Pedestrian detection and tracking from camera images

Yuqing Jiao
University of Nevada, Reno for exchange summer in 2017
Department of Computer Science & Engineering, Robot Autonomous Lab
litte_lemon@163.com

ABSTRACT

During the project, I used HOG describer and SVM classier to achieve the function of pedestrian detection. Then Lucas-Kanade method is used to estimate optical flow and achieve tracking moving objects process. Besides, particle filter will be used as a tool to estimate and predict position of a few frames later (not only shows next frame). Ultimately, my goal is to combine the first step and second step--tracking moving pedestrian in detected bounding boxes. The images data or messages come from two different cameras and dataset of the bus driving around the city.

Human detection; particle filter; optical flow tracking; computer vision

I. INTRODUCTION

Detecting and tracking humans from webcam videos are the oldest and challenging tasks in computer vision. Classification in pedestrian detection is to make a distinction between human targets and other objects. The difficulty includes: (1) Color proximity between human clothes and background. (2) Be blocked by obstacles. (3) Shifting figures and poses of pedestrians.
Video tracking refers to the process of estimating, over time, the location of one or multiple objects in a video stream captured by a camera.
It is found in a wide variety of commercial products including face recognition, traffic control, human interaction application, medical imaging, security and surveillance. It is under constant pressure to increase both its quality and speed.
In order to achieve these goals firstly, we need a describer (HOG) to describe the features of human, secondly a classifier (Linear SVM) to distinguish a human or an object, followed by a tracker (Optical flow & Particle filter) to assess the motion and transform motions from 3D-integer coordinates into a 2D real-valued coordinates, and finally also a filter (Particle filter) to estimate the states and predict the location of human if they are in edge of an image. Figure 1.1 shows the whole process. The most important and computational one is the tracker.

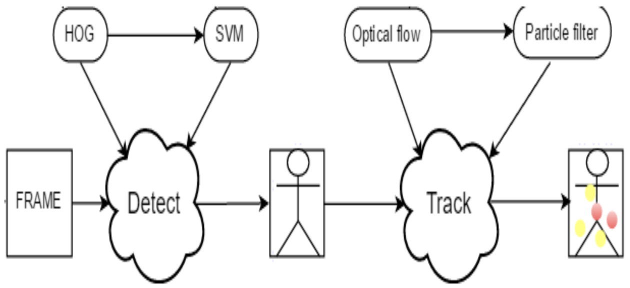


Figure 1.1 total process

The requirements of the tracker are: (1) Accurate, efficient and stable tracker. (2) Track in real traffic environments. (3) Track in real-time. (4) Track pedestrian in a particular area. To make the tracker meet the requirements of its tasks, the main challenges are: (1) Object Describe and Detection. (2) Equipment limitations. (3) Environmental changes. (4) Pose variations. (5) Clutter.[1]

II. RELATED WORK

A. *Histogram of Oriented Gradient (HOG)*

Haar-like features

Viola and Jones adapted the idea of using Haar wavelets and developed the so-called Haar-like features. A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. It concludes following features: edge features (two-rectangle), line features (three-rectangle) and four-rectangle features.

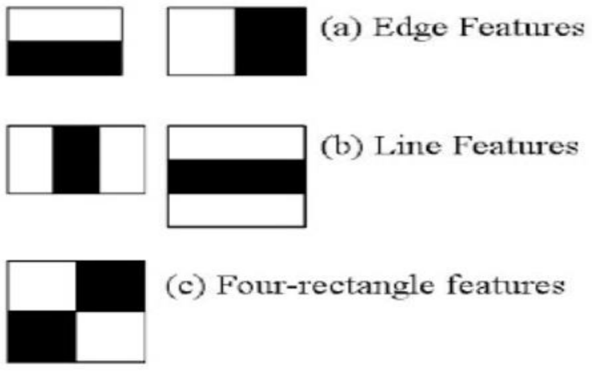


Figure 2.1.1 haar-like features

Each feature is a single value obtained by subtracting the sum of pixels under white rectangle from sum of pixels under a black rectangle.
But haar-like features describer is sensitive to some simple graphical structures, such as edges, lines or rectangle areas, so they can only describe the specific structure (horizontal, vertical, diagonal).
And it is commonly used in a part of human bodies, such as faces. So we use another better feature describer called Histogram of Oriented Gradient.

Histogram of Oriented Gradient
Histogram of Oriented Gradient descriptor was first suggested by Navneet Dalal and Bill Triggs in 2005. It provides a dense overlapping description of image regions to answer the question of feature sets for robust visual object recognition significantly. HOG descriptor has many advantages over other descriptors. First, it is invariant to geometric and photometric transformations then would be applied in larger spatial regions. Second, as Dalal and Triggs discovered, coarse spatial sampling, fine orientation sampling, and strong local photometric normalization permits the individual body movement of pedestrians to be ignored so long as they maintain a roughly upright position.[3] The HOG descriptor is thus particularly suited for human detection in images.
This method has following processes:

- Input image and divide the imade into several detection windows.
- Normalize gamma & color.We used grayscalce in test.
  $\mathrm{I}(\mathrm{x},\mathrm{y}) = I(x,y)^{gamma}$
- Computer gradients.
  $G_x(x,y) = H(x+1,y) - H(x-1,y)$
  $G_y(x,y) = H(x,y+1) - H(x,y-1)$
  $\mathrm{G}(\mathrm{x},\mathrm{y}) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$(Value of gradients)
  $\varphi(x,y) = \tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y)}\right)$ (Angle of gradients)
- Accumulate weighted votes for gradient orientation over spatial cells.
  This step is the fundamental nonlinearity of the descriptor. Each pixel calculates a weighted vote for an edge orientation histogram channel based on the orientation of the gradient element centred on it, and the votes are accumulated into orientation bins over small regions called cells. [3]
  The orientation bins are generaly spaced over 0–360 degree. And nine bins have significantly performance.

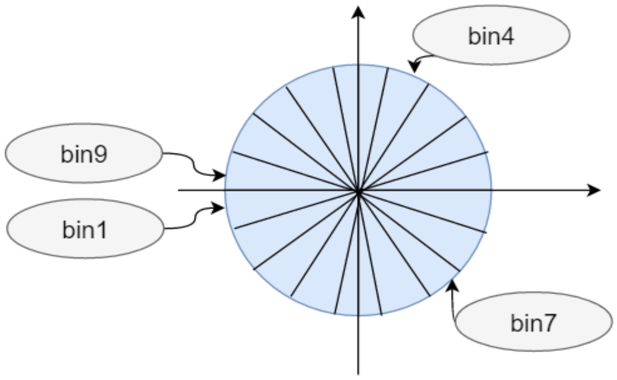
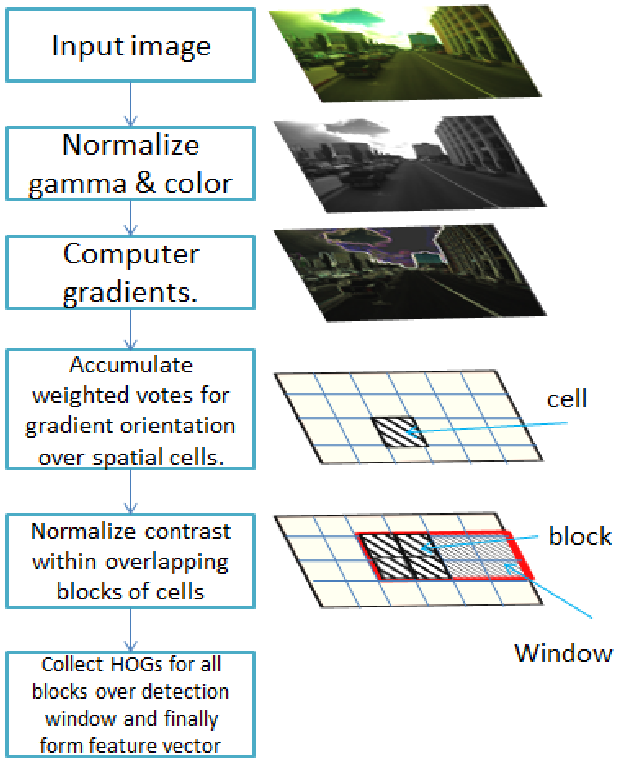


Figure 2.1.2 gradient distribution

- Normalize contrast within overlapping blocks of cells.
- Collect HOGs for all blocks over detection window and finally form feature vector.


Input image
Normalize gamma & color
Computer gradients.
Accumulate weighted votes for gradient orientation over spatial cells.
cell
Normalize contrast within overlapping blocks of cells
block
Window
Collect HOGs for all blocks over detection window and finally form feature vector
Figure2.1.2 the feature extraction and object detection chain.

A simply picture describes how to divide an image into several areas and get vectors. As the figure illustrates a cell is 1*1 and one block equates to four cells which are 2*2, one window is equal to two blocks and eight cells in 4*2 and the image will be scanned by the window. In programming work we can set window stride (It must be a multiple of block stride) and scale0 as coefficient of the detection window increase.
Totally, the detector window is tiled with a grid of overlapping blocks in which Histogram of Oriented

Gradient feature vectors are extracted and will be sent to a linear SVM in next step.

B. *Support Vector Machine (SVM)*

Machine learning helps human deal with mass data and turns data into information by extracting rules or patterns from that data. Support Vector Machine is an accurate object classifier always used in machine learning. It was introduced in 1992 by Boser, Guyon and Vapnik and became rather popular since. The current standard incarnation (soft margin) was proposed by Corinna Cortes and Vapnik in 1993.
There are six steps to train machines and detect human automagically:

- Positive samples (P) from training data of the objects. Negative samples (N) from negative training set that do not contain any of the objects. (N >> P)
- Train a Linear Support Vector Machine on positive and negative samples.
- Extract HOG descriptors and apply classifier. The classifier detects an object with sufficiently large probability from scanning detect window. Then records the bounding box of the window.
- Hard-negative mining. If the classifier incorrectly classifies an object (called false-positives), record the feature vector associated with the false-positive patch along with the probability of the classification.
- Re-train the classifier using these hard-negative samples.
- The classifier can be applied to test dataset.

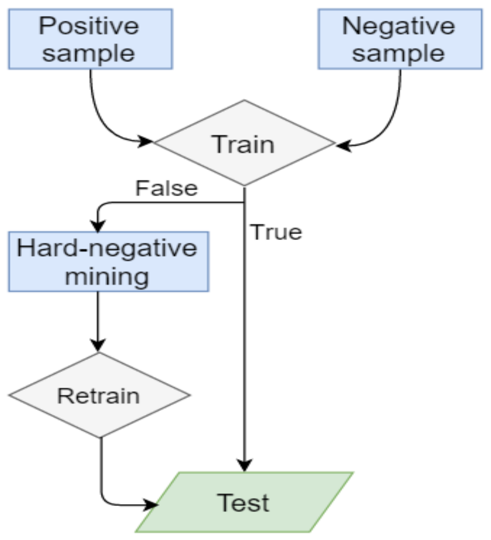


Figure2.2.1 training process

Linear SVM
The simply goal of a linear SVM is to classify two different kinds of vectors, such as red rectangles and blue circles in the given figure2.2.2 by designing a hyperplane (the blue line, for example is an optimal hyperplane).It is clear that there would not be only one hyperplane which can separate binary sets and classify correctly all the instances. But the best choice will be the hyperplane that leave the maximum margin from both instances. Such hyperplane is defined by three equations:

$$f(x) = \beta_0 + \beta^T x$$
$$|\beta_0 + \beta^T x| = 1$$
$$\text{distance} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|} = \frac{1}{\|\beta\|} = \frac{1}{2}M$$

So the total margin which is composed by this distance will be computed by the third equation. If we minimize $\|\beta\|$, the distance will turn the biggest. Nevertheless, this kind of method can only explain for linear conditions, the example is also a very easy 2D or 3D figure so that the hyperplane would be regard as a line or a plane surface $\vec{w} \cdot \vec{x} + \vec{b} = 0$. Another kind of SVM is called non-linear SVM but we don't use such kind of complex tool.

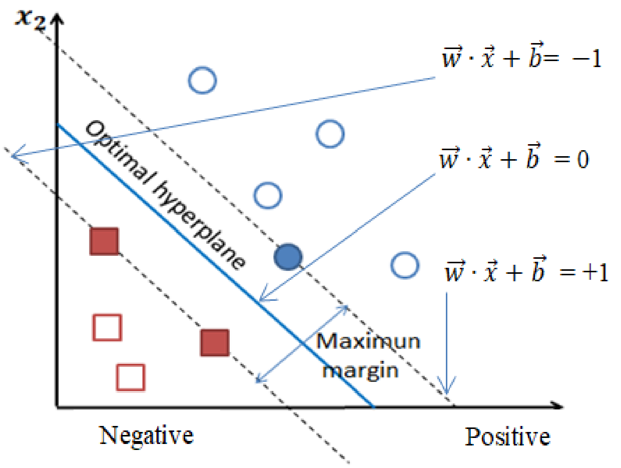
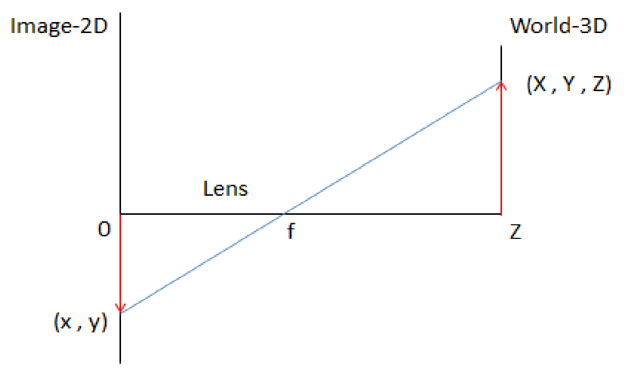

x_2
$\vec{w} \cdot \vec{x} + \vec{b} = -1$
$\vec{w} \cdot \vec{x} + \vec{b} = 0$
$\vec{w} \cdot \vec{x} + \vec{b} = +1$
Optimal hyperplane
Maximun margin
Negative
Positive

Figure2.2.2 optimal hyperplane

Nonlinear SVM
Kernels trick are used to compute nonlinear classifiers by transforming the data from low dimensions space into the higher dimension feature space. Separation may be easier in higher dimensions then a non-linear problem will be transfer to a binary classify problem..

C. *Optical Flow*

Camera Model and Calibration
Because there exist lens distortion, sensors' noisy, the motions of objects are quite different between Cartesian coordinates and image coordinates. We can easily find 2D image points from images. Therefore we need to convert 3D motion vector in real world into 2D optical flow vector in CCD or CMOS.



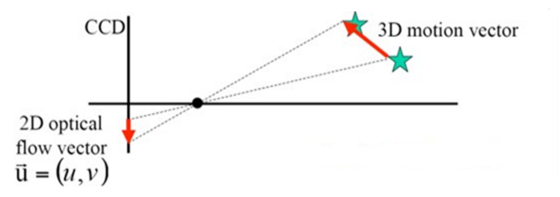


Figure2.3.1 camera calibration

Optical Flow
Once we need to assess motion only between two frames, without any prior frames, we can use optical flow to associate velocity or displacement distance of each pixel in the images. The application of optical flow includes motion and structure of objects in the video scene. The velocities are showed in one-dimension and two-dimension.
One dimension:

$$v = \frac{\partial I}{\partial t} / \frac{\partial I}{\partial x} = \frac{\partial x}{\partial t}$$

Two dimension:

$$u = \frac{dx}{dt}$$

$$v = \frac{dy}{dt}$$

$$-\frac{\partial I}{\partial t} = \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v$$

Lucas-Kanade Method
Compared with Horn-Schunck Method which is a kind of dense optical flow method and associates a velocity with every pixel in an image, the LK algorithm can be applied in a sparse method that reduce high computational costs of dense optical flow. The local information is derived from some small windows surrounding each of the points of interest. The basic idea of the LK algorithm rests on three assumptions. [5]

- Brightness constancy. A pixel from the image of an object in the scene does not change in appearance as it (possibly) moves from frame to frame. For grayscale images, this means we assume that the brightness of a pixel does not change as it is tracked from frame to frame.
- Temporal persistence or "small movements". The image motion of a surface patch changes slowly in time. In practice, this means the temporal increments are fast enough relative to the scale of motion in the image that the object does not move much from frame to frame.
- Spatial coherence. Neighboring points in a scene belong to the same surface, have similar motion, and project to nearby points on the image plane.

Improved optical flow
a) Pyramid Lucas-Kanade
To solve the small movements problem of LK algorithm, a developed pyramid Lucas-Kanade start to track objects from first level (highest) of the image pyramid and then come to the lower levels then to use resulting motion estimates as the starting point to track the next lower and bigger level and therefore, by using this way we can track over image pyramids allows large motions to be caught by local windows.

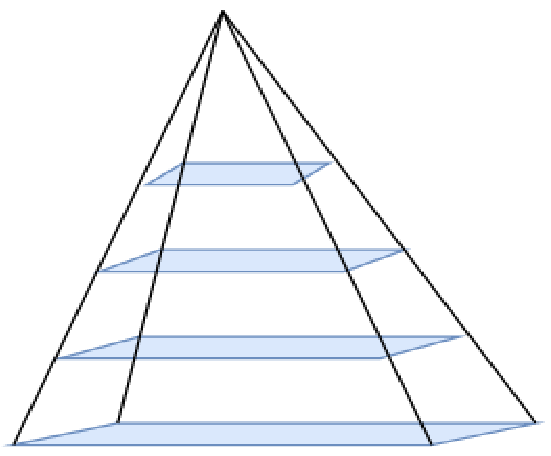


Figure2.3.2 Pyramid Lucas-Kanade

b)Good features points
We look for points, such as a strong derivative that have some significant change and easy to track in the next frame of a video instead of track each pixels of image. Unique or nearly unique points are the best we want. Intuitively, a good feature needs at least: Texture (or ambiguity in tracking) and Corner (or aperture problem).
Such points that contain special information to be picked from one frame to the next were called "corners". The most commonly used definition of a corner was provides by Harris. This definition relies on the matrix of the second derivatives $(\partial^2 x, \partial^2 y, \partial x \partial y)$ and come from the Hessian matrix around a point. The matrix is as follow:

$$H(p) = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix}$$

Overall, we find some points to track first, and receive the optical flow vectors of those points. But when there is large motion, we go for pyramids. After applying Lucas-Kanade method there, we get optical flow along with the scale.

D. *Particle Filter*

Particle Filter

The term "particle filters" was first coined in 1996 by Del Moral. Particle filters methods are a set of genetic-type particle Monte Carlo methodologies to solve filtering problems arising in signal processing and Bayesian statistical inference.

The goal of filtering problem is to estimate states in dynamical systems, considering partial observations as well as random perturbations or noise of sensors. The objective is to compute the posterior probability of the states of some Markov process, given some noisy and partial observations.

In following formulas, $E\big(f(x_n)\big)$ is the mean of states, where x_n represents the state that we want to know in the system. Considering of the complex iteration process, we use Monte Carlo principle to compute conditional probability, which means that expectation instead of integral during math process.

As Sequential Importance Sampling method ①&② describes: We need to ask for the states expectation from posterior probability. Hence, we don't compute the average directly, rather, we need $W_k(x_k)$ such method of weighting. In SIS filtering, there is a degradation problem .The number of effective particles decrease with iteration, while the invalid ones increase and cause computational cost. From ⑤, the less the effective particle number is, the greater the variance of the weight is, the bigger difference between the bigger one and the smaller one there will be, which indicates that the weight degradation is more serious. An improved sequential importance sampling called sequential importance resampling.

Predict (Prior probability)

$$p(x_k|y_{1:k-1}) = \int p(x_k, x_{k-1}|y_{1:k-1})dx_{k-1}$$
$$= \int p(x_k|x_{k-1}, y_{1:k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1}$$
$$= \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1}$$

Update (Posterior probability)

$$\mathrm{p}(x_k|y_{1:k}) = \frac{p(y_k|x_k, y_{1:k-1})p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})}$$
$$= \frac{p(y_k|x_k)p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})}$$
$$p(y_k|y_{1:k-1}) = \int p(y_k|x_k)p(x_k|y_{1:k-1})dx_k$$

Monte Carlo principle

$$\mathrm{E}\big(\mathrm{f}(x_n)\big) \approx \int f(x_n)\hat{p}(x_n|y_{1:k})dx_n = \frac{1}{N}\sum_{i=1}^{N} f(x_n^{(i)})$$
$$\hat{p}(x_n|y_{1:k}) = \frac{1}{N}\sum_{i=1}^{N} \delta(x_n - x_n^{(i)}) \approx \mathrm{p}(x_n|y_{1:k})$$

Sequential Importance Sampling (SIS)

$$\mathbf{E}\big(\mathbf{f}(\boldsymbol{x_n})\big) = \int f(x_n)\frac{p(x_k|y_{1:k})}{q(x_k|y_{1:k})}q(x_k|y_{1:k})dx_k$$
$$= \int f(x_n)\frac{p(y_{1:k}|x_k)p(x_k)}{p(y_{1:k})q(x_k|y_{1:k})}q(x_k|y_{1:k})dx_k$$
$$= \int f(x_n)\frac{W_k(x_k)}{p(y_{1:k})}q(x_k|y_{1:k})dx_k$$
$$= \int f(x_n)\frac{W_k(x_k)}{\int p(y_{1:k}|x_k)\, p(x_k)dx_k}q(x_k|y_{1:k})dx_k$$
$$= \frac{\int f(x_n)W_k(x_k)q(x_k|y_{1:k})dx_k}{\int W_k(x_k)q(x_k|y_{1:k})\, dx_k}$$
$$\approx \frac{\frac{1}{N}\sum_{i=1}^{N} W_k(x_k^{(i)})f(x_k^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} W_k(x_k^{(i)})}$$
$$\mathbf{E}\big(\mathbf{f}(\boldsymbol{x_n})\big) = \sum_{i=1}^{N} \widetilde{\boldsymbol{W}}_{\boldsymbol{k}}\left(\boldsymbol{x}_{\boldsymbol{k}}^{(\boldsymbol{i})}\right)\boldsymbol{f}\left(\boldsymbol{x}_{\boldsymbol{k}}^{(\boldsymbol{i})}\right) \quad ①$$

$$\boldsymbol{W_k}(\boldsymbol{x_k}) = \frac{p(y_{1:k}|x_k)p(x_k)}{q(x_k|y_{1:k})} \propto \frac{p(x_k|y_{1:k})}{q(x_k|y_{1:k})}$$
$$W_k{}^{(i)} \propto \frac{p\big(x_{0:k}{}^{(i)}\big|Y_k\big)}{q\big(x_{0:k}{}^{(i)}\big|Y_k\big)}$$
$$[Y_k = y_{1:k};\ x_{0:k} = x_k]$$
$$= \frac{\dfrac{p\big(y_k\big|x_{0:k}{}^{(i)}, Y_k\big)p\big(x_{0:k}{}^{(i)}\big|Y_{k-1}\big)}{p\big(y_k\big|Y_{k-1}\big)}}{q(x_k{}^{(i)}|x_{0:k-1}{}^{(i)}, Y_k)q(x_{0:k-1}{}^{(i)}|Y_{k-1})}$$
$$= \frac{p\big(y_k\big|x_{0:k}{}^{(i)}, Y_{k-1}\big)p\big(x_k{}^{(i)}\big|x_{0:k-1}{}^{(i)}, Y_{k-1}\big)p\big(x_{0:k-1}{}^{(i)}\big|Y_{k-1}\big)}{p(y_k|Y_{k-1})q(x_k{}^{(i)}|x_{0:k-1}{}^{(i)}, Y_k)q(x_{0:k-1}{}^{(i)}|Y_{k-1})}$$
$$= \frac{p\big(y_k\big|x_k{}^{(i)}\big)p\big(x_k{}^{(i)}\big|x_{k-1}{}^{(i)}, Y_{k-1}\big)p\big(x_{0:k-1}{}^{(i)}\big|Y_{k-1}\big)}{p(y_k|Y_{k-1})q(x_k{}^{(i)}|x_{0:k-1}{}^{(i)}, Y_k)q(x_{0:k-1}{}^{(i)}|Y_{k-1})}$$

$$W_k{}^{(i)} \propto \frac{p\big(y_k\big|x_k{}^{(i)}\big)p\big(x_k{}^{(i)}\big|x_{k-1}{}^{(i)}\big)p\big(x_{0:k-1}{}^{(i)}\big|Y_{k-1}\big)}{q(x_k{}^{(i)}|x_{0:k-1}{}^{(i)}, Y_k)q(x_{0:k-1}{}^{(i)}|Y_{k-1})}$$
$$= W_{k-1}{}^{(i)}\frac{p\big(y_k\big|x_k{}^{(i)}\big)p\big(x_k{}^{(i)}\big|x_{k-1}{}^{(i)}\big)}{q(x_k{}^{(i)}|x_{0:k-1}{}^{(i)}, Y_k)}$$
$$= W_{k-1}{}^{(i)}\frac{p\big(y_k\big|x_k{}^{(i)}\big)p\big(x_k{}^{(i)}\big|x_{k-1}{}^{(i)}\big)}{q(x_k{}^{(i)}|x_{0:k-1}{}^{(i)}, Y_k)}$$

$[q(x_k|x_{0:k-1}, y_{1:k}) = q(x_k|x_{k-1}, y_k)]$

$$\boldsymbol{W_k}^{(i)} \propto \boldsymbol{W_{k-1}}^{(i)} \frac{\boldsymbol{p(y_k|x_k{}^{(i)})p(x_k{}^{(i)}|x_{k-1}{}^{(i)})}}{\boldsymbol{q(x_k{}^{(i)}|x_{k-1}{}^{(i)}, y_k)}} \quad ②$$

Sequential Importance Resampling (SIR)

$$N_{eff} = N/(1 + var(w_k^{*(i)})) \quad ⑤$$

$$w_k^{*(i)} = \frac{p(x_k{}^{(i)}|y_{1:k})}{q(x_k{}^{(i)}|x_{k-1}{}^{(i)}, y_{1:k})}$$

$$W_k{}^{(i)} \propto W_{k-1}{}^{(i)} \frac{p(y_k|x_k{}^{(i)})p(x_k{}^{(i)}|x_{k-1}{}^{(i)})}{q(x_k{}^{(i)}|x_{k-1}{}^{(i)}, y_k)}$$

$$[q(x_k{}^{(i)}|x_{k-1}{}^{(i)}, y_k) = p(x_k{}^{(i)}|x_{k-1}{}^{(i)})]$$

$$\boldsymbol{W_k}^{(i)} \propto \boldsymbol{p(y_k|x_k{}^{(i)})} \quad ③$$

$$\boldsymbol{x_k}^{(i)} \sim \boldsymbol{p(x_k|x_{k-1}{}^{(i)})} \quad ④$$

$$\mathbf{E}(\mathbf{f}(\boldsymbol{x_n})) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \widetilde{\boldsymbol{W}}_{\boldsymbol{k}}\left(\boldsymbol{x_k^{(i)}}\right) \boldsymbol{f}\left(\boldsymbol{x_k^{(i)}}\right)$$

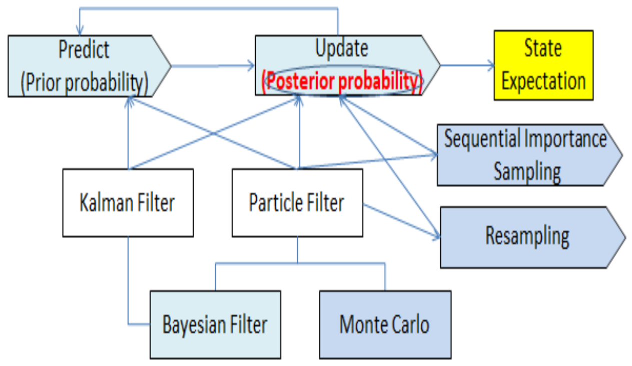
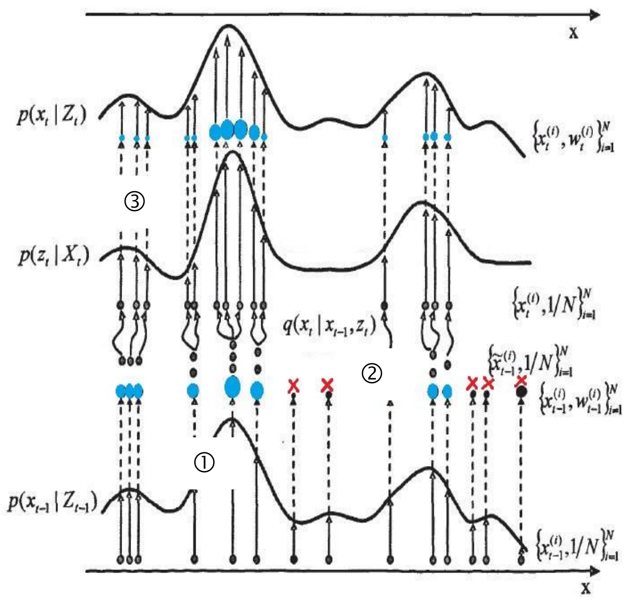


Figure2.4.1 Particle & Kalman Filter process



x
$p(x_t|Z_t)$
$\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$
③
$p(z_t|X_t)$
$\{x_t^{(i)}, 1/N\}_{i=1}^N$
$q(x_t|x_{t-1}, z_t)$
②
$\{\tilde{x}_{t-1}^{(i)}, 1/N\}_{i=1}^N$
$\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^N$
①
$p(x_{t-1}|Z_{t-1})$
$\{x_{t-1}^{(i)}, 1/N\}_{i=1}^N$
x

Figure2.4.2 resampling particles

The given figure 2.4.2 describes the particles resampling process. ① shows particles turn big or small though weighing.② is resampling process used to avoid the problem of degeneracy, small ones will not applied in next sampling. In ③ $p(x_k|Z_t)$ means posterior probability and the particles update then weight again.
A normal SIR filter is as follows:

- Draw samples from the proposal distribution.
- Update the importance weights up to a normalizing constant.
- Compute the normalized importance weights.
- Compute an estimate of the effective number of particles.
- If the effective number of particles is less than a given threshold, draw N particles from the current particle set with probabilities proportional to their weights. Replace the current particle set with this new one.

Kalman Filter

The Kalman filter is a special class of particle filter, also known as a linear Gaussian state space model, which is analytically tractable. When the linearity or Gaussian conditions do not hold, the extended Kalman filter can be used. However, for highly non-linear and non-Gaussian problems they fail to provide a reasonable estimate.

$x_t = f(x_{t-1}, u_t)$, where $u_t \sim$ N (0, Σ)

$y_t = h(x_t, v_t)$, where $v_t \sim$ N (0, Σ y)

Predict

$$\bar{x_t} = g(x_{t-1} + u_t)$$
$$\bar{P_t} = G_t P_t G_t^T + R_t$$

Update

$$K_t = P_t H_t^T$$
$$x_t = \bar{x_t} + K_t(z_t - h(\bar{x_t}))$$
$$P_t = (I - K_t H)\bar{P_t}$$

III. RESULTS

We test the tracker as well as the detector, using image messages data from cameras and buses.

a)Compared two different Cameras：There are two cameras used to test.

Camera	**Microsoft LifeCam Cinema**	**CM3-U3-13Y3C-CS point Grey Inc.**
Resolution	1280 × 720	1280 × 1024
Sensor	CMOS sensor	CMOS sensor
Power	/	5 V via USB 3.0
Image rate	30 fps	0-149fps

Picture		

Chart 3.1 Cameras specifications

It is very clear that the lifespan camera works faster than the second one, which has high resolution as well as CPU occupancy rates and need to change frame rate at 10fps to keep real-time tracking. By contract, with high resolution, CMU is easier to find corners from targets. In fact, the quality of camera is also very important in computer vision.

b)Dataset from buses around the city(Reno, NV)
Green rectangles are bounding boxes show location of human. And we also applied non-maximum suppression to remove redundant and overlapping bounding boxes.
Few yellow points represent resample particles first time, and red ones represent resample particles second time if too few effective particles at first time.
Figure 3.1-3.3 show good results. The detector and tracker work well include identifying human, finding good feature points, following human's movements, and predicting locations of pedestrians.
Figure 3.4 are bad results. Both detector and tracker make mistakes. The classifier regards streetlight as human (draw a green rectangle). Besides, the road signs are a part of environments and will never move, but particles track them and regard them as targets.

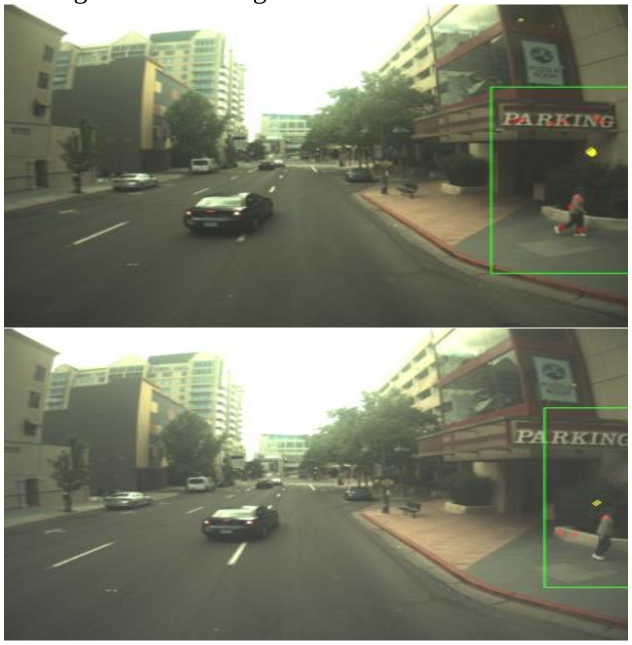


Figure 3.1 find good points

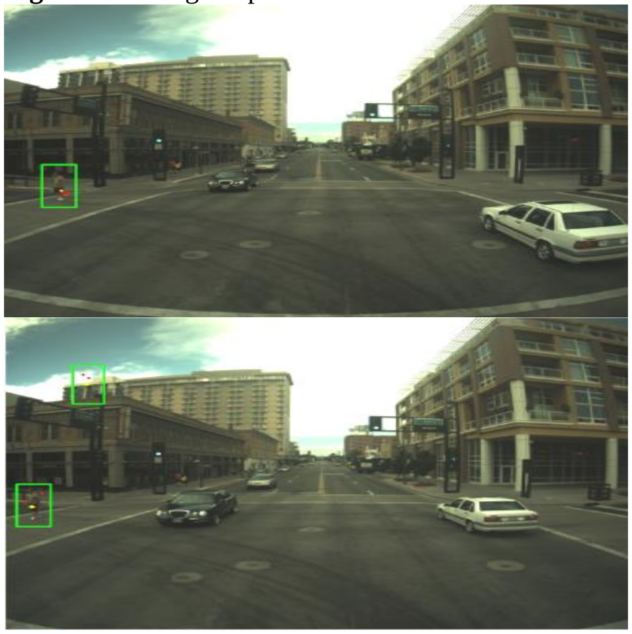


Figure 3.2 particles

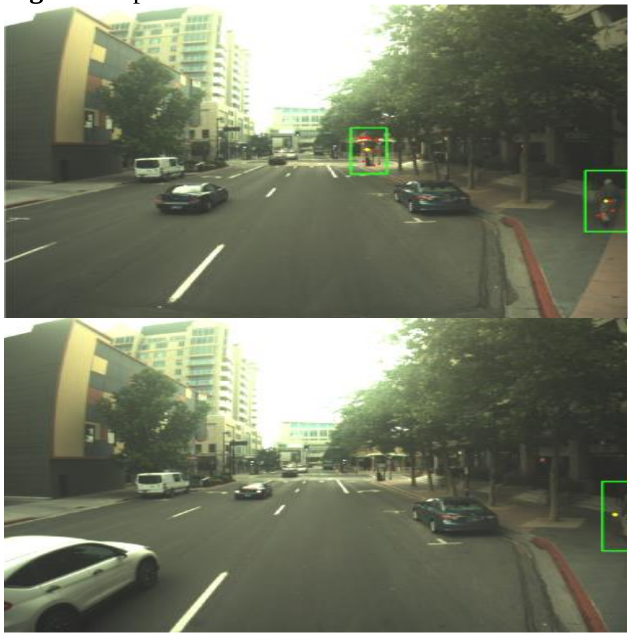
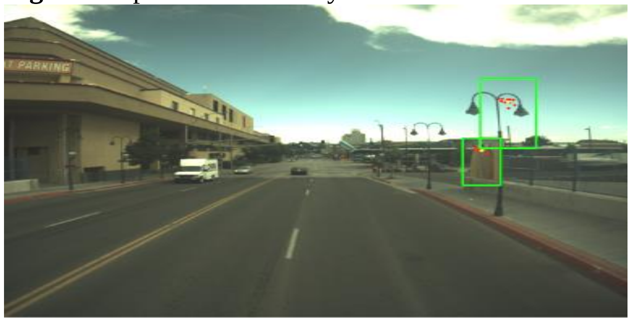


Figure 3.3 predict where they are



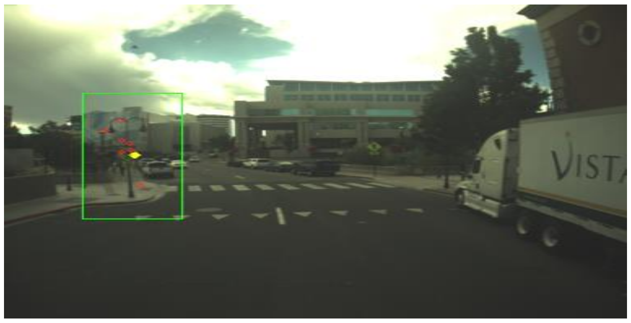


Figure 3.4 negative results

IV. CONCLUSION

Combined four main algorithms we can detect and track objects from video scenes effectively and successfully.

However there are also some problems during testing process, especially these algorithms and methods possibly will be used in cities traffic system and autonomous vehicles.

1. We need to create more positive samples of pedestrian images with a large range of human poses and fast-moving background, and also more negative samples, such as trees, signs, buildings, traffic lights from traffic environments which are easily cause confusion and bad result. The new and good samples will be sent to train our own SVM classifier (instead of classifier provided by OpenCV, which doesn't suit traffic system) then help the detector to identify human accurately.
2. We found that the performance of the tracker varied with the kind of video sequence. It was difficult to find good values for the different parameters (number of particles, stable environment, HD pictures, number of weak classifiers, slow frame rate) that would give an overall good performance. If the tracker will be used in real-time and real-environment, the difficult is related to algorithm, processor and cameras.
3. From the test results, I think there are some limitations due to optical flow algorithm.
   Firstly, three assumptions of optical flow ask for brightness constancy, temporal persistence or "small movements" and spatial coherence.
   However, our test data from buses driving around the city may not match requirements. The high speed changing pictures could not be regarded as "small movements" or keep constant brightness. Moreover the pattern of apparent motion of image objects between two consecutive frames cause by movement of object or camera. But in fact, the environments moved very fast because of vehicle speeds. Compared with buses or cameras, walking pedestrian movements are not apparent and difficult to track.
   Secondly, although large motions had been solved with "image pyramidal", the tracking areas are changeable when the bounding boxes change locations easily. Because of this, variable surroundings cannot guarantee brightness constancy and cause high computational costs. A fixed region of the images will track very fast, while changed one will track slowly.

V. REFERENCES(IF NECESSARY)

[1]Samuelsson O. Video Tracking Algorithm for Unmanned Aerial Vehicle Surveillance[J]. Electrical Engineering Electronic Engineering Information Engineering, 2012.
[2]Lucas B D, Kanade T. An iterative image registration technique with an application to stereo vision[C] International Joint Conference on Artificial Intelligence. Morgan Kaufmann Publishers Inc. 1981:674-679.
[3]Dalal N, Triggs B. Histograms of oriented gradients for human detection[C] IEEE Computer Society Conference on Computer Vision & Pattern Recognition. IEEE Computer Society, 2005:886-893.
[4]Cortes C,Vapnik V(1995)."Support-vector networks". Machine Learning. 20 (3): 273–297.
[5] "Learning OpenCV" Book by Adrian Kaehler and Gary Rost Bradski.
Blog concludes PyImagesearch and CSDN.
Websites includes online OpenCV Tutorial, Wikipedia, Kostas Alexis's online lectures ,Stanford Computer Science online lectures.

VI. CODES (IF NECESSARY)

I used OpenCV3.2.0-python2.7 during the project. Here are some core codes to achieve algorithms:

```
Hog = cv2.HOGDescriptor()
Hog.setSVMDetector
Hog.detectMultiScale
cv2.goodFeaturesToTrack
cv2.calcOpticalFlowPyrLK
cv2.calcOpticalFlowFarneback
```