Face Tracking using a Particle Filter with Color Features and a Self-Updating Window

Alexandra Hotti

Abstract

In this report, an algorithm for face tracking in consecutive video image sequences was successfully implemented and evaluated. Face tracking within videos has a wide area of applications from being used in surveillance systems to video games.

The tracking was achieved by using a particle filter with color features and a self-updating tracking window. Color-based characteristics are vigorous at handling rotational variations, changes in the size of the object being tracked and occlusion. Additionally, the self-updating tracking window is able to handle significant scale changes in the object being tracked.

1. Introduction

In this paper, an algorithm for tracking faces between consecutive image sequences was implemented.

Face tracking over video frames is used within several different computer vision applications such as video games (Tung & Matsuyama, 2008), real-time surveillance (Segen, 1996) and applications for human-computer interaction to recognize facial expressions (Black & Jepson, 1998).

Currently, there are two popular approaches to solving this problem. One approach using the Mean Shift algorithm (Cheng, 1995) and another particle-based solution (Gordon et al., 1995).

In this project a particle filter based on color image features partially based on Nummiaro et al. (2003) was implemented. However, instead of including the window surrounding the face as part of the state, a self-updating tracking window was used in accordance with Wang et al. (2019).

1.1. Problem Statement

The purpose of this project was to successfully implement a face tracking algorithm based on a color model. This algorithm had a self-updating tracking window as described by (Wang et al., 2019).

To qualitatively assess the rigor of the tracking algorithm, it was applied to video sequences under varying image conditions. Such conditions included even and uneven lighting as well as varying background colors. It was also qualitatively evaluated whether the tracking algorithm could withstand short term occlusion as well as a face moving at considerable speed. Lastly, an experiment was conducted to assess the self-updating tracking windows Wang et al. (2019) ability to handle scale changes.

1.2. Contribution

The following contributions were made by this project

- The initial system belief was set using the Viola-Jones Algorithm. Thus, no manual human input was required.
- The object tracking could resume after occlusion, i.e. when the face of a person disappeared and reappeared from the screen.
- The particle filter was capable of tracking the face of a person moving rapidly.
- The tracking algorithm worked despite the video background being somewhat similar in color to the faces being tracked.
- The bounding box enclosing the face would rescale as the person in the video moved closer to or further away from the camera.

1.3. Outline

In section 2 the previous work that forms the basis of this paper is briefly described.

Next in in sections 3.1 - 3.5 the theory behind the algorithm is depicted. The theory is then presented in the order which it was implemented in section 3.6.

In segment 4 the results from four experiments evaluating the rigor of the face tracking algorithm can be found.

Lastly in section 5 a summary, as well as conclusions drawn from the results, are presented.

2. Related work

As mentioned in section 1 this project applies a color-based particle filter with a self-updating tracking window to trace faces between image sequences. This approach is mainly based on three previous works.

First, the Viola-Jones object detection framework proposed by Viola et al. (2001) was used to detect the initial face location. This position was then used as the belief of the particle filter at the first time step. The Viola-Jones framework detects faces using Haar Features which rely on the symmetry properties in human faces.

The tracking was achieved using a particle filter where the object being tracked was believed to be at the weighted mean location of these particles. Observations used to update weights were based on the colors of the areas surrounding the particle, as described by Nummiaro et al. (2003). In order to transition to a new video frame, the particles were propagated. Weights were subsequently updated by measuring the similarities in color between neighborhoods of the propagated particles and the object being tracked. According to Nummiaro et al. (2003), there are numerous benefits of tracing objects based on their coloring. Such as computational efficiency and resilience against occlusion, scale invariance and objects changing pose relative to the camera, i.e. rotation.

However, unlike Nummiaro et al. (2003) the window enclosing the face was not included as part of the state. Thus, instead of propagating the individual particle bounding boxes a self-updating tracking window as described by Wang et al. (2019) was implemented. At each time step, the average distance of all particles to the target center was computed. This average distance was then compared between the current and the previous time step. When the average distance increased the face was presumed to have moved towards the camera and thus the scale of the bounding box increased. According to Wang et al. (2019) this methodology improves the tracking algorithms capability of adjusting the bounding box when the size of the object being tracked substantially changes between consecutive frames.

3. Method

3.1. Particle Filter

The algorithm used to track faces between video frames relies on a particle filter. A face being tracked is represented by a target state vector X_t at a specific time step t. The vector Z_t represents a sequence of observations from video frames up to time t: $\{z_1, z_2, .., z_t\}$. The objective of the particle filter is to estimate the posterior density $p(X_t|Z_t)$ (Nummiaro et al., 2003).

The distribution is estimated using a particle set $S = \{(s^m, \pi^m | m = 1, .., M)\}$ were M is the number of particles. Each particle s^m is a hypothetical state of the face and has an individual sampling probability π^m called a weight. As this is a valid probability distribution the weights are constrained to be non-negative and $\sum_{m=1}^{M} \pi^m = 1$ (Nummiaro et al., 2003).

These hypothetical states can be combined to estimate the current location of the face being tracked. At each time step a weighted mean of the particles x- and y-coordinates are used to compute a mean state (Nummiaro et al., 2003).

$$\mathbb{E}[S] = \sum_{m=1}^{M} \pi^m \cdot s^m \tag{1}$$

3.2. Model

3.2.1. The state representation

As earlier mentioned, each particle at a certain time step represents a hypothetical location of the face. A particle has a location in the image denoted as x,y which represents the center of a rectangular box that encompasses the face being tracked. These center locations also have velocities $\dot{x}$, $\dot{y}$. Depending on the direction of the motion in the image, these velocities could be either positive or negative. To summarize the face being tracked will have four components in its state s

$$s = \{x, y, \dot{x}, \dot{y}\} \tag{2}$$

This state representation could be considered compact. In the state representation used by Nummiaro et al. (2003), additional information included in the state is the bounding box height, width and between state scale change. At each time step these bounding box parameters are updated in the dynamic model see 3.2.2. In this paper, the window is instead updated using a method described by Wang et al. (2019). Which is described in section 3.5.

3.2.2. Dynamic model

A dynamic model is used to predict the target state at the next time step. During the prediction step the location as well as velocity of the particles is propagated using eq. 3.

$$s_t = A \cdot s_{t-1} + G \tag{3}$$

Where G is Gaussian noise and A is a deterministic component given by

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

All in all, the locations and velocities of the particles will be propagated in the following manner

$$\begin{aligned} dx_t &= dx_{t-1} + G \\ dy_t &= dy_{t-1} + G \\ x_t &= x_{t-1} + dx_t + G \\ y_t &= y_{t-1} + dy_t + G \end{aligned} \tag{5}$$

3.3. Likelihood Model

Given that the particles have been propagated their weights need to be recomputed. For each particle, the likelihood of the position of that particle corresponding to the face's location is calculated. These likelihoods are then normalized and used as weights π^m as. See section 3.1.

Conceptually, the likelihoods are obtained by measuring how similar the color distribution of each particle's bounding box is to the mean state color distribution from the previous time step (Nummiaro et al., 2003). This process is outlined in detail below.

3.3.1. Color Histograms

In particle filtering, observations are based on features. In this project, observations correspond to the colors within the bounding boxes of each particle. The color distributions are represented as histograms (Nummiaro et al., 2003).

These color distributions are created in an RGB-space where each color dimension is discretized into 8 bins. For computational efficiency each RGB-dimension is treated independently. Such that a single particle obtains $8 \cdot 3 = 24$ bins instead of $8^3 = 512$ bins.

A pixel located near the edge of a bounding box run larger risks of being part of the background or becoming occluded. Therefore, the histograms should be to weighted such that the colors of pixels close to the center get larger weights compared to boundary. This is achieved by the following weighting function (Nummiaro et al., 2003)

$$k(r) = \begin{cases} 1 - r^2 & r < 1 \\ 0 & otherwise \end{cases} \tag{6}$$

here r is the distance from a pixel x_i to that particular particles bounding box center (Nummiaro et al., 2003).

Now all information needed to define the color histogram has been outlined. The color distribution for a particular particle is built by filling all of the $8x8x8$ bins in the RGB space. To fill one bin b for one particle m with center location $m_{x,y}$ the following formula is used

$$p_m^{(b)} = f \sum_{i=1}^{I} k(\frac{||m_{x,y} - x_i||}{a}) \delta(h(x_i) - b) \tag{7}$$

where I is the number of pixels in the bounding box, x_i represents a specific pixel and $||m_{x,y} - x_i||$ is the distance between the bounding box center and the pixel. The function $h(x_i)$ assigns the color at the pixel location x_i to its corresponding color bin. When either of the RGB-components of the pixel matches the current bin b the dirac function will be equal to 1. The following equation was used for setting the a parameter $a = \sqrt{H_x^2 + H_y^2}$. The a parameter adapts the region size such that an equivalent weighting is achieved irrespective of the size of the bounding box. Meaning that the weighting is unaffected by how close the face is to the camera. Lastly, f is a normalization constant.

3.3.2. Distribution similarities - The Bhattacharyya distance

The individual particle color histograms are then compared against the mean state color distribution from the previous time step. The mean state is presumed to be covering the face. Thereby, the measurement estimates how similar the coloring in the neighborhoods of the propagated particles are to the face being tracked (Nummiaro et al., 2003).

The comparison is performed using the Bhattacharyya distance d, which measures the similarity between two probability distributions $p(b)$ and $q(b)$ (Nummiaro et al., 2003). First, the Bhattacharyya coefficient ρ is calculated

$$\rho[p, q] = \sum_{b=1}^{\#bins} \sqrt{p^{(b)} \cdot q^{(b)}} \tag{8}$$

Here p is a particle color distribution and q is the color distribution of the mean state from the previous time step. The coefficient takes a maximum value of $\rho = 1$. The maximum indicates that the two distributions are identical (Nummiaro et al., 2003).

The coefficient is then used to calculate the Bhattacharyya distance between the distributions

$$d = \sqrt{1 - \rho[p, q]} \tag{9}$$

3.3.3. The face color likelihoods

The Bhattacharyya distances computed in 3.3.2 are then used to update the particle weights. A smaller distance entails a coloring closer to the mean state and should thereby give a larger weight. A large weight increases a particles chance of being resampled, see step 5 in 3.6. Some particles will have several decedents while some will not be resampled and thus has a lineage that dies out.

For every particle m a weight is computed using the following function

$$\pi^{(m)} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d^2}{2\sigma^2}} \tag{10}$$

Note that the weights are normalized such that $\sum_m^M \pi^{(m)} = 1$. Here, d is the Bhattacharyya distance defined in eq. 9. Larger values for d entail a closer similarity to the previous mean state coloring. Thus, a large d value will result in a large weight $\pi^{(m)}$ (Nummiaro et al., 2003).

The σ value specifies how much weight is given to a particle in relation to their d value. Thus, as σ increases greater weights are given to particles with larger distances. Meaning that the algorithm becomes less stringent about how similar a particle's surrounding coloring is to the mean state. Consequently, the value of σ could be increased to handle changes in lighting conditions. However, this could potentially result in particle degeneracy as areas of the surrounding imagery would also achieve higher likelihoods. If σ instead is to small the particles will not spread out around the face and as the target moves the particles could lose track of it.

3.4. Mean state update

Once the weights have been updated, they are used to estimate the current face state. This is done by using eq. 1. Such that the center of the face is estimated as the weighted mean of the particles' locations (Nummiaro et al., 2003).

In addition to the face location, the color distribution of the mean state is updated. This distribution is referred to as q_t and is used in eq. 8 to compute the Bhattacharyya distance d. Thereby, as lighting conditions and the pose of the target varies throughout the video q_t is adjusted accordingly. The updated is performed using eq. 11 (Nummiaro et al., 2003)

$$q_t = (1 - \alpha)q_{t-1} + \alpha p_{mean} \tag{11}$$

here α specifies how much to incorporate of the new reference state color distribution into q_t and how much we want to keep of historic mean state color histograms. Which means that the influence of earlier states decreases as the tracking evolves (Nummiaro et al., 2003).

The effects of outliers on the mean state estimate is reduced by including a threshold on the observation likelihoods. Occasionally a video frame could contain a large amount of noise or the target might disappear from screen. This would result in a mean state p_{mean} with a low likelihood. Thus, a current state is only included in the update of q_t if it is larger than a predetermined threshold (Nummiaro et al., 2003).

3.5. Self updating Bounding Box

A method described by Wang et al. (2019) is used for updating the size of the bounding box surrounding the face.

$$\begin{cases} w_t = w_{t-1} \cdot \frac{d}{d_l} \\ h_t = h_{t-1} \cdot \frac{d}{d_l} \end{cases} \tag{12}$$

here d is the average distance between the particles and the mean state center

$$d = \frac{1}{M} \sum_{i=1}^{M} \sqrt{(x - x_i)^2 + (y - y_i)^2} \tag{13}$$

d_l is the average distance from the previous frame to the mean state center. Meaning that the window width and height is scaled by how much the average distance to the target center has changed from the previous to the current video frame (Wang et al., 2019).

3.6. The Tracking Algorithm

Here the steps in the tracking algorithm are presented in the order in which they were implemented.

The algorithm is run for each frame t in a sequence of frames in a video.

step 0. Initialization , $t = 0$

0.1. At $t = 0$, the system is initialized. First the parameters of the initial state $x_0, y_0, \dot{x}, \dot{y}$ are set. The initial velocities: $\dot{x}, \dot{y}$ are set to 0 and the center of the initial face bounding box: x_0, y_0, is found via the Viola-Jones algorithm (Viola et al., 2001).

0.2. The Viola-Jones algorithm also returns the initial width and height of the bounding box surrounding the face at the first time step.

0.3. Next, M particles are initialized uniformly within the bounding box found in the previous step. The particles are given zero velocities and uniform weights of $1/M$.

0.4. Equation 12 requires an average distance between the particles and the bounding box center for the current d_t and the previous d_{t-1} time steps. d_0 is initialized by calculating the average distance between the center found by the Viola-Jones algorithm and the initialized particles.

0.5. The color histogram of the image area within the initial face bounding box is used as the initial target q_0. It is estimated using eq. 7.

step 1. Updating the bounding box, $t \geq 1$

1.1. The bounding box at time step t is predicted using eq. 12. Where the average distance between the particles and the bounding box center d_t is calculated via eq. 13.

step 2. Propagation, $t \geq 1$

2.1. The particles are propagated using to eq. 3 and 5.

step 3. Weight update, $t \geq 1$

3.1 A color histogram is estimated for each particle based on the area within each particle's bounding box. The particle color histograms are calculated using eq. 7.

3.2 **The Bhattacharyya coefficient:** Next, the Bhattacharyya coefficient is calculated for each particle using the color histograms and the target state q's color histogram, see eq. 8. The Bhattacharyya distance is calculated based on this coefficient using eq. 9.

3.3 **Calculate the Particle Weights:** The particle weights are calculated using eq. 10 and the Bhattacharyya distances. This results in observation y_c likelihood $p(y_c|x)$ for each particle x. The likelihoods are then normalized into weights.

step 4. Estimate target state: The current state is now estimated using eq. 1.

step 5. Systematic Resampling: The particles are resampled based on the updated weights. The weight of a resampled particle is set to $1/M$.

step 6. Update histogram of target state: The histogram of the bounding box surrounding the current mean state is calculated using eq. 7. This histogram is used as the reference distribution at the next time step q_{t+1}.

3.7. Implementation

All of the code was implemented in Matlab.

The initial face location which was found with the Viola-Jones algorithm was implemented with a Matlab package called $vision.CascadeObjectDetector$ (Mathworks, 2012).

4. Experimental results

As mentioned in the problem statement, to qualitatively asses the tracking algorithm it has applied it to several video sequences with varying conditions. First, the videos contain varying lighting conditions and different background colors. An experiment was performed to assess whether the tracking algorithm would continue running after short term occlusion as well as a face moving with considerable speed. Lastly, the self-updating tracking windows abilities to handle changes in scale was evaluated.

In the results in Figure 1-5 the small scattered blue and red circles are the particles at the current time step. The green boxes are the estimated bounding boxes of the current state.

Figure 1. Here are the results from tracking a person's face when he moves quickly to the left. An extra element of difficulty in this sequence is the red background which bears a resemblance to the faces coloring. The top image is the first image in a sequence and the bottom image is the last. When he is standing still in the first image the bounding box tightly encloses his face. When he quickly starts moving in the second image the size of the bounding box increases. As he slows down in the last image the bounding box shrinks around his face.

4.1. Red Background with Fast Movement and Even Lighting

The tracking algorithm was applied to a video where a person moves quickly to the left. See Figure 1. The difficulty of tracking is greater for the fast compared to the slow movements. An extra element of difficulty in this sequence is the red background which bears a resemblance to the faces coloring compared to the results in for instance section 4.2. The tracking procedure is potentially abbreviated by the lighting, which is even throughout the video compared to the results in 4.2.

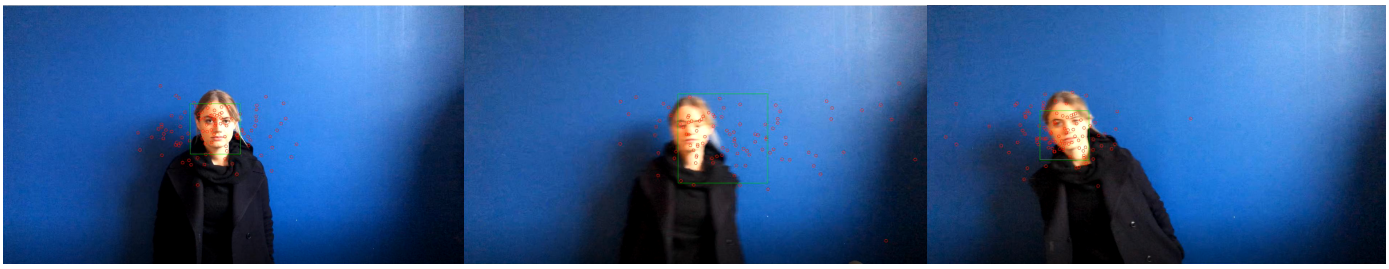

Figure 2. Here are the results from tracking a person's face when she moves quickly to the left. An added complexity in this sequence is lighting conditions which vary across the screen. In the first image from the left, the person stands still and a tight bounding box is obtained. In the next image, the person moves rapidly to the left and the bounding box has difficulties with covering the face. Also, the rapid motion results in a bounding box considerably larger than the person's face. In the last image, the person is moving at a lesser speed and the algorithm produces a tight window around the face.

When the person stands still the algorithm tightly encloses the person's face within a bounding box. See the first image in Figure 1. As, the person swiftly shift to the left the face maintains within the bounding box. However, there appears to be a trade off between keeping the bounding box on the target and having a tight bounding box. A potential explanation is related to the parameter setting of the dynamic model. To model fast motion, large noise needs to be sampled in the propagation of the particles in eq.5. If fast motion is allowed in one direction it is simultaneously permitted in the opposite direction. Thus, to sustain the face within the bounding box when the person moves to the left, the particles can also move swiftly to the right. Then how the particles moving to the left are handled depends on the parameter setting in the likelihood computation in eq. 10. If the σ parameter is set conservatively these particles would receive small weights. However, this would potentially increase the difficulty of tracking in a room where the luminance varies. In this particular image the particles are allowed to move swiftly and thus the face stays within the box. However, the size of the window is considerably large compared to the face.

4.2. Blue Background with Fast Movement and Uneven Lighting

The particle filter was then applied to another video sequence where the face is moving fast at times. However, this background bears less of a resemblance to the face coloring compared to section 4.1. Also, the luminance changes in different parts of the frame. See Figure 2.

In the initial frame to the left, the window closely encompasses the face. As the person swiftly shifts in the middle image the bounding box is enlarged. These issues are similar to those identified in section 4.1. In the last image, the person is moving at a lesser speed and the algorithm produces a tight window around the face.

An added complexity in this sequence is lighting conditions which vary across the screen. Meaning that the color distribution of the face changes as the person moves around. This issue could perchance be reduced by increasing the α value in eq. 11. This would increase the weight given to newer frames during the update of the target reference color histogram. Meaning that local lighting conditions would have a greater significance.

4.3. Occlusion

The next experiment centered on occlusion. It was evaluated whether the tracking would continue after the person disappeared and reappeared from view. The results for this section can be found in Figure 3 and 4.

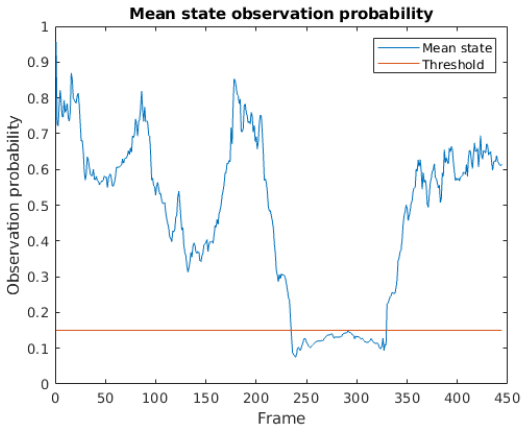

Figure 3. The graph shows the evolution of the observation likelihood of the estimated mean state throughout the video sequence in Figure 4. Around frame 200 the man disappears from screen and the observation likelihood plunges. As the man reenters into the screen around frame 350 the likelihood swiftly increases. The red line shows the likelihood threshold used to decide when not to update the target reference state q.

Figure 4. Here the particle filter's ability to handle short periods of occlusion is evaluated. In the first image at the top, the person is moving towards the image boundary. In the next image, his face is completely out of sight. As he reapers in the last image the bounding box centers around his face.

In Figure 4 the particle filter's ability to handle short periods of occlusion is evaluated. In the fast image at the top, the person is moving towards the image boundary. In the second image, his face is completely out of sight and the particles move towards the light blue regions of the background. Possibly since these lighter regions bear more resemblance to the paleness of his face.

In Figure 3 the effects occlusion has on the observation likelihood is assessed. The graph shows the evolution of the observation likelihood of the estimated mean state throughout the video sequence in Figure 4. Around frame 200 the man disappears from screen and the observation likelihood plunges. As the man reenters into the screen around frame 350 the likelihood swiftly increases. The red line shows the likelihood threshold used to decide when not to update the target reference state q. See section 11. Thus, the target reference histogram should still be representative of his facial colors. When the man returns some particles still populate regions close to where he exited. These particles likely receive large weights in relation to the other particles and thus are resampled the more frequently. This is consistent with the bounding box quickly centering around the man's face as he enters the image frame.

4.4. Self-Updating Tracking Window

The last experiment focused on examining the self-updating tracking window's capabilities of handling changes in scale. These results can be found in Figure 5. The image sequence begins with the top image and finishes with the image at the bottom. Meaning that the girl steps towards the camera and then moves back towards the wall.

The bounding box has been enlarged in the second and the third images. Yet, the window is unable to completely encompass the face in the third image. Perhaps this is caused by an increase in saturation in the third image. Such effects could be mitigated by increasing the α parameter in eq. 11 as described in section 4.2.

In the fourth and fifth images from the top, the person is moving towards the wall. As she moves backward the size of the bounding box shrinks. However, the face is not entirely enclosed until the saturation has significantly decreased in the fifth image.

5. Summary and Conclusions

The four experiments investigated the particle filter's proficiency in handling different image conditions. First, the filter was capable of tracking the face when the background coloring was similar and dissimilar to the face coloring.

The tracking algorithm could stay on target during rapid movements. However, this required an increased amount of additive noise during the particle propagation. Consequently, resulting in an enlarged bounding box.

The particle filter was able to handle short periods of occlusion. Also, the bounding box was effectively resized as the target scale changed.

However, uneven lighting conditions seemed to negatively impact the bounding box's ability to encircle the target. Enclosing the face entirely within the bounding box appeared easier when the lighting was even throughout the screen. Thus, future experiments should focus on mitigating the effects of variable brightness.

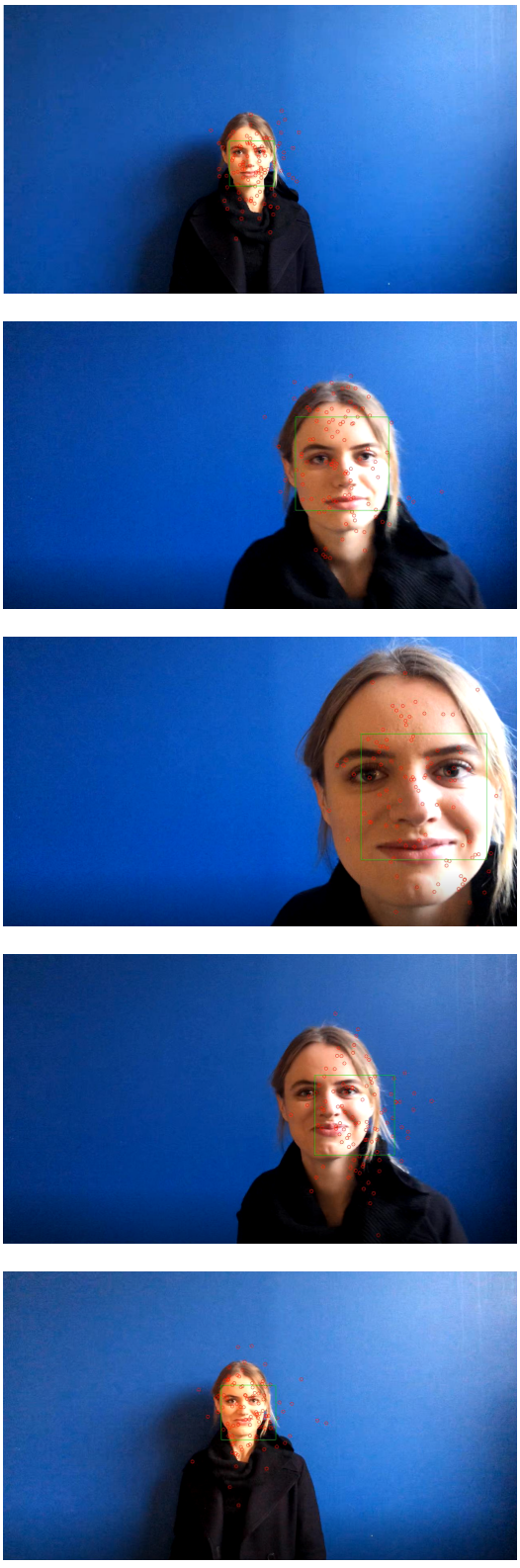

Figure 5. Here the self-updating tracking window's capabilities of handling changes in scale is examined. The image sequence begins with the top image and finishes with the image at the bottom. Meaning that the girl steps towards the camera and then moves back towards the wall.

References

Black, M. and Jepson, A. A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. *ECCV*, 11 1998. doi: 10.1007/BFb0055712.

Cheng, Y. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

Gordon, N., Salmond, D., and Ewing, C. Bayesian state estimation for tracking and guidance using the bootstrap filter. *Journal of Guidance, Control, and Dynamics*, 18 (6):1434–1443, 1995.

Mathworks. vision.cascadeobjectdetector. https://se.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html, 2012. Accessed: 2019-12-18.

Nummiaro, K., Koller-Meier, E., and Van Gool, L. An adaptive color-based particle filter. *Image and vision computing*, 21(1):99–110, 2003.

Segen, J. A camera-based system for tracking people in real time. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 3, pp. 63–67. IEEE, 1996.

Tung, T. and Matsuyama, T. Human motion tracking using a color-based particle filter driven by optical flow. 2008.

Viola, P., Jones, M., et al. Robust real-time object detection. *International journal of computer vision*, 4(34-47): 4, 2001.

Wang, T., Wang, W., Liu, H., and Li, T. Research on a face real-time tracking algorithm based on particle filter multi-feature fusion. *Sensors*, 19(5):1245, 2019.