# Tracking ball in video with histogram colour-based particle filter

Badi Mirzai, badim@kth.se
Lab partners: Kristiyan Lazarov, klazarov@th.se
Valentine Lin, velelin@kth.se

January 5, 2021

**Abstract**

This paper implemented a colour based particle filter for object tracking of a ball moving between a sequence of frames in a video. The particle filter focused on RGB colour intensity histograms, where particles were analyzed as pixels and were compared to the reference histogram of the ball which illustrated the distribution of colour intensities among the pixels. A multinominal resampling method was implemented as well as particle weighting based similarities between the reference histogram and the particles colours. The results were sufficient and limited, but fulfilled its objective in demonstrated the properties of particle filter by tracking the movement of the ball.

## 1 Introduction

Object tracking consists of identifying the desired object and also localizing its position. Tracking an object in a video basically focuses on analyzing the video frame by frame with a model that determines where the object might be for each frame with a certainty. Object tracking have become crucial in many fields with the emerge of the digital age, which would include (but not limited to) facial recognition, autonomous vehicles, and artificial intelligence.

With the increased computer processing power and available data during recent years, Machine Learning models such as Convolutional Neural Networks (CNN) have become more popular during the recent years. One example is [1], which is the first end-to end learning method based on Deep Learning for object tracking. It uses a Quad-CNN that determines the similarities between detected objects by taking into consideration their labels as well as their time stamps for each frame. The paper [1] also use bounding-box regression to help train the network, which they achieve by minimizing a combination of the generated loss function between the weighted bounding-box and the data association. However, due to the scope to the course in which this project is being executed in, this paper attempts to focus more on traditional mathematical methods and colour histograms. More precisely, analyzing the colours of the picture to compare them with the reference object's colour distribution among its pixels. There are multiple ways to object tracking with colour based models. For example [2] uses a improved model of mean shift segmentation, where they smooth the color density (kernel) function between frames which flattens the cost function and avoids local local optimization points rather than global ones. (Mean shift is essentially clustering the image into different regions,

based on similarities between neighbouring colours, which can be used as a way to track certain objects). Their method resulted in slower changes of the number and positions of optimization modes when trying to cluster the object in colour spatial domains and this made it easier to track objects between frames.

This report was not based on any specific previous work. However, inspiration was give by [3], where they combined colour based histogram (Bhattacharyya distances for similaritied between colours) with a moment based model (taking into consideration spatial distances), merging colour with distance when weighting the particles in a particle filter for facial recognition. Our approach does not however include the spatial distance, since the objective was to test the simplicity in only colour based histograms. A more similar work to ours can be found in [4], where a colour based histogram particle filter was implemented for tracking multiple objects in video. They achieved this by extracting colour histograms in the particles state region for each video frame and compared it with the reference frame. Unlike our approach, they used a Gaussian density distribution as likelihood which essentially determined the similarity between the reference histogram of object to be tracked and the computed histogram for specific regions in each frame. They also used regions instead of pixels for calculating the particle histograms as well as the Bhattachary distace for measuring colour histogram similarities as distances.

The method used in this paper consists of a non-parametric method for tracking a ball through a particle filter, where we analyze the colour of the video and have the particle filter make educated guess of the ball's position for each frame that is based on a specified reference of the object. The choice of only analyzing the colour distribution of the frames were made due to test the simplicity of the particle filter, which meant that the object had to have a distinct colour compared to the rest of the frame. Hence, this paper did not take into consideration spatial domain.

# 2 Contribution

The main contribution of this paper was:

- Obtaining sufficient results of object tracking of a red ball which was achieved through only colour based particle filter modeling.
- Managed to use a very simple prediction model that still produced sufficient results and demonstrated how powerful particle filters can be.
- Was able to track the position of the ball in pixel-particles to a certain degree without the need of moment-based models like previous work [3] in the same course.
- Estimated the position of the ball by taking the average of the 20 particles with the highest weights as measured model.

# 3 Implementation

## 3.1 Colour based histogram

A video can be split up into a sequence of frames, where each frame contains pixels that have a position as well as a RGB (Red, Green and Blue) colours.This means that all colours can be

represented in these colours. Each pixel will thus have 3 values for the respective colours. Histograms can therefor be used as a graphical representations of the different colour in a regions of pixels. An example can be to analyze the colour histograms of a specific object in a frame as in Figure 2, were its correspinding colour histogram in RGB colors can be represented in Figure 1. A histogram is a graphical representation which displays the distirbution of intensity bins. Analyzing Fig 1, it can be seen that there are indeed more pixels that have high intensity in the colour red than there are in the blue or green colour. This makes sense, since the ball is indeed red, which will be the key feature property when tracking the ball.

A histogram is a graphical display of data using bars of different heights. In a histogram, each bar groups numbers into ranges. Taller bars show that more data falls in that range. A histogram displays the shape and spread of continuous sample data.

**Algorithm 1** Color reference histograms

$reference.frame = frame(y0 : y1; x0 : x1)$
$reference.histograms = matlab.histcounts.N$
$reference.edges = matlab.histcounts.edges$

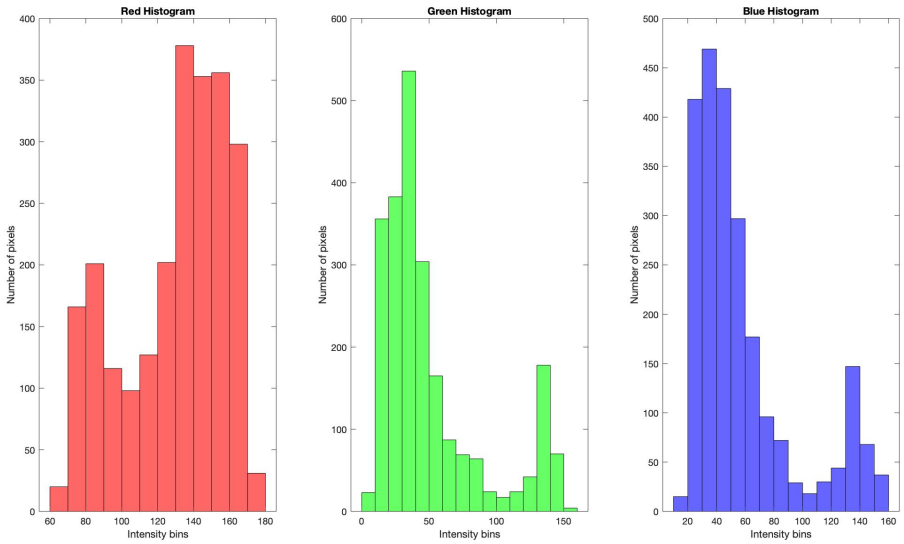



Figure 1: Histogram of the ball in Fig 2 as reference in RGB colours.

Figure 2: Ball used as reference by specifying certain pixel region in an entire frame.

**Algorithm 2** Color reference histograms

$reference.frame = frame(y0 : y1; x0 : x1)$
$reference.histograms = matlab.histcounts.N$
$reference.edges = matlab.histcounts.edges$

Algorithm 2 explains briefly on how to extract the histogram in Figure 1 with build-in tools in programming languages such as MATLAB. The colours are split for each RGB colour and the histograms display the distribution of their intensities among the pixels.

## 3.2 Prediction model

The prediction model for modeling how the object moves were chosen in a very simple way which in this case is described in Algorithm 3 and also further in Algorithm 7 later in the implementation.

**Algorithm 3** Prediction model for particle filter

**Input:** previous.positions, current.positions
**return** $current.positions - previous.positions$

In words, Algorithm 3 basically returns a new predicted control action object for the next frame which in this case is the difference between the current position and the previous position. This prediction is made as an a priori which is further used in Algorithm 7 to predict the position for the object in the next state (in this case the next video frame). The model is only in spatial movement and does not take into account neither colour or some covariaince of the particles movement in randomness, but was later used (although Algorithm 7 does take into account covariance for random movement of the particles).

## 3.3 Colour based observation model

## 3.4 Particle filter

The particle filter is a non parametric version of the Bayes filter, which approximates the posterior distribution with a finite number of hypothesise known as particles. This is done by representing the posterior belief $bel(x_t)$ by sampling with a set of random states drawn from this posterior distribution. The effect is that the distribution is represented by these particles drawn from the posterior belief, rather than having a parametric distribution. The particles are denoted $S_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}$. Each particle $x_t^{[m]} (1 \leq m \leq M)$ is a hypothesis of the true value of the state for the time $t$. $M$ is the total number of particles (hypothesises) of the particle set $S_t$ [5].

I words, it can be said that the general algorithm for particle filter in Algorithm 4 is set up in a prediction step, weighting and a re-sampling step. However, the algorithm used in this paper is slightly different, as we will see later in the report.

The objective of the particle filter is to approximate the belief $bel(x_t)$ with the particle set $S_t$. The likelihood (prediction step in Algorithm 4) of the particle set is proportional to its Byes posterior $bel(x_t)$:

$$x_t^{[m]} \propto p(x_t | z_{1:t}, u_{1:t}) \quad [3.1]$$

**Algorithm 4** Generic Particle Filter

**Input:** Particle set $S_{t-1} = < x_{t-1}^i, w_{t-1}^i > | i - 1, ..., N$
Control signal $u_t$ $\hat{S}_t = S_T = \emptyset$
Initialize reference frame of the object's histogram.
**for** m=1 to M **do**
sample $x_t^m \sim p(x_t|u_t, x^m t - 1)$ {Prediction}
**end for**
**for** m=1 to M **do**
$w_t^m = p(z_t|x_t^m)$ {Weighting}
**end for**
$\hat{S}t = \bigcup m = 1^M < x_t^m, w_t^m >$
**for** m=1 to M **do**
draw $i \propto w_t^t$ {Re-sampling}
$S_t = S_t \cup < x_t^i, \frac{1}{M}$
**end for**
**return** $S_t$

Table 1: Brief explanation of particle filter used in this report

1. Initialize the reference frame for the object with corresponding histogram.
2. Initialize M particles across the frame randomly, with equal weights.
3. Calculate weights with accordance to the reference histogram of the object.
4. Select 20 particels with highest weights
5. Initialize particle set of M particles with weights.
6. Calculate cumulative weights of M particle: $cdf(m) = \sum_{i=0}^{m} w_i$, for $m \in [0, M]$.
7. For particle m = 1, 2, ..., M
   (a) Generate random number $r_m \in [0, 1]$.

(b) Find the first particle whose cumulative weight is above r and place it into the new particle set.

8. Reset all weight for new particles uniformly to $1/M$

Here in equation 3.1, the particles are predicted by sampling proportional from the distribution of being

For a finite $M$, the particles are drawn from slightly different distribution, but the difference can be neglected for large values of $M$. The belief $bel(x_t)$ is recursively drawn one time step earlier from $bel(x_{t-1})$ and thus are the set of particles $S_t$ drawn from $S_{t-1}$.

Each particle also have individual weights, which represents how strong the particle is likely to be correct. If the weights are normalized for all particles in its set $S_t$, then it will represent the probability of each particle of being correct.

The particle filter implemented in this experiment is similar but slightly different from the Generic particicle filter shown in Algorithm 4.

### 3.4.1 Weighting

The are many ways in how to assign weights to the particles. In this report, the particles were assigned weights according to Algorithm 5 which looked at the pixel colour histogram of each color (RBG) for each particle and compared it with the reference histogram of the object for the corresponding colour. The choice was made due to its simplicity but relevance, since each

particle was a pixel, which was being compared with a colour histogram of a bounding box. It can bee noticed from Algorithm 5 that the particle were comparing its intensity with which bins it belonged to the corresponding one in the reference histogram. The reference histogram had to be normalized in order to generate a probability to the particles' weights.

**Algorithm 5** Weights calculation

**Input:** particles, reference.histograms, reference.edges
  **for** m=1 to M **do** Loop all particles
    **for** k=2 to length(reference.edges) **do** Loop all reference edges
      **if** $particles_i \leq reference.edges_k$ **then**
        $probability_i = reference.histograms_k$
        break
      **end if**
    **end for**
  **end for**
  **return** $probability_i$

### 3.4.2 Resampling

The resampling of particles is important since it prevents particle deprevation, making sure that the most reliable particles are sampled with greater probability. This increases the partcile weights where there is a larger probability of them being there and decreasing the particle weights where it matters less. The resamping method used in this report multinomial resampling, with the pseudo algorithm shown in Algorithm 6. Other resampling methods, such as Vanilla (Systematic) resampling could also have been chose. However the Multinomial resampling proved to be sufficient and with a larger variance. This is particularly preferable when the observation model and movement model are not to be trusted as much, which can be argued were the case in the simple prediction mdoel used in this report.

The algorithms for the multinomial resampling and the generic particle filter explained in Algorithms 4 and 6 are complete. However, they are also explained in further detail with words in Table 2 and Table 1.

**Algorithm 6** General Multinomial Re-Sampling algorithm

  $St = \emptyset$
  **for** m = 1 to M **do**
    $CFD(m) = \sum_{i=1}^{m} w_t^i$
  **end for**
  **for** m = 1 to M **do**
    $r_m = rand\{0 \leq r_m \leq 1\}$
    $i = argmin_j CDF(j) \geq r_m$
    $S_t = S_t \cup < x_t^i, \frac{1}{M} >$
  **end for**
  **return** $S_t$

Table 2: Brief explanation of Multinomial Re-Sampling used in this report

1. Initialize particle set of M particles with weights.
2. Calculate cumulative weights of M particle: $cdf(m) = \sum_{i=0}^{m} w_i$, for $m \in [0, M]$.
3. For particle m = 1, 2, ..., M
   (a) Generate random number $r_m \in [0, 1]$.
(b) Find the first particle whose cumulative weight is above r and place it into the new particle set.
4. Reset all weight for new particles uniformly to $1/M$.

# 4 Experiments

## 4.1 Implementation

The object that was specified in the first frame of the video and its corresponding histogram according to Algorithm 2, where the pixels of the ball used as reference can be seen in Figure 2 with its corresponding RGB histogram in Figure 1. For specifics, the ball was chosen as the pixels between 835-880 and 325-375 for its x and y pixel position respectively of the first frame. Each frame had $1050x750$ pixels, where each pixel also had 3 values for their RGB colours. The reference frame's histogram was also normalized to represent probability with MATLABs histcount function. This was done to be able to compare the RGB values for the reference histogram with each particle's to generate a probability (weight) for each particle, which was done through Algorithm 5. Then the 20 particles with the highest weights (i.e corresponding to the most similar with respect to the reference frame's histogram) were used for calculating their average positions to estimate the position of the ball. This was done for each frame, where the estimated position would be drawn out by changing the pixel value at that particle in the current frame with a green value and no red or blue intensity.

There were 10000 particles used, where each had a pixel value for their spatial position, their own weights and their corresponding histograms. The particles were stored in a variable "particles" that was a $Mx5$ matrix ($M = 10000$ in this case), where the second dimension represented position and RGB values (R,G,B,x,y). For the first frame in the video, the particles were initialized by randomization in the spatial domain before their corresponding weights were calculated with Algorithm 5, which resulted in a probability for each particle being the correct one based on their histograms compared to the reference histograms.

**Algorithm 7** move_particles

**Input:** particles, movement
$covar$ {random $1 \cdot length(particles)$ array from normal distribution, mean=1, variance = 0.2 }
$particles.position = particles.position + movement \cdot covar$
**return** $particles$

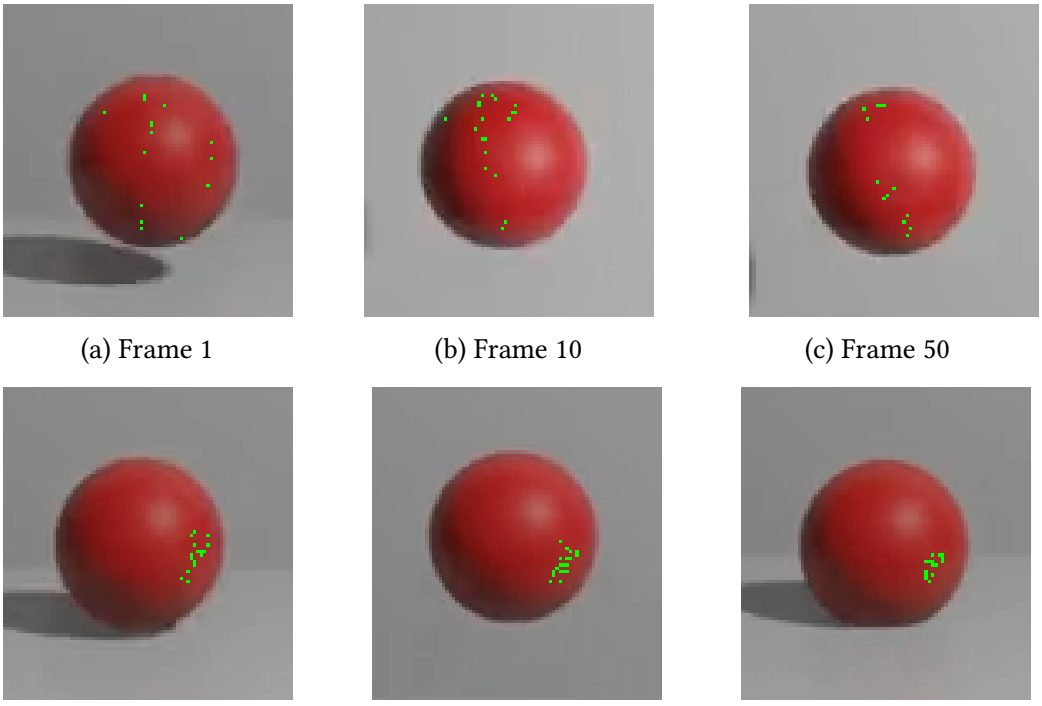
It can be seen from Algorithm 8 that it is slightly different from the generic algorithm shown in Algorithm 4. For example, one difference is that the resampling step is executed first instead of last. However, this is not necessarily the case, since Algorithm 8 shows the sequence after the first frame have been executed as described above, which means that the resampling comes last in the first frame.

**Algorithm 8** Implemented particle filter sequence frames

Initialize states, calculate weights
**while** HasFrame **do**
  $frame = readframe(Video)$
  $weights = Resampling(particles, weights, frame)$ {Algorithm 6}
  $movement = prediction(current\_position, previous\_position)$ {Algorithm 3 }
  $particles = move\_particles(particles, movement)$ {Algorithm 7}
  $old\_position = new\_position$
  **for** m = 1 to M **do**
    Update RGB values for the $m^{th}$ particle's pixel position
  **end for**
  $weights = calc\_weights(particles, ref\_hist, ref\_edges)$ {Algorithm 5}
  $[\sim, indices] = maxk(weights, 20)$ {indices for 20 largest particle weights}
  $new\_particles = particles(indices, 4 : 5)$ {pixel positions for 20 largest particle weights}
  **for** all $new\_particles$ **do**
    Draw green pixel at position in frame of each new_particle
    current_position = current_position + new_particles.position {Update positions}
  **end for**
  current_position = current_position./length(new_particle) {Average position of ball}
**end while**

## 4.2 Results

The results of some of the frames in the video can be seen below in Fig 3, where it can be seen that the 20 best particles are marked green and are able to track the ball throughout the video. The entire video demonstrating the tracking of the particle filter is attached in the assignment of this report, where the best particles in all 179 frames are able to track the red ball bouncing. It is worth noting that the each particle is a pixel but the reference frame is a bounding box around the ball. This means that the ball will have included some other background colours and shadowed parts, which causes the histogram in Figure 1 to be effected slightly. However the results proved that it was a sufficient comparison, due to the discrete red colouring of the ball compared to the grey background colour.



(a) Frame 1 (b) Frame 10 (c) Frame 50

(d) Frame 100 (e) Frame 150 (f) Frame 179 (final frame)

Figure 3: Results of particle filter in different frames with the 20 particles with highest weights (green pixels).

# 5 Conclusion

The results showed that it was indeed possible to implement a colour based particle filter for sufficient object tracking. The implementation is not solid in it's performance, since it can give different result depending on the computer's processor. It might therefore be good to have in mind that the implementation needs to be expanded, with for example spatial distances as in [3].

This report mainly demonstrates an simple implementation of particle filter for object tracking, which only relied on color-based observation models. The results showed that it was sufficient, but is far away from perfect and needs improvements since it does not take into consideration spatial distances.

The particles were also clustered at the later frames in the video, which might not be a good solution, even though they they do converge to the correct part. The colour of the ball might also change due to change in lighting and angles which results the colour histogram of the ball. The reference histogram also consisted parts of the background, which might have had an impact on the results.

This works gave hands on experience on particle filters and colour histograms which showed the authors how powerful yet easy particle filters can be, despite the hype around other Machine Learning methods.

# 6 Future work

Future work could include using bounding boxes for tracing the ball, which would result in being able to compare the histogram of each bounding box (particle) directly with the reference histogram of the object. Another future work could also include spacial filtering, where one would decrease the particle weights for particles that are spatially further away from the previous measured position of the object. It would also have been better to use Gaussian density distribution models for the likelihood of the particles for determining the similarities between the frames (like mentioned in the work [4]) and perhaps look more than one frame into the past when calculating similarities between frames. This would also include representing the particles as regions of pixels, instead of single pixels like we did in this paper.

# References


[1] J. Son, M. Baek, M. Cho, and B. Han, "Multi-object tracking with quadruplet convolutional neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3786–3795.

[2] A. Dargazany, A. Soleimani, and A. Ahmadyfard, "Multibandwidth kernel-based object tracking." *Advances in Artificial Intelligence (16877470)*, 2010.

[3] T. Zhao and F. Liu, "Face tracking in video using an integrated color-based and moment-based particle filter."

[4] M. Jaward, L. Mihaylova, N. Canagarajah, and D. Bull, "Multiple object tracking using particle filters," in *2006 IEEE Aerospace Conference*, 2006, pp. 8 pp.–.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* Reading, Massachusetts: The MIT Press, 2005.