figure 25: Experiment 12

Are you able to track the ball? Why or why not?

The optimal parameters of video2 seem to be tracking the ball well enough. The distinct black color of the ball makes it easily distinguishable from the background, and its consistent movement aids in efficient

tracking. Both the model without a velocity assumption but with an adequate level of noise assumption, as well as the model with a constant velocity, yield smooth and anticipated outcomes. This is attributed to their capacity to accommodate sufficient flexibility in locating the ball in the subsequent frame and adjusting the estimate to its new position.

What is the effect of using a constant velocity motion model?

Observing Experiments 10-12, the integration of the constant velocity model proves effective and potentially yields the most optimal outcomes, as evident in Experiment 12. However, in certain configurations, the a priori estimates tend to become more unstable, particularly with elevated σvelocity, resulting in less smooth and precise tracking. This is likely attributed to the ball undergoing minimal velocity changes in reality, making the noise component inherently small. Consequently, the model needs to continuously adapt and correct for these minute velocity fluctuations. Nevertheless, further fine-tuning of all parameters while employing the constant velocity model could potentially unveil even more enhanced results, comparable or potentially superior to those observed in Experiment 12.

What is the effect of assuming decreased/increased system noise?

We can observe the impact of adjusting the position noise assumption using the parameter σposition in experiments 6 and 7. Decreasing it in my scenario leads to a smoother overall system, given the minimal noise in the ball's position evolution. Consequently, this adjustment enhances the model's performance. On the contrary, increasing σposition makes the entire model slightly less stable and introduces a subtle level of shakiness.

The influence of altering the assumption on velocity noise, denoted by the parameter σvelocity in the constant velocity model, is examined in experiments 11 and 12. The findings indicate that heightened velocity noise results in more jittery predictions, leading to a less smooth model. As mentioned earlier, this is likely because the ball maintains a very constant velocity, resulting in minimal noise in its velocity. Hence, it is recommended to maintain a reasonably low standard deviation for σvelocity.

What is the effect of assuming decreased/increased measurement noise?

Now, let's examine experiment 3, where I manipulate the assumption regarding measurement noise using the parameter σobserve. In my scenario, increasing σobserve has a notable impact on the results: the sparser observations appear to divert the model's attention with the picture frame, resulting in an unsuccessful tracking of the ball. Consequently, maintaining σobserve reasonably low, similar to σposition, emerges as the most optimal conclusion.

Conclusions

What is the effect of using more or fewer particles?

The impact varies depending on the setup. In the case of simple hand tracking in video 1, the number of particles significantly influences the results. Insufficient particles, tend to result in tracking failure, likely due to the hand requiring a certain point density for effective tracking. Conversely, an excess of particles, causes the entire arm to carry more weight, leading to the tracking of the arm rather than just the hand.

In video 2, the effect becomes even more pivotal. Experiment 4, with more particles and denser modeling of the surroundings, proves beneficial in overcoming occlusions like the milk carton and distractions from

the image background. On the contrary, experiment 3 experiences decreased performance with fewer particles, making it even more challenging to bias the model over occlusions or backgrounds.

For video 3, there appears to be minimal observable effect when adjusting the number of particles in experiments 4 and 5. This is likely due to the ball moving openly with a very constant velocity, a characteristic that can be captured effectively with both dense and sparse modeling.

What is the effect of using more or fewer bins in the histogram color model?

In all videos, varying the number of bins in the histogram color model only resulted in a minor effect on the histogram.. The most significant exception is observed on video 1 when increasing the number of bins, introducing more colors to the model. In this case, it appears to have slightly diverted the model's focus towards tracking the entire arm rather than just the hand. Additionally, it is worth noting that using more bins in my context led to a slightly increased computation time for the code, although it remained within reasonable limits.

What is the advantage/disadvantage of allowing appearance model updating?

The impact varies across videos. In the case of video 1, allowing for more frequent updates improves the results. This is likely because the hand is easily trackable and undergoes noticeable changes in appearance throughout the video, such as variations in finger positions in the open air.

However, in video 2, characterized by a significant occlusion, excessive model updating can lead to challenges. Experiment 2 illustrates that a lower α (update rate) distinctly enhances the results. This adjustment is crucial to prevent the searched-for appearance from changing too much behind the milk carton, making it unrecognizable once the hand reappears.

For video 3, the impact of the α value is marginal. In my specific case, increasing α yields slightly inferior results compared to mostly sticking with the previous appearance (low α). Nevertheless, considering the physical nature of the ball, which should exhibit minimal changes in appearance throughout its movement, α becomes a negligible parameter in this scenario.

References

[1] Isard, M., Blake, A. - 'CONDENSATION - conditional density propagation for visual tracking', IJCV 1998.