ISSN:0975-9646

Object Tracking and Motion Estimation Using Particle Swarm Optimization

Yateen Kedare[1], Rohit Thakare[2], Jayesh Raghuwanshi[3], Vedant Deokar[4]

Department of Computer Science, Smt. Kashibai Navale College of Engineering, Savitribai Phule Pune Pniversity, Pune, India

***Abstract*— In order to robustly track any random object we usually require lots of processing power. Tracking objects from our day to day computational devices may introduce a lag in the tracking system. In order to speed up the process we are using Particle swarm optimization for tracking. This method allows us to track objects with minimal hardware requirements. Output from Particle Swarm Optimization usually contains little noise. In order to eliminate this noise and estimation the trajectory of the object we use Kalman filter.**

***Keywords*— PSO, Kalman Filter, Object tracking, Motion Estimation.**

I. Introduction

Conventional methods of object detection and tracking require lots of processing power. Even the best of the hardware require lots of time in order to process the visual data when it comes to object following. Object detection techniques give robust outputs. Such detection algorithms if used repeatedly for tracking the object will consume lot of CPU time and hence will introduce a lag in the system.

In this project, in order to deal with the object tracking problem, we are using particle swamp optimization technique along with Kalman filters in order to predict the path of the object and remove the noise in the output.

The focus of this project is to enable our day to day computational devices such as laptops, hand held devices and embedded devices to track objects robustly without any lag. Empowering embedded devices to process visual data efficiently can be of great use in field of Robotics and Automation.

II. Related Work

Particle Swarm Optimization was first cited in [1] where general idea of PSO was explained as an optimization algorithm to expedite searching. In [2] Histograms were used in order to track objects. Every frame was divided in a grid and histogram was calculated for each of the element in the grid and result was computed by comparing the fitness function value between target object histogram and grid elements. In [3] PSO was used for object detection in a given search space and grey level histograms were used for object tracking. In [4], PSO was used to improve the performance of face recognition by optimizing elastic bunch graph matching technique.

III. Particle swarm optimization

PSO is an optimization algorithm which originates from a swarm of birds trying to search for food. When birds are searching for food (goal state) in a swarm all birds move around in a random direction. Every bird looks out for food by itself (local best) and also keeps a tap on other birds to see whether anyone one of them has found food (global best). When food is spotted by some bird in the swarm every bird changes its position according to the new global best.

In order to track the object using the above method we create multiple particles randomly in the search space. Histogram of the target object to be tracked is considered as the goal state. Each particle computes histogram of the selected search space and compares it with the goal state histogram in order to find out the fitness value of any iteration for the given particle. Global best is the best value of fitness function achieved in iteration. Local best value is the best position achieved by the particle so far. In order to compute fitness function, Bhattacharyya Coefficient can be used [6].

$$BC\big(H(t), H(p_i, t+1)\big) = \sum_{x \in X} \sqrt{H_x t, H_x(p_i, t+1)}$$

Where *H(t)* represents the histogram of target object, *$H(p_i,t+1)$* is the histogram of particle *i* and *X* denotes the distribution domain.

By using the above equation the distance between two histograms can be calculated as [7]:

$$D\big(H(t), H(p_i, t+1)\big) = \sqrt{1 - BC\big(H(t), H(p_i, t+1)\big)}$$

Thus the fitness function for every particle *i* should be inversely proportional to the distance between $H_x(p_i, t+1)$ and $H_x(t)$:

$$F(p_i, t+1) = 1/D\big(H(t), H(p_i, t+1)\big)$$

The position and velocity of the particles are updated according to the following equations at any iteration:

$$v_{t+1}^{id} = w.v_t^{id} + c_1.\varphi_1\big(p_t^{id} - x_t^{id}\big) + c_2.\varphi_2\big(p_t^{gd} - x_t^{id}\big)$$

$$x_{t+1}^{id} = x_t^{id} + v_{t+1}^{id}$$

Where *i* represent the particles in dimension *d*. v_t is the current velocity of the particle and v_{t+1} is the new velocity. *w, c1, c2* are constant weights. φ_1 and φ_2 are randomly generated variables between (0,1) interval. p_t^{id} is the local best value achieved by the i^{th} particle in dimension *d* and p_t^{gd} is the global best fitness value achieved in that iteration.

Object tracking using PSO can be termed as a four variable problem which contains Δx, Δy, Δh and Δw, where *(x,y)* is the position of the particle in *xy* plane, Δh is change in height and Δw is change in width of the output bounding box. Above equations of PSO can be applied to all these four variables for the result.

IV. KALMAN FILTER

The Kalman Filter algorithm is an asymptotic state estimator for multiple dimensions. It predicts the future state of the system based on the previous states. We use this algorithm in order to filter the output generated by PSO and also to estimate the trajectory of the object.

Kalman filter can be used for filtering of multi-dimensional data. In our case, we get data from two dimensions, i.e. x and y. This method can be extended and applied to height and width parameters of PSO for further smoothening of output bounding box.

Since we want to predict the data in two dimensions i.e. x and y our standard models of kalman filter would be as follows.

A. State Update Model

The State update model for 2D kalman filter will contain components of position x, y and velocity $\dot{x}$ *and* $\dot{y}$. This can be represented as:

$$\overline{x_t} = Ax_{t-1} + B\mu_t + E_x$$

Where $\overline{x_t}$ denotes estimate of the new state, x_{t-1} is the data from previous state. A and B are coefficient matrices. μ_t is the control input variable and E_x is the error variance.

This can be represented in the matrix form as:

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} \frac{T^2}{2} \\ \frac{T^2}{2} \\ T \\ T \end{bmatrix} . \mu_t + E_x.$$

B. Measurement Update model

Measurement update model is used to change our parameters of current model when we get new data. In this case we'll be measuring two parameters i.e. x position and y position.

$$\overline{z_t} = C\overline{x_t}$$

Where z_t is the measurement and $\overline{x_t}$ is the current state update model. C is our measurement function which we apply to the state estimate in order to get our expected new measurement.

We can represent measurement function matrix C as:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Thus we can apply our state update model and measurement update model as described in [8].

C. Error Variance

We can define the measurement error as:

$$E_z = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$

Where σ_x is variance in x and σ_y is the variance measure in y.

Error in the input to the system or the process can be described as:

$$E_x = \begin{bmatrix} \frac{T^4}{4} & 0 & \frac{T^3}{2} & 0 \\ 0 & \frac{T^4}{4} & 0 & \frac{T^3}{2} \\ \frac{T^3}{2} & 0 & T^2 & 0 \\ 0 & \frac{T^3}{2} & 0 & T^2 \end{bmatrix}$$

V. PSO-KALMAN ALGORITHM SUMMARY

The PSO-Kalman algorithm for object tracking and motion estimation can be summarised by the following pseudo code.

```
While(newImage is available) {
    If(target object histogram not valid) {
        Select target on the image frame;
        Update target histogram;
    }
    Else {
        If(particles not initialised) {
            Initialise particles;
        }
        Else {
            For (each particle) {
                Calculate fitness;
                Update local best;
            }
            Compute Global best;
            For(each particle) {
                Update position;
                Update velocity;
            }
            Apply kalman filter on the best particle;
            Display the results of kalman filter;
        }
    }
}
```

VI. EXPERIMENTAL RESULTS

In order to test the efficiency and robustness the above algorithm was implemented in openCV 3.0 with C++. It was tested under dynamic conditions on various systems with varying OS platforms. Some of the results are depicted in the figures below.

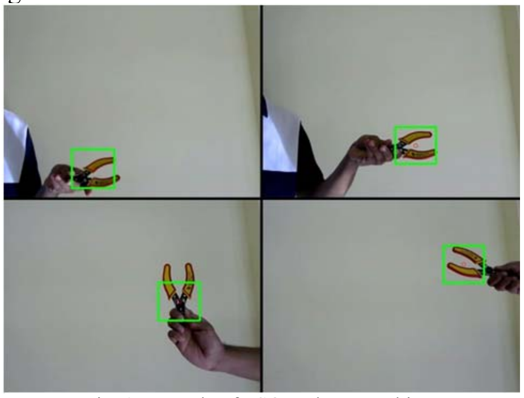
Fig. 1 Example of PSO-Kalman tracking.

We can see that the PSO-Kalman based tracking algorithm is rotation invariant from Fig. 1. It shows the result of an object rotated by 180^{o}. Thus the object in the image frames need not be in the same position as the given goal state.

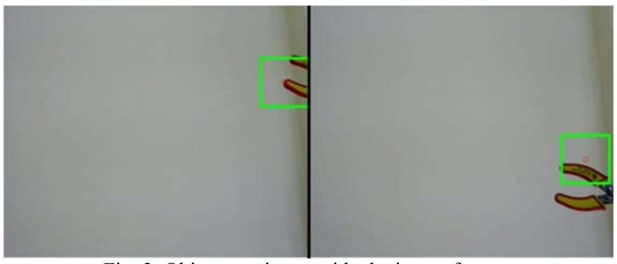
Fig. 2. Object moving outside the image frame.

In Fig. 2(a) the object moves out of the frame. The algorithm tracks the object till it has completely gone outside the frame. Once the object has gone outside the frame the particles get scattered due because no definite goal state has been found out by any particle. In Fig. 2(b) when the object returns in the image frame global best value is updated and the all the particles start converging on the object.

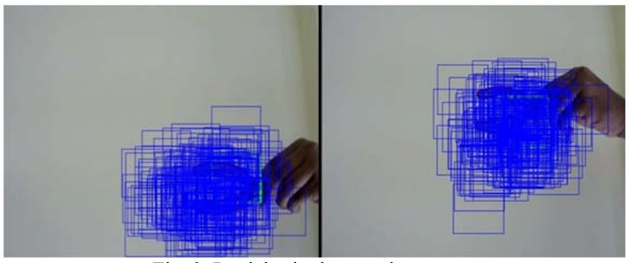
Fig. 3. Particles in the search space.

In Fig. 3 we can see the particles randomly distributed around the object. These particles move along with the object according to the global best and the local best values of fitness.

In Fig. 1, Fig. 2 and Fig. 3 kalman filter was applied in order to smoothen out the output generated by the PSO algorithm. Kalman filter also estimates the path of the object whose co-ordinates can be easily retrieved from the code.

VII. CONCLUSIONS

The experimental result of the proposed algorithm gives robust output. This system is rotation invariant and hence tracks the objects irrespective of its orientation.

Since this system is highly optimized and platform independent it can be executed on any embedded system.

Multiple target object histograms can be incorporated in the proposed algorithm in order to track multiple objects in real time. Since the proposed algorithm is an application of an optimization technique, the system does not introduce any lag.

ACKNOWLEDGMENT

We would like to take this opportunity to thank Prof. Vijay Ghule for giving us all the help and guidance we needed. We are really grateful to him for his kind support. His valuable suggestions were very helpful.

REFERENCES

[1] Kennedy and R.C. Eberhart, Particle Swarm Optimization, in Proceedings of IEEE International Conference on Neural Networks, 1942-1948, 1995.

[2] M. Mason, Z. Duric , Using Histograms to Detect and Track Objects, *in Color Video, in Proceedings of IEEE30th Applied Imagery Pattern Recognition Workshop, ISBN 0-7695-1245-3.*

[3] Chen-Chien Hsu, Guo-Tang Dai, Multiple Object Tracking using Particle Swarm Optimization, *in International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering Vol:6, No:8, 2012.*

[4] T. Kailath, The Divergence and Bhattacharyya Distance Measures, *in Signal Selection, IEEE Trans. Commune. Tech., COM-15:52-60, 1967.*

[5] Yuhua Zheng and Yan Meng, Object Detection and Tracking using Bayes-Constrained Particle Swarm Optimization, *in Computer Vision Research Progress, ISBN 978-1-60021-992-4.*

[6] X. Xiao, E.R. Dow, R.C. Eberhart, Z. Ben miled, and R.J. Oppelt, Gene Clustering using Self-Organizing Maps and Particle Swarm Optimization, *in Proceedings of the Second IEEE International Workshop on High Performance Computational Biology, pp. 10 2003.*

[7] Y. Owechko, S. Medasani, and N. Srinivasa, "Classifier Swarms for Human Detection in Infrared Imagery," *in 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04), Vol. 8, pp. 121, 2004.*

[8] R. E. Kalman, A new approach to linear filtering and prediction problems, *J. Basic Eng., vol. 82, no. 1, pp. 35–45, Mar. 1960.*