Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision
and Geometry Lab

Computer Vision Lecture

# Assignment 6 Report

**Autumn Term 2021**

**Author:**
Pascal Lieberherr

# Contents

# Chapter 1

# Report

## 1.1 Approach

### 1.1.1 Color histogram

In order to compute the color histogram I loop over all 3 color channels. My color histogram has the shape [3,number histogram bins]. To compute the histogram for one color channel I use numpy.histogram(). After that I normalize the histogram such that it sums to 1. This procedure is done for every color channel.

### 1.1.2 Matrix A

For the model with zero velocity the A matrix is simply the identitiy $I_{2\times2}$. Since the previous position is also the propagated position + the noise because our model assumes zero velocity. The states are [x,y] i.e. x and y position.

For the model with constant velocity the A matrix is [[1,0,1,0],[0,1,0,1],[0,0,1,0],[0,0,0,1]]. This is because the velocity has unit [pixel/frame] thus $x_{k+1} = x_k + \dot{x}_k$. k denotes the k-th frame of the video stream. The last two rows of A result form the fact that the velocity is constant: $\dot{x}_{k+1} = \dot{x}_k$. The state is $[x,y,\dot{x},\dot{y}]$

### 1.1.3 Propagation

For the propagation I first build the noise vectors w by sampling from the normal distribution with zero mean and the respective variance: for x and y this is $sigma_{position}$ and for the velocity it's $sigma_{velocity}$. Once I built the noise vector I can carry out the propagation step by matrix multiplication / addition: $x_{k+1} = Ax_k + w$ whereas $x$ denotes the full state.
After I obtained the propagated particles I check whether they are inside the frame. If not, I clip them to the boundary of the frame.

### 1.1.4 Observation

For the observation part I loop over all particles. For each particle I compute the color histogram by considering a box around each particle of the same size as was used for the target histogram. I use the earlier implemented function colorhistogram(). Then I check how similar the particle histograms are with the target histogram. Note that the target histogram is computed when the user draws the box around the object of interest. The more similar the two histograms are the higher the particle will be weighted. We use the chi squared distance as a similarity measurement for the histograms. This Chi squared distance is then fed into equation 6 of the assignment description. Now we have the weights for all particles. Next, I normalize them such that they sum up to one.

### 1.1.5 Estimation

Now, we compute the weighted mean for the weighted particles. This gives us the new estimate. In the plots this new estimate is represented with a red box.

### 1.1.6 Resampling

Before we can start the next iteration of this algorithm we need to sample new particles from the discrete distribution of the weighted particles. This means the higher weighted particles will be sampled with higher likelihood. After this step we get n particles, all with the same weight. Now we can start again with the propagation step.

## 1.2 Experiments

### 1.2.1 Effect of different Parameters

Number of particles:
The more particles we use, the higher the computational cost. We can notice that the tracking is not realtime anymore if we e.g. use 2000 particles. At the same time the variance of the estimate decreases when more particles are used. Generally this results in a more smooth tracking path.

Sigma position and sigma velocity:
These two std. deviations are the system/process noise. The lower they are the more we trust the propagation step i.e. our model which is the A matrix. The more certain we are that the model of e.g. constant velocity is correct, the lower we can choose these values.

Sigma observation:
This value is the std. deviation of the Gaussian that specifies the probability of each particle based on the Chi squared distance. The lower it is, the more weight is given to the particles with a low Chi squared distance. The less the object we want to track to changes it's appearance, the better the target histogram represents the object we want to track to during the entire image stream. Thus, I chose sigma observe small if the object we want to track to doesn't change it's appearance a lot during the entire image stream.

Alpha:
This parameter updates the target histogram as a weighted sum of the estimated color histogram and the previous target histogram. If the object we want to track to changes it's appearance during the video I chose alpha rather large. Otherwise it can be set to zero.

Hist bin:
The number of bins should be sufficiently large such that the histogram is distinctive enough. To illustrate this: Think of a particle that is far away from the object (we want to track to) but has the same histogram as the particle that is perfectly in the center of the object. As a result the particle far away would incorrectly get a high weight instead of a low. Thus, the number of bins should be not to small like e.g. 2 or 4. In terms of the run time I only noticed a very slight increase when the number of bins is high. When light conditions change, the appearance of the object also changes. To make the tracking more robust against such changing lightning conditions I would choose the number of bins not too high. Because, the target histogram and the histogram of the object are more likely to be similar during changing lightning conditions when the number of bins is not too high. This adds some extra robustness for changing lightning conditions. When I chose e.g. number of bins = 2 the tracking failed in video2 because it was not distinctive enough. I did not see the tracking to fail in any of the videos when I set the number of bins = 256 (the max possible value). For all videos I achieved good results with 8 bins.

### 1.2.2 video1

The challenge for this video is that only the fingers are visible at the beginning. Then after a few frames the arm plus the hand are visible. I think this is the reason why it fails to track only the fingers instead of the arm. There are only very few solutions in the x direction but many along the y direction. Because what the color histogram actually sees is a few skin colored pixels and a few background pixels. This results in ambiguous solutions especially in the y direction. A box partially seeing the arm and partially seeing the background has the same color histogram as one that sees the fingers and the background between the fingers. As a result, it tracks the arm instead of the fingers, see fig. 1.1. To account for that challenge I would use a more sophisticated feature descriptor than just a color histogram. Because imagine an image and you compute the color histogram for it. Then you shuffle all the pixels in the image and again compute the color histogram. Both color histogram will be identical. A feature descriptor like e.g. HOG which does not suffer from this would maybe be able to track the fingers. I achieved the best results with the following parameters:
Parameters for video1 for a successful tracking:
hist bin=16, alpha = 0.01, sigma observe = 0.3, model = 0, num particles = 500, sigma position = 15, sigma velocity = 1, initial velocity = (1,10)

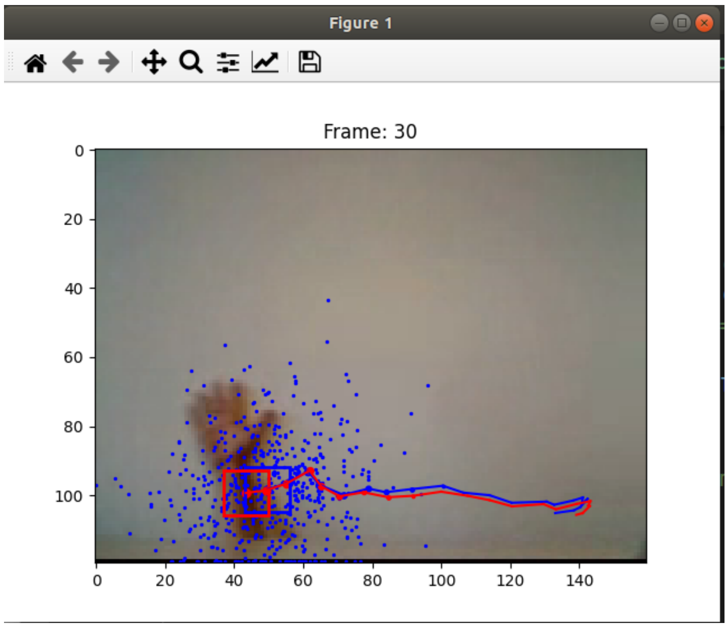

Figure 1.1: Tracking works for video 1

The tracking failed for this parameters, see fig. 1.2:
hist bin=16, alpha = 0.6, sigma observe = 3, model = 0, num particles = 100, sigma position = 5, sigma velocity = 1, initial velocity = (1,10)
Particularly, because the number of particles is low and the alpha value high. Once we get an estimate that is off we update the target histogram base on a particle that is not on the object we want to track to. I also decreased sigma position such that I provoke a wrong estimate because the particles don't scatter enough.

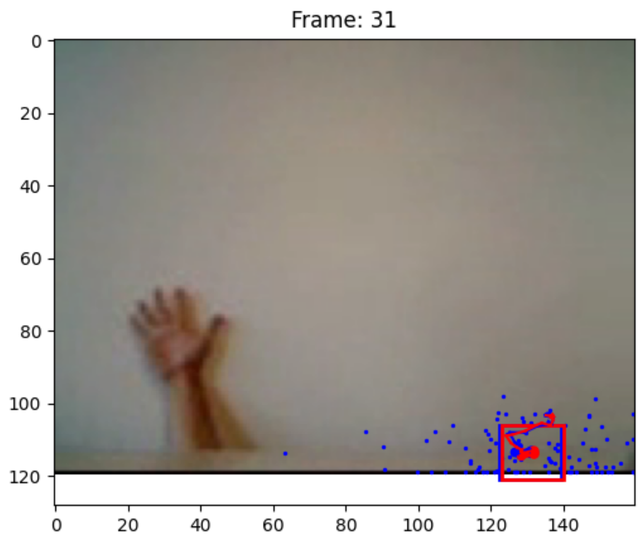

Figure 1.2: Tracking does NOT work for video 1

### 1.2.3 video2

In this video we face occlusion and background changes. We notice a constant velocity of the object from left to right. Thus, we are confident in our model with constant velocity. When the hand is occluded it comes in handy when we have a constant velocity model that makes sure we keep going to the right even though the hand is currently occluded.
Parameters for video2 for a successful tracking:
hist bin=16, alpha = 0.1, sigma observe = 0.5, model = 1, num particles = 500, sigma position = 15, sigma velocity = 1, initial velocity = (4,0), results see fig. 1.3

Rationale: We notice a constant velocity from left to right. Therefore, I set the initial velocity to (4,0). When the hand is occluded it comes in handy to have a constant velocity model that makes sure we keep going to the right even though the hand is currently occluded. Thus I selected model 1. Since the background changes I set alpha = 0.1 such that the target histogram can adapt to the changing background. I set the number of particles to 500. This way the tracking trajectory is more smooth due to the lower variance as explained above. Since the background changes and there is occlusion, I decided to increase the sigma observe to put rather equal weight on the propagation and the observation. This makes it more robust especially when there is occlusion and changing background.

For the following sets of parameters the tracking fails when the hand is occluded:
hist bin=16, alpha = 0.1, sigma observe = 0.8, model = 0, num particles = 200, sigma position = 5, sigma velocity = 1, initial velocity = (4,0), results see fig. 1.4

Now if we only change to the constant velocity model (model 1) the tracking works again. This underlines the reasoning why we should choose the constant velocity model when we face occlusion. Here the params (I just changed the model):
hist bin=16, alpha = 0.1, sigma observe = 0.8, model = 1, num particles = 200, sigma position = 5, sigma velocity = 1, initial velocity = (4,0), results see fig. 1.5

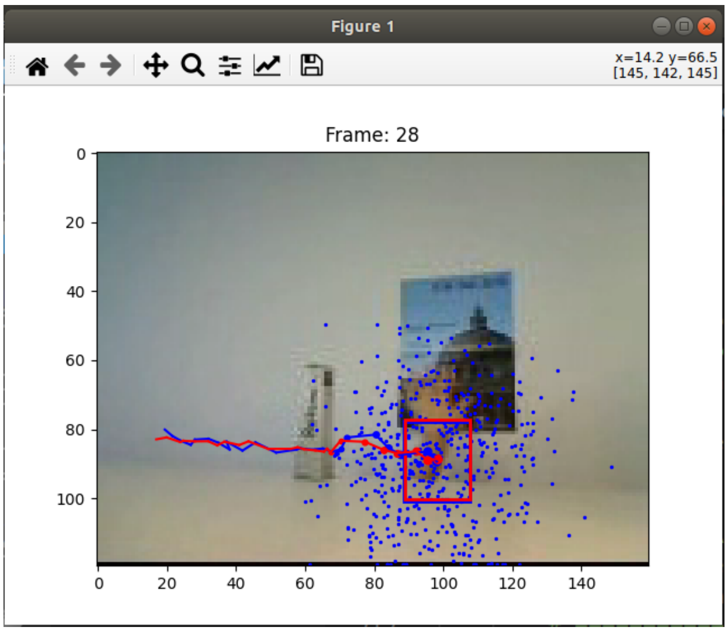

Figure 1.3: Tracking works for video 2

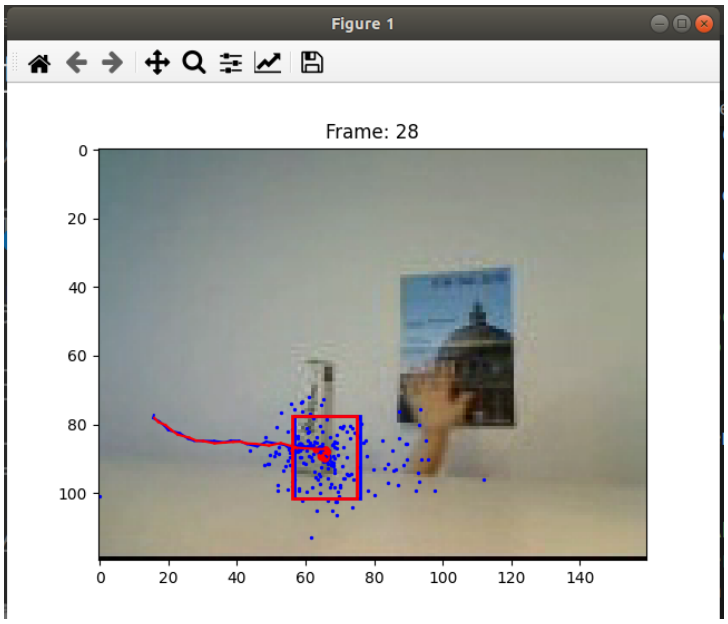

Figure 1.4: Tracking does work NOT for video 2

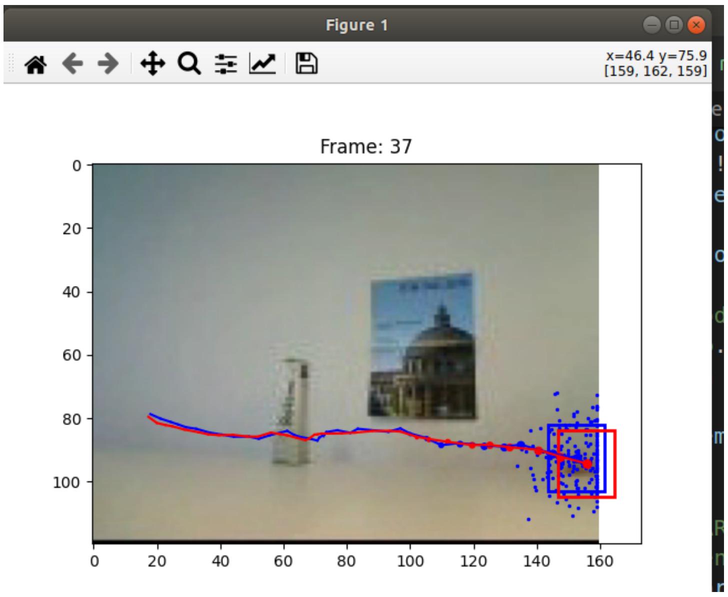

Figure 1.5: Tracking does NOT work for video 2

### 1.2.4 video3

Parameters for video3 for a successful tracking:
hist bin=16, alpha = 0.0, sigma observe = 0.1, model = 0, num particles = 500, sigma position = 15, sigma velocity = 1, initial velocity = (10,0)
Rationale: In this video the ball doesn't change it's appearance and the background is always the same. There is also no occlusion. Thus is decided to trust the observation a lot an decreased the sigma observation and set a higher value for the position noise. A lower sigma observation puts more weight on the measurement/observation and improves the tracking in this video. The challenge here is that the direction of the ball changes when it bumps into the wall. If I use the constant velocity model, the propagation lags behind the ball when it changes direction. If I use the zero velocity model the propagation is better when the ball changes direction. Thus, I used the zero velocity model. However, the estimate is very good in both cases. This is mainly due to: no changing background, no occlusion and the ball doesn't change it's appearance. Thus I also set alpha to zero. There is no need to update the target color histogram as there is no changing background and no change in appearance.

If I set alpha = 0.9 I risk to update the target color histogram with a bad estimate e.g. when the estimate is off in one step then this will become the new target color histogram. This high alpha caused the tracking to fail, see fig. 1.7 The following params were used:
hist bin=16, alpha = 0.9, sigma observe = 0.1, model = 0, num particles = 100, sigma position = 6, sigma velocity = 1, initial velocity = (10,0)

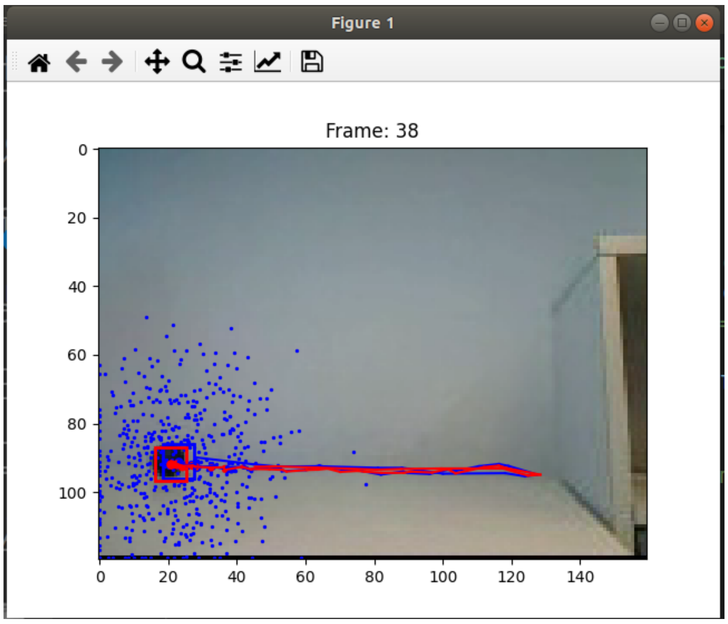

Figure 1.6: Tracking works for video 3

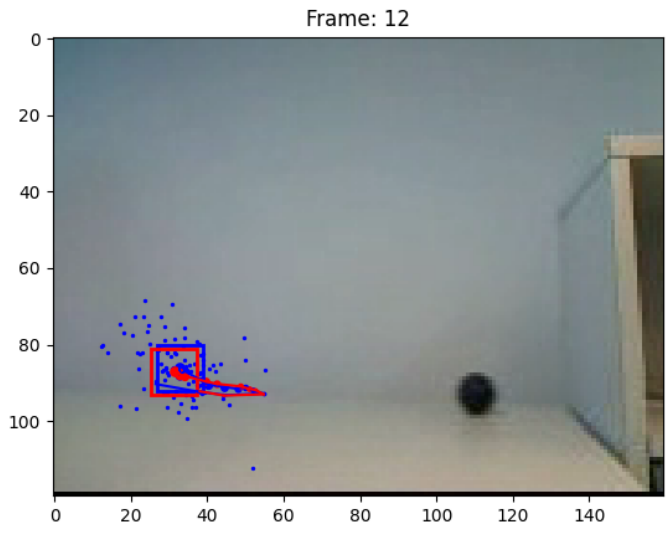

Figure 1.7: Tracking does NOT work for video 3