OBJECT TRACKING

Harsha Vardhan IMT2016101
Shivam Kumar Singh IMT2016114
Sushma Bhandaru IMT2016116

Introduction:

In the current age of technology, everything is digitalised, every place has surveillance. There are a lot of cameras in place in many places like malls, shopping stores, traffic signals, offices and many more. Everyone is utilizing these surveillance techniques. One such application used is object/motion tracking. This application has been found useful for human-computer interaction, traffic control, augmented reality and so on. We have worked on detecting and tracking vehicles.
Approach:
We use Motion model(speed and direction of motion) + an Appearance model(How the object looks like) to accurately predict the location of the object. But appearances can vary sometimes, so appearance model is usually a classifier to be absolutely sure(if true,1 else 0). This could be done in two ways:

- Offline Classifier(takes 10000+ images to train)
- Online Classifier(trains on the fly at runtime)

We started out with studying two algorithms, namely:

- Kalman Filter
- Condensation Algorithm(Particle Filter)

We will look into these in detail:

Kalman Filter: This is an algorithm that uses a series of measurements observed over time containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone. Kalman filter estimates a joint probability distribution over the variables for each time frame. It is a recursive estimator. The algorithm works in two steps: ***Predict*** and ***Update.***

The model assumes the state of an object to be :

$$\mathbf{x}_t = \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{w}_t,$$

Where xt = state vector, ut = vector containing control inputs, wt = noise parameters, Ft = state transition matrix, Bt = control input matrix.

The main equations involved in this algorithm are:

State update:
$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n \left(y_n - \hat{x}_{n,n-1} \right)$$

State Extrapolation:
$$\hat{x}_{n,n-1} = \hat{x}_{n-1,n-1} + \Delta t \hat{\dot{x}}_{n-1,n-1}$$
$$\hat{\dot{x}}_{n,n-1} = \hat{\dot{x}}_{n-1,n-1}$$
(For constant velocity dynamics)

Kalman Gain :
$$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n}$$

Condensation algorithm: Tracking is the propagation of shape and motion estimates over time, driven by a temporal stream of observations. Tracking objects through highly cluttered scenes is difficult. The noisy observations that arise in realistic issues demand a robust approach involving propagation of probability distributions over time. Modest levels of noise could also be treated satisfactorily using Gaussian densities, and this can be achieved effectively by Kalman filtering. more pervasive noise distributions, as ordinarily arise in visual background litter, demand a lot of powerful, non-Gaussian approach.
An effective approach is to use random sampling. The CONDENSATION algorithm, described here, combines random sampling with learned dynamical models to propagate a whole probability distribution for object position and form, over time. The results the accurate tracking of agile motion in clutter, decidedly more robust than what has previously been possible by Kalman filtering. Despite the utilization of random sampling, the algorithm is efficient, running in near real-time once applied to visual tracking.
.

SAMPLING METHOD :

A standard problem in statistical pattern recognition is to find an object parameterised as x with prior p(x), using data z from a single image. The posterior density p(xlz) represents all the knowledge about x that is deducible from the data. It can be evaluated in principle by applying Bayes' rule

p(xlz) = kp(zlx)p(x)

where k is a normalisation constant that is independent of x. However, p(zlx) may become sufficiently complex that p(xlz) cannot be evaluated simply in closed form. Such complexity arises typically in visual clutter, when the superfluity of observable features tends to suggest multiple, competing hypotheses for x.

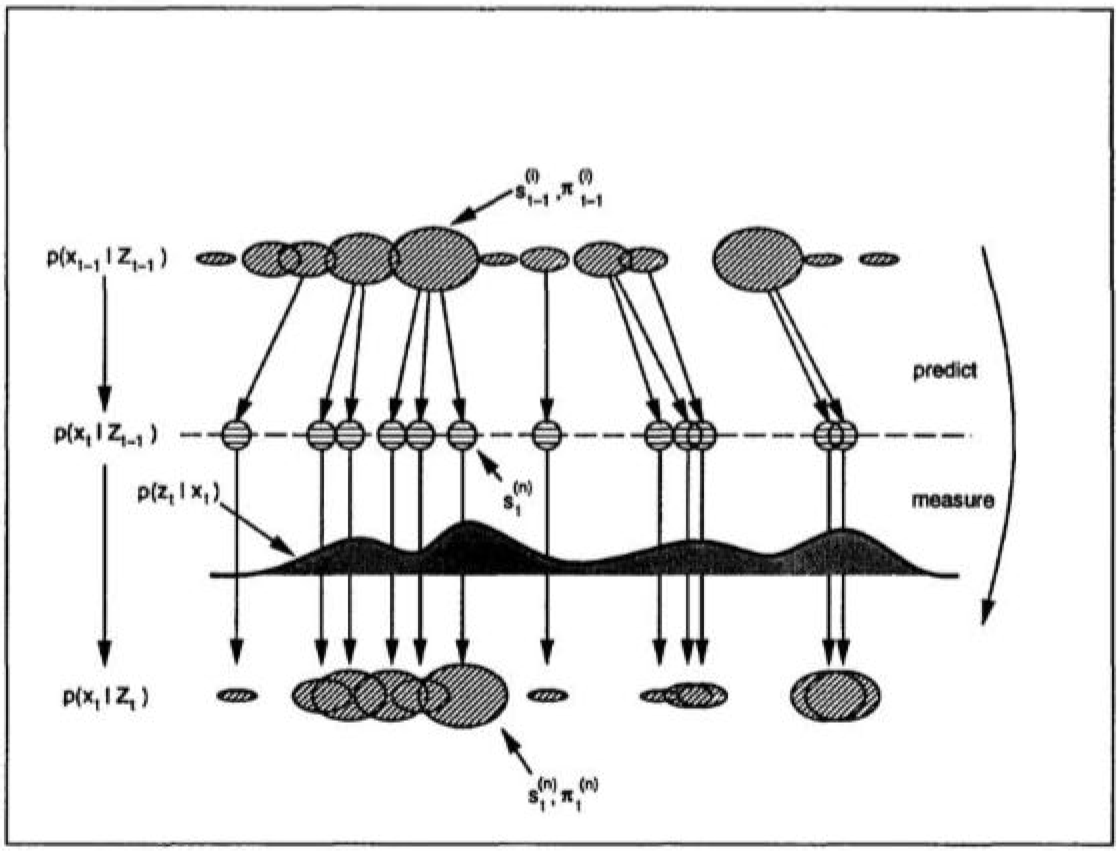
Direct evaluation of p(xlz) is infeasible, iterative sampling techniques can be used The factored sampling algorithm generates a random variate x from a distribution p(x) that approximates the posterior p(xlz).
First a sample-set {s(1), ... , s(N)} is generated from the prior density p(x) and then a sample x = Xi, i belongs to {I, ... , N} is chosen with probability

$$\pi_i = \frac{p(\mathbf{z}|\mathbf{x} = \mathbf{s}^{(i)})}{\sum_{j=1}^{N} p(\mathbf{z}|\mathbf{x} = \mathbf{s}^{(j)})}.$$

Sampling methods have proved remarkably effective for recovering static objects from cluttered images. For such problems, x is multi-dimensional, a set of parameters for curve position and shape. In that case the sample-set {s(1), ... , s(N)} represents a distribution of x-values which can be seen as a distribution of curves in the image plane.

CONDENSATION ALGORITHM :

The CONDENSATION algorithm is based on factored sampling but extended to apply iteratively to successive images in a sequence. Given that the estimation process at each time-step is a self-contained iteration of factored sampling, the output of an iteration will be a weighted, time-stamped sample-set,I representing approximately the conditional state-density $p(x_t\text{I}Z_t)$ at time t, where Z_t = (ZI, ... I Z_t). Clearly, the process must begin with a prior density and the effective prior for time-step t should be $p(x_t\text{I}Z_{t-1}\}$. This prior is, of course, multi-modal in general and no functional representation of it is available. It is derived from the sample set representation $\{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}),$ n = 1, ... , N of $p(X_{t-1}\text{I}Z_{t-1})$, the output from the previous time-step, to which prediction must then be applied.



At the top of the diagram, the output from time-step t - 1 is the weighted sample-set, $\{(s_{t-1}^{(n)}, \pi_{t-1}^{(n)}),$ n = I, . .. ,N}. The aim is to maintain, at successive time-steps, sample sets of fixed size N, so that the algorithm can be guaranteed to run within a given computational resource. The first operation, therefore, is to sample.N times from the set {St-1(n)}' choosing a given element with probability pit-1(n))I' Some elements, especially those with high weights, may be chosen several times, leading to identical copies of elements in the new set. Others with relatively low weights may not be chosen at all.

Each element chosen from the new set is now subjected to a predictive step. {The dynamical model we generally use for prediction is a linear stochastic differential equation (s.d.e.) learned from training sets of sample object motion The predictive step includes a random component, so identical elements may now split as each undergoes its own independent random motion step. At this stage, the sample set {st(n))} for the new time-step has been generated but, as yet, without its weights; it is approximately a fair random sample from the effective prior density p(XtIZt-I) for time-step t. Finally, the observation step from factored sampling is applied, generating weights from the observation density p(Zt IXt) to obtain the sample-set representation {(st(n)), pit(n)'} of state-density for time t. The process for a single time-step consists of N iterations to generate the N elements of the new sample set. Each iteration has three steps

1. **Select** nth new sample s't(n) to be some St-1(j) from the old sample set, sampled with replacement with probability pit-1(j) This is achieved efficiently by using cumulative weights Ct-1(j) (constructed in step 3).
2. **Predict** by sampling randomly from the conditional density for the dynamical model to generate a sample for the new sample-set.
3. **Measure** in order to generate weights pit(n) for the new sample. Each weight is evaluated from the observation density function which, being multimodal in general, "infuses" multimodality. In general," infuse" multimodality into the state density.

At any time-step, it is possible to "report" on the current state, for example by evaluating some

$$\mathcal{E}[f(\mathbf{x}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f\left(\mathbf{s}_t^{(n)}\right).$$

moment of the state density as

Iterate

From the "old" sample-set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}, n = 1, \ldots, N\}$ at time-step $t-1$, construct a "new" sample-set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}\}, n = 1, \ldots, N$ for time t.

Construct the n^{th} of N new samples as follows:

1. **Select** a sample $\mathbf{s}_t'^{(n)}$ as follows:
   (a) generate a random number $r \in [0, 1]$, uniformly distributed.
   (b) find, by binary subdivision, the smallest j for which $c_{t-1}^{(j)} \geq r$
   (c) set $\mathbf{s}_t'^{(n)} = \mathbf{s}_{t-1}^{(j)}$
2. **Predict** by sampling from

$$p(\mathbf{x}_t | \mathbf{x}_{t-1} = \mathbf{s}'^{(n)}_{t-1})$$

   to choose each $\mathbf{s}_t^{(n)}$.
3. **Measure** and weight the new position in terms of the measured features $\mathbf{z}_t$:

$$\pi_t^{(n)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(n)})$$

   then normalise so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$ where

$$\begin{aligned} c_t^{(0)} &= 0, \\ c_t^{(n)} &= c_t^{(n-1)} + \pi_t^{(n)} \quad (n = 1 \ldots N). \end{aligned}$$

Our Implementation:

We implemented the condensation algorithm from scratch in python. Used opencv for video rendering and image observation
We implemented two colour models depending on the surrounding, RGB model and HSV model. In most cases, HSV performed better. We used camshift method of open cv to ensure that the bounding box for the object is dynamic.
We implemented the same with other algorithms and checked its performance. The other algorithms we implemented are mentioned in the next subsection.

Link to the code: https://github.com/shivamkumarsingh114/Kalman-And-Particle-Filter
This repo has GUI based implementation of Kalman filter and Condensation Algorithm. It also has openCV based implementation of Condensation ALgorithm.

Other Algorithms:

Boosting Tracker: Old and a classic method of scanning all the pixels and the score of the classifier is recorded. The maximum score is where the new position of the object is. Needs to be trained at runtime with positive and negative samples of the object.

- It does not reliably know when it fails.
- Used by HAAR cascade face detector internally.

MIL(Multiple Instance Learning) Tracker: Similar to boosting tracker. It considers the nearby locations around the current location to generate several potential positive examples by considering positive and negative bags.

- Performance is pretty good, does a good job and doesn't drift much. Performs reasonably well in case of partial occlusion.
- Does not recover in the case of full occlusion.

KCF (Kernelized Correlation Filters) Tracker : This tracker utilises the fact that multiple positive samples used in MIL tracker have overlapping regions.

- This overlapping data leads to some nice mathematical properties which is exploited by this tracker.
- Faster tracking and more accurate than MIL. Reports tracking failure to a good extent.
- Does not recover from full occlusion.

TLD (Tracking Learning and Detection) Tracker: This tracker decomposes the tracking task into three components: tracking, learning and detection. The tracker follows the object from frame to frame. The detector localises all appearances that have been observed so far and corrects the tracker if necessary. The learning estimates detector's errors and updates it to avoid these errors in the future.

- Works well with occlusion under multiple frames and also tracks best over scale changes.
- Lot of false positives.

Median Flow Tracker: This trackers tracks the object in both forward and backward directions in time and measures the discrepancies(errors) between both the trajectories.

- Detects the tracking failures reliably by minimising the forward-backwards error.
- Works well when the motion is predictable, small and no occlusion.
- Fails under large motion.

GO TURN Tracker: This tracker is based on CNN(Convolution Neural Networks). It is robust to viewpoint changes, lighting changes, and deformations.

- Does not handle occlusions well.

MOSSE Tracker: Minimum output sum of squared error uses adaptive correlation for object tracking which produces stable correlation filters when initialised using a single frame.

- Good performance but not better than the ones based on deep learning

CSRT Tracker: This is a discriminative correlation filter with the channel and spatial reliability. Uses spatial reliability map for adjusting the filter support to the part of the selected region from the frame for tracking.

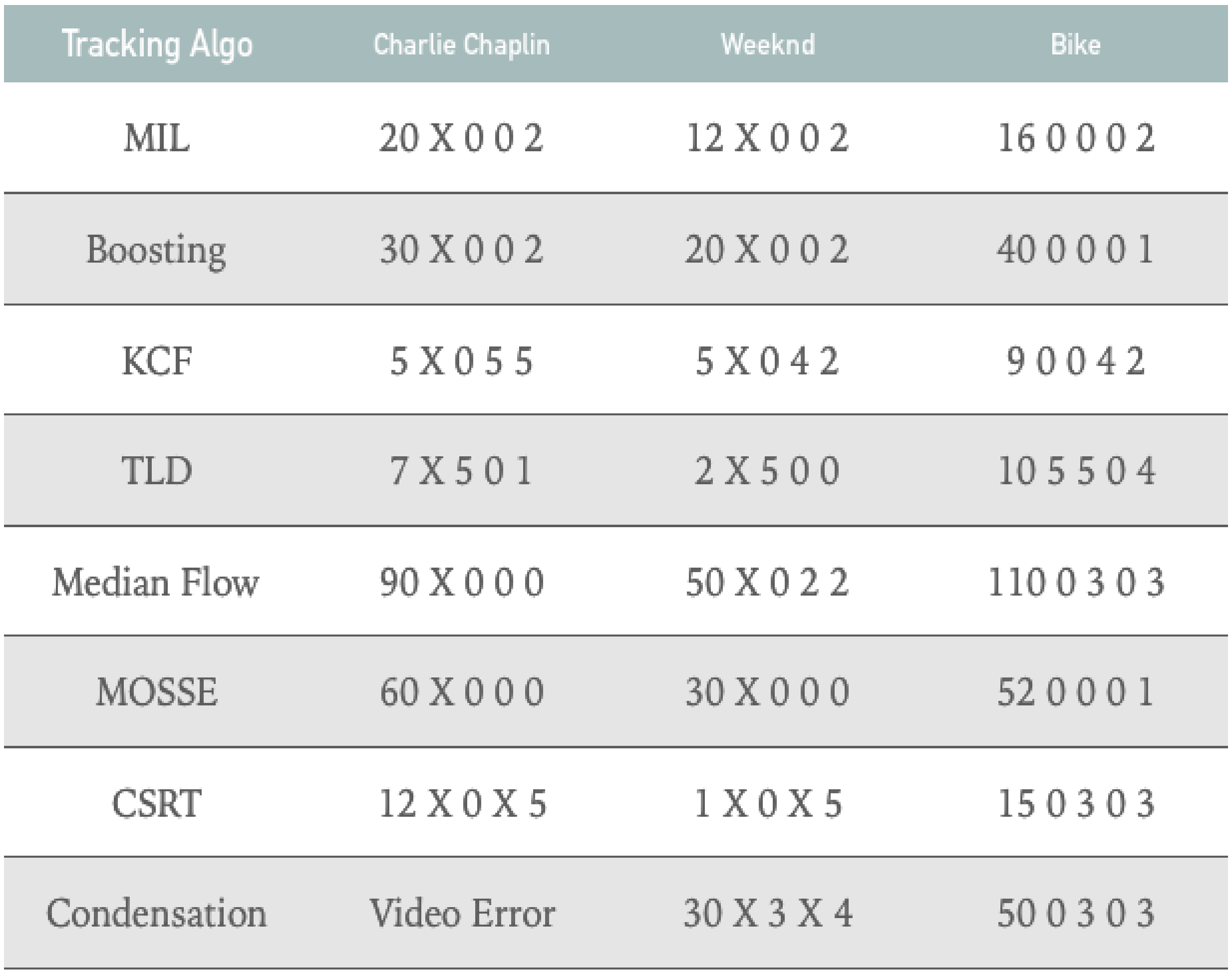
OBSERVATIONS:

After running the algorithms, we rated the performance of the tracking algorithms based on 5 characteristics, namely:

- FPS
- Occlusion
- False positives
- Error detection
- Accuracy

Results:

(FPS, OCCLUSION, FALSE POSITIVE, ERROR DET, ACCURACY)



Tracking Algo	Charlie Chaplin	Weeknd	Bike
MIL	20 X 0 0 2	12 X 0 0 2	16 0 0 0 2
Boosting	30 X 0 0 2	20 X 0 0 2	40 0 0 0 1
KCF	5 X 0 5 5	5 X 0 4 2	9 0 0 4 2
TLD	7 X 5 0 1	2 X 5 0 0	10 5 5 0 4
Median Flow	90 X 0 0 0	50 X 0 2 2	110 0 3 0 3
MOSSE	60 X 0 0 0	30 X 0 0 0	52 0 0 0 1
CSRT	12 X 0 X 5	1 X 0 X 5	15 0 3 0 3
Condensation	Video Error	30 X 3 X 4	50 0 3 0 3

Conclusions:

Condensation Algorithm is able to track objects well. colour model is having a significant impact on the performance of the algorithm which hinders our ability to scale the model significantly.
More in depth analysis is needed to make an scalable model.
We need to modify the implementation to suit our requirements.

Future Work:

We want to implement an error detection mechanism for the algorithm so that it can inform in the case of tracking failure.
We plan to use different colour schemes and check for the best scheme.

We also plan to use the kernel to reduce noisy measurements. We also plan to extend the model to track multiple objects.

References:

1. https://en.wikipedia.org/wiki/Kalman_filter
2. https://www.kalmanfilter.net/default.aspx
3. Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation- Lecture notes, Ramsey Faragher