Multiple object tracking using circular Hough transform and particle filter on billiard balls

Report Author
Martin Isaksson

Group Partners
Ali Kamran
Robert Lundberg

Abstract

By combining the technique of how to observe features in images and the technique of particle filter, it is possible to track and localize multiple objects in a video clip. This report will describe such techniques implemented to track billiard balls. Because the objects that are going to be tracked are balls the technique of detecting these objects in an image is circle detection, more specific circular Hough transform. This is something that can be used in many purposes, e.g. in medical purpose when it is important to keep tracking cells or in robot soccer where the robot needs to track the ball all the time.

1 Introduction

Tracking and localizing multiple objects is something that is necessary to master if e.g. the development of a robotic society is going to work. Since a video clip just is a lot of images, or frames, one might think that only image processing is needed to track objects. But to since the fps (frames per second) is limited there will be distortions of the objects in the pictures. Therefore the particle filter is needed, to still being able to track the object even though there is non-rigid deformations of the objects or if the measurement/observation from the circular Hough transform disappears for a little while.

There is a difference of tracking multiple objects compared to tracking a single object. When tracking multiple objects one must think of a computational efficient way, since more issues occur when tracking multiple objects compared to tracking a single object. An issue e.g. can be that to decide which resampling technique to use and how to keep down the sample variance.

When introducing the circular Hough transform, one creates an accumulator space. At those positions in the accumulator space where there are maximums, there are a circle in the original image. If the radius of the billiards balls is unknown, the accumulator space will be 3D. But if a post-processing step is being done, the radius of the billiard balls will be constant (from a camera that is recording from a top-view angle). Then the accumulator space will be 2D and the position of the maximums in the accumulator space will be the centers of the circles in the original image. This means that the tracking can be seen as it is being performed in the accumulator space/Hough space or as it is being performed in the spatial domain/original image.

The basic steps of the circular Hough transform are:

1. Convert image/frame to grayscale.
2. Convolve grayscale image with Sobel operator [1] and calculate the gradient magnitude.

3. Get a binary image by thresholding the image that is the output from step 2.
4. Create an accumulator space with a constant radius and locate where it has its maxima's.

The particle filter basically consists of the following parts (this will be described in detail in section 3):

- Initialize particles
- Prediction
- Data association
- Weighting
- Resampling

The problem of tracking multiple objects is solved by using a particle filter and assigning each measurement (the center of each circle detected by the Hough transform) a cluster center where the particles groups at each circular object. To keep the filter as robust as possible, each cluster get its own resample. This is implemented by using K-means clustering and stratified resampling. If this is going to work properly, the particles has to be able to move from one cluster to another.

Another way to approach the problem of multiple object tracking is to use Gibbs sampling (this will be mentioned more in detail in section 2).

The results of tracking the billiard balls are satisfying, but a better result can easily be achieved by putting more time into the image processing part, e.g. use morphological operators to reduce noise [2]. But since that is not the interesting part here, the focus has been on building and implementing a robust and novel perspective of the particle filter.

1.1 Contribution

By building a circle detection program and implementing it on video frames, one has the foundation of a multiple object tracking program. Once the circle detection is built the particle filter has to be built and adjusted for the specific problem.

Important things to have in mind is how the association should be defined, how to initiate the particles, how should the observation model look like, how should the predict step be built and should there be a control signal **u**, what kind of resampling suits best for the problem, how should the weight be defined when having outliers.

To solve the issues that arose when solving the mentioned problem parts of the particle filter (tailor-made for this problem), a K-means clustering algorithm had to be implemented.

1.2 Outline

Section 1 is the introduction. In section 2 the related work will be discussed. In section 3 the method and implementation will be described in detail. The circular Hough transform will be explained, or more focus will be on Hough space/accumulator space. Then the particle filter will be explained in detail, where the adjustment for this specific problem are discussed and also the problems that arose when implementing the theory. The implementation of K-means clustering algorithm is baked into this.

In section 4 the results will be presented and also the experimental setup.

In section 5 the results and the implementation will be discussed and also the issues that arose. A short summary will be presented and what parts of the method/implementation can be replaced/experimented by other methods in the future.

2 Related work

The Particle Filter

The particle filter is a non-parametric implementation of the well-known Bayes filter [3].Particle filters approximate the posterior by a finite number of parameters. So the key idea with the particle filter is to represent the posterior $bel(x_t)$ by a set of random state samples (also known as particles) drawn from this posterior. As one can understand, this representation of the states is approximate. But since it is non-parametric it can represent a broader space of distributions, not just Gaussians (as for the Kalman filter) [3].

The following articles that will be described, is in many ways related to the work in this report. In most cases the only big difference is how well the image processing part has been made, and then just using a standard particle filter. Therefore some cases where other versions of the particle filter has been implemented are being brought up, since it is interesting also for the issue of this report. In other words, this methods could also had been implemented to solve the problem of tracking billiard balls.

Mixture Particle filter and Learning System

In the article "*A boosted particle filter: Multitarget detection and tracking*" [4] the issue is to track hockey players. In a way this problem is very much alike the problem discussed in this report, that is tracking multiple objects that can enter or leaving the image/frame and also be non-rigid since the fps can be too low. As the title says the problem has been solved in a different way than the problem in this report has been solved. The problem of tracking hockey players is being solved by using a mixture particle filter, which is ideally suited to multi-target tracking. It assigns a mixture component to each player. But this also generates other issues, like how to choose the proposal distribution and how to handle objects (players) entering and leaving the frame. The problem has been solved by constructing a proposal distribution by having a mixture model that uses information from a dynamic model of each object. The other issue is being handled by using Adaboost, i.e. having a system that is capable of learning. Adaboost generates detection hypothesis. By combining Adaboost and the mixture particle filter the tracking will be very powerful. The results of tracking gets better the more training data the program gets. This is something that could have been implemented on the issue in this report, but it is a more complex method to build a program/system that is capable of learning, and therefore the method that has been used to solve the issue in this report is more efficient in the manner of how simple it is to implement.

Fast Particle Filter

In the article "*Multiple object tracking using particle filters*" [5] there is a very detailed explanation from where the particle filter comes from and how it works in different ways. The article describes how to track multiple objects with the same computational complexity as the standard particle filter but with a factor multiplied to that. Different methods that can be used in particle filtering is mentioned and described, e.g. different ways to track multiple objects. So the article describes the whole particle filter very well, but there is a focus on how to keep down the computational complexity, which isn't considered in this report very much.

Similar Problem as Billiard Ball Tracking

In the article "*Real time lane tracking using particle filtering in Hough space*" [2] the issue is kind of similar to the one in this report, but instead of using circular Hough transform, the standard Hough transform for line detection is being used and also there is a knowledge of that there always is two lanes to track, never more than that. Also there is a knowledge of that one lane is the left lane and

the other the right lane, because of that one can use (which has been done here) multiple parallel particle filters. The requirement of being able to use that is that the measurements (in this article that is the maximums in the Hough space) are associated to the correct filter.

The Gibbs Sampler

In the article "Tracking Multiple Objects with Particle Filtering" [6] also here a multi-target tracking problem is being considered. The rare thing that is being done here, compared to the other articles (including this report), is that samples are obtained iteratively from joint posteriors using a MCMC (Markov Chain Monte Carlo) technique, the Gibbs sampler. Gibbs sampler is a special case of the Metropolis-Hasting algorithm.

The Gibbs sampler is for obtaining a sequence of observations that are approximated from the joint probability distribution of multiple random variables. Gibbs sampler generates a Markov chain of samples which is correlated with samples nearby. Therefore problems can arise when independent samples are desired. To go around these problem, *thinning* is usually performed, which means that only every *kth* value is chosen (k is a positive integer, e.g. 1000). This can on the other hand lead to that samples in the beginning of the chain doesn't properly represent the desired distribution.

The Gibbs sampler works in the following way:

Assume that there is a desire of wanting *i* samples of $\mathbf{X}=(x_1, \ldots, x_n)$ from a joint distribution $p(x_1, \ldots, x_n)$. Every *m*th sample is described by $\mathbf{X}^{(m)}=(x^{(m)}_1, \ldots, x^{(m)}_n)$. Begin with some initial value $\mathbf{X}^{(0)}$. Get the next sample each component variable $x^{(m+1)}_j$ from the distribution of that variable conditioned on all other variables by making use of the most recent values and updating the variable with the new value that it gets when it gets sampled. To achieve this, updating of each component variables is required. If the actual component is the *j*th component, an update has to be made according to $p=(x_j | x^{(m+1)}_1, \ldots, x^{(m+1)}_{j-1}, x^{(m)}_{j+1}, \ldots, x^{(m)}_n)$. Then do all this over and over again *i* times [7].

3 My method

Circular Hough Transform

The first thing to do is to make sure that each ball in the video clip gets detected. Also it is important that there aren't too much noise/disturbances in the images/frames.

Grayscale

The first step of building the circular Hough transform is to convert each image/frame to grayscale. This means that first the image is 3D and then it gets converted to 2D. When that has been achieved, the interesting parts of the images are where the gradient magnitude value is higher. To calculate the gradient in x-direction and y-direction filter masks (Sobel operator [1]) are convolved in each direction with the image. Then the magnitude is calculated from those to components.

Thresholding to Binary Image

The magnitudes of the gradients of the image are being selected by a threshold, which means that if the gradient magnitude is larger or equal to a certain threshold, that position will be represented as 1 and if the gradient magnitude is less than that threshold, that position will be represented as 0. This creates a binary image. To know what threshold to use, trial-and-error is the easiest way.

After trial-and-error, when the "perfect" threshold has been picked, the binary image should be black with white circles representing each billiard ball.

Hough Space

The next step is to create the accumulator space/Hough space. If a post-processing step is being done, one can calculate the radius of the billiard balls (assuming that each billiard ball is the same size since the camera angle is from top-view, i.e. the camera angle is perpendicular to the billiard table) and the radius can be constant and then the Hough space will be 2D instead of 3D. This saves computational time. For each white pixel in the binary image, a circle will be drawn in the Hough space (see Figure 1). If there is a circle in the binary image, there will be a maximum in the middle of the "real circle" since a value is added for each time there is an intersection in the accumulator space (see Figure 2).

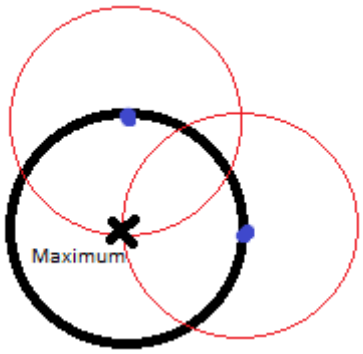

Figure 1 The figure shows a basic example of how a maximum in the Hough space is generated. Here only two circles has been drawn on the "real circle" (the thick black one), but imagine that such a circle with the blue dot as center is being drawn for every black pixel in this figure (for every white pixel in the binary image).
Then the maximum will be where there are the most intersections

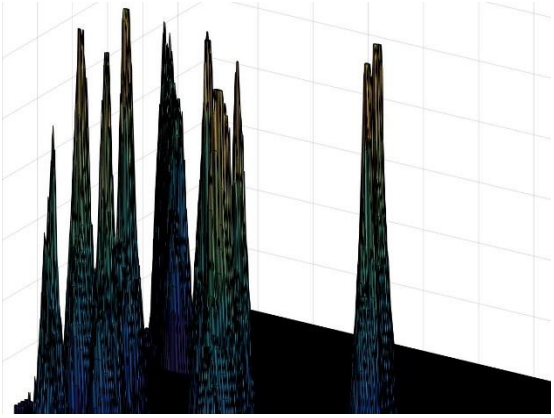

Figure 2 This is the Hough space or the accumulator space. The peaks are the maximums which corresponds to the centers of the circles.

Filter the Hough Space

The last step is to filter the Hough space, i.e. to set up a threshold so that we only get the maximums that are larger than a certain threshold.

Particle Filter

Initialization

The particle filter represents a continuous distribution with a discrete one. It is discretized by randomly picking samples from the distribution. A set of particles $x_{t-1}^{(m)}$ (m=1,…,M particles) is being initialized/drawn from

$$p(x_{t-1}|z_1, \dots, z_{t-1}, u_1, \dots, u_{t-1})$$

, where $\{\boldsymbol{z}_{t-1}\}$ are the measurements, which in this case are the positions of the circles detected by the circular Hough transform. In robot tracking a motion model can often be used which can give a much better tracking result if there is a knowledge of how the robot is about to move, e.g. in a straight line or in a circle. In this case in tracking objects in a video clip when each billiard ball has its own movement, billiard balls can enter the image or disappear from the image and the movement is unknown, the set of control signals $\{\boldsymbol{u}_{t-1}\}$ is zero here.

Prediction

The next step is to propagate to new time. The update corresponds to the prediction step in this case/report. Propagation to new time is done by taking a sample from

$$p\left(x_t \middle| x_{t-1}^{(m)}, u_t\right) \rightarrow \hat{x}_t^{(m)}$$

, but since u_t is zero this is predicted with a diffusion attached to each particle. This can be easily seen when looking at the following equation (knowing that u_t is zero):

$$\bar{x}_t^{(m)} = (x_{t-1}^{(m)} + \bar{u}_t) + N(0, \Sigma_R)$$

, which corresponds to

$$predicted\ state = applying\ motion + diffusion$$

, where Σ_R is the covariance matrix of the modeled process noise. If Σ_R is zero the particles won't be spread out and therefore there won't be a prediction step. The particles will end up on the same spot. If Σ_R is too large the particles won't be able to represent good positions of the billiard ball.

By simplifying the model like this (u_t is zero), it will certainly introduce more uncertainty than if there was a motion model.

Weight and Data Association

The next step is to compute the weight/importance factor for each particle

$$w_t^{(m)} = p(z_t|\hat{x}_t^{(m)})$$

, which is needed to adjust the difference between the sampled distribution (proposal distribution)

$$p(\boldsymbol{x}_t|\{\boldsymbol{u}_t\}, \{\boldsymbol{z}_{t-1}\})$$

and the true posterior (target distribution)

$$p(\boldsymbol{x}_t|\{\boldsymbol{z}_t\}, \{\boldsymbol{u}_t\})$$

So if a particle is very close to the measurement (actually it is the approximated measurement because circles can get detected by the Hough transform that isn't a billiard ball), that particle will get a higher weight than a particle that is further away from the measurement. The weights are normalized so that they all sums up to 1.

When calculating the weights the covariance matrix of the measurement noise is important. Since the weights are calculated from the likelihood (the probability of the measurement given the particle state):

$$\psi = \frac{1}{2\pi|\Sigma_Q|^{\frac{1}{2}}} e^{-\frac{1}{2}\nu^T {\Sigma_Q}^{-1} \nu}$$

, where ν (the innovation) is the difference between the measurement and the predicted state, there is easy to see why the covariance matrix of the measurement noise, Σ_Q, is important. If Σ_Q is close to zero the particles won't get any high weight, not even the particles that should. If Σ_Q is too large the area of where the particles will go to gets too large.

The measurements can be seen as landmarks. If a measurement isn't close enough to a landmark, that measurements is seen as an outlier. That is because one have to look at each cluster separately at each billiard ball. If it's not done this way, e.g. a particle for another billiard ball would be seen as very far away, which would give that particle a low weight, even though that particle is very close to another measurement.

Resampling

The next step is to resample by drawing from the set of $\hat{x}_t^{(m)}$ with probability $w_t^{(m)} \rightarrow x_t^{(m)}$. By using stratified sampling (separate resample of particles of each measurement/cluster) instead of systematic resampling, there will be a lower sample variance which results in that the hypothesis will exist longer, i.e. it is lower risk of having particle deprivation (that is when there aren't any particles where it should be particles, in this case when a billiard ball is detected and there won't be any particles there). The procedure is that first a number of samples are drawn randomly from each subset. That number is decided after the total weight of the particles contained in the subset. Then individual samples are drawn randomly from each subset using a low variance sampling [3].

When using systematic resampling one first calculate $u = \frac{\tilde{u}+(m-1)}{M}$, where $\tilde{u} \sim U[0,1)$, m is the particle number and M is the total number of particles. Then the cumulative sum of all the weights is calculated. For those indexes where the cumulative sum is equal or greater than u, the particles with the same index gets chosen [3].

Stratified sampling works in a very similar way, but except for what was mentioned earlier about the stratified sampling method, systematic resampling generates more uniform samples [8].

K-means Clustering

To group particles around each billiard ball (or measurement) K-means clustering is being used. K-means clustering creates K (the number of how many clusters that should be created) cluster centers. The cluster centers will for each iteration go towards where the highest mean of particles are. When it has reached the mean of the cluster, the iteration will stop. The area of the clusters are defined by the distance from the cluster centers and the particle most far away (by the definition). The way that K-means clustering has been implemented is that there will be as many cluster centers created as there are measurements.

There are some issues that occurs when using K-means clustering. If a cluster of particles gets bigger and bigger, i.e. more spread (this is what happens when a measurement is disappearing). The particle cloud "searches" (see Figure 4) for the measurement. That is thanks to the diffusion step) and when the cloud can't find the measurement, the particles will join the closest cluster. This is something that can be avoided in, e.g. lane tracking [2] when there always are two objects to track. If there are more clusters than measurement it will occur something that looks like a splash/explosion of particles (see

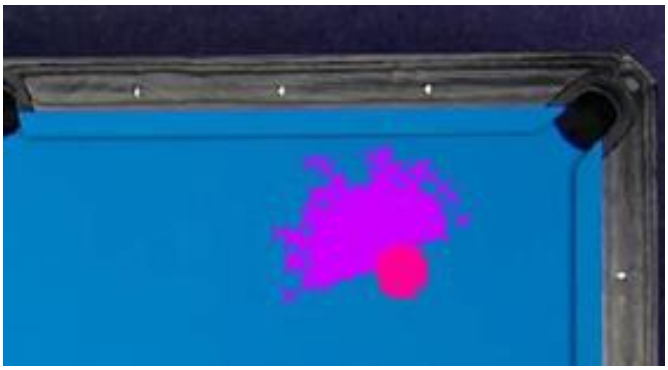

Figure 3 This is what the "splash" looks like.

Figure 4 In this figure there is both merging of clusters and one can see that the measurements has disappeared for two balls, and therefore the particle cluster "searches" for the closest measurement.

Figure 3). This can be solved by merging clusters, but then other issues appear. One issue is that if two clusters that are correct are too close to each other, they will merge, which would be wrong. This is how this problem has been handled in this report though (see Figure 4).

3.1 Implementation

A post-processing step is used to generate all the measurements. This is to save computational time when tracking the billiard balls with the particle filter, since the circular Hough transform can take some time.

Since $\{\boldsymbol{u}_t\}$ is zero, it's by adding the diffusion part to the particle set that is/gives the prediction (which also is the update step in this report/case). This means that the quality of the tracking will basically be depending on the update step.

4 Experimental Results

The experiments that were performed was to test different methods (two different tests) and to see how the different methods affected the result, which in this case is how large the mean square and maximum error is. The tests that were made was to run the tracking (and localizing) program with K-means clustering and the use of systematic resampling. The second test (the most interesting one) K-means clustering and stratified sampling was used, which is the combination mentioned under the method section in this report.

Stratified sampling is when there are groupings and then "individual" sampling, so it's kind of wrong to say "Stratified sampling with K-means clustering", but this is just to make it easier for the reader to understand what has been done/changed in the experiments.

In all the experiments the following process and measurements noise covariance matrices were used, respectively:

$$\Sigma_R = \begin{bmatrix} 750 & 0 \\ 0 & 750 \end{bmatrix}$$

$$\Sigma_Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

, where Σ_R models the error in the position and Σ_Q models the error in the position x and y in Cartesian coordinates.

The ground truth, or the true state, is kind of hard to find. Therefore the distance between the measurements and the cluster centers has been made as an approximation of the true mean square error.

Since that circles/measurements disappear from time to time (because of the image processing part), the distance chosen to represent the mean square error is the shortest distance between the measurements and the cluster centers.

Systematic Resampling with K-means Clustering

This experiment showed a mean square error that is around 7-10 pixels (the distance), which is almost the same size of a billiard ball in the video clip. In this case the maximum error can be as large as 80 but it's more around 50 when it peaks. But it isn't that stable compared to the next scenario with another resampling method. In the end of the video clip when all the balls has stopped moving, the error is just around 1-2 pixels.

Stratified Sampling with K-means Clustering

In this experiment the mean square error was the same as for the case when systematic resampling was used. But when the maximum error peaks, which is also around 50 pixels, it is more stable, i.e. the maximum error doesn't go up to 80 like it did when systematic resampling was used. Also here, in the end of the video clip when all the balls has stopped moving, the error is just around 1-2 pixels.

The Distribution of the Particles in the Beginning

As can be seen in Figure 5, the distribution of the particles is Gaussian, which it should be. This means that everything works fine until the innovation part.

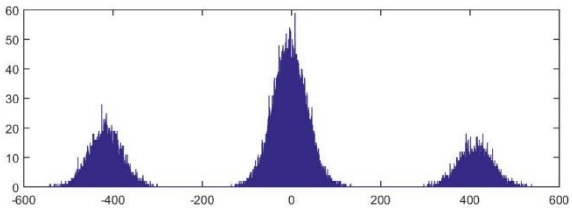

Figure 5 This is the particles distribution in the beginning. On the x-axis there is innovation and on the y-axis there is the number of particles.

5 Summary and Conclusions

As can be seen from last section stratified sampling gave the best results, but systematic resampling wasn't that much worse. That is because both methods of the resampling doesn't differ that much, as mentioned in section 3.

Particle filter was chosen as tracking method because it is easy to implement and can be used for any probability distribution. It can be multimodal (as in the case of this report) and it can be non-linear. The extended Kalman filter (EKF), for example, wouldn't work well when trying to solve the issue of multiple object tracking. That is because the EKF have a lack of basically everything positive mentioned about the particle filter. Though, there is other versions of the EKF like MHT (Multi-Hypothesis Tracking) [3] that can work on multimodal cases.

Figure 6 In this figure there is the result of when all balls are detected and has each measurement.

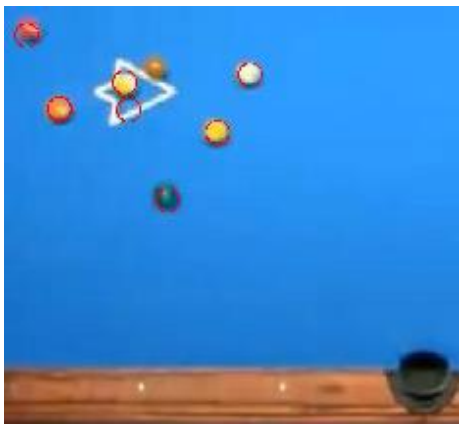

Figure 7 In this figure one ball doesn't get detected and also the white triangular-shaped figure gets a measurement because of the not perfect image processing part.

One can see in Figure 7 that the image processing part, the circular Hough transform, misses billiard balls/measurements, which is a problem when tracking multiple objects. It is good that is can miss a billiard ball in one or two frames, but then come back, just to try the particle filter/tracking part. But when the measurement/detection isn't there for too many frames or even not at all, that is a problem because then the whole part about tracking objects doesn't get fulfilled. Also there is a white triangular

shaped figure in this movie clip that generates measurements/circles, and again that is because the image processing part isn't done well enough (see Figure 7). A way to fix this is to implement a non-linear operator called morphologic operator [2]. That will reduce the noise in the binary image (in the steps of the circular Hough transform algorithm) and also make sure that it detects all billiard balls. When all balls gets detected it looks like in Figure 6.

So, from section 2 one can easily understand that the particle filter is something that can vary in many different way. The implementation is seldom of the standard particle filter anymore (when tracking multiple objects), because one can easy improve that algorithm.

Referenser

[1] ”wikipedia,” 8 January 2016. [Online]. Available: https://en.wikipedia.org/wiki/Sobel_operator. [Använd 11 January 2016].

[2] B. M. &. M. Larsson, ”Real time lane tracking using particle filter in Hough space”.

[3] W. B. &. D. F. Sebastian Thrun, Probabilistic robotics, 2006.

[4] A. T. N. d. F. J. J. L. &. D. G. L. Kenji Okuma, ”A boosted particle filter: Multitarget detection and tracking,” 2004.

[5] H. Ryu, ”Multiple object tracking using particle filters,” August 2006.

[6] J.-P. L. C. &. P. C.Hue, ”Tracking multiple objects with particle filtering,” July 2002.

[7] ”Wikipedia,” 8 December 2015. [Online]. Available: https://en.wikipedia.org/wiki/Gibbs_sampling. [Använd 11 January 2016].

[8] J. D. Hol, ”Resampling in particle filters,” 2004.