Kalman Filter and Hungarian Matching

Author: Hy Truong Son

TTIC 31170 - Planning, Learning, and Estimation for Robotics and Artificial Intelligence

The University of Chicago

Chicago, June 2017

Contents

This paper aims to solve the problem of multi-object tracking in Computer Vision by Kalman filter and Hungarian matching algorithm on bipartite graph. We experimented our method with a synthetic dataset that simulates robotic soccer. The program is implemented in Java programming language with Graphical User Interface.

Color detection

function Color Detection (Image $\mathcal{I}$, Color classification f)

    for $i = 1 \to n$:

        for $j = 1 \to m$:

            $\hat{\mathcal{I}}(i,j) \leftarrow f(\mathcal{I}(i,j))$

        end for

    end for

    return $\hat{\mathcal{I}}$

end function

Breadth-First Search - Part 1

function Object Detection (Image $\hat{\mathcal{I}}$)

Initialize a mark that checks if a pixel is visited by BFS

$M : [n] \times [m] \to \{T, F\}$

$M(x, y) \leftarrow F$, $\forall (x, y) \in [n] \times [m]$

Initilize the number of objects: $\mathcal{O} \leftarrow 0$

Brute-force each pixel in the image

for $x = 1 \to n$:

    for $y = 1 \to m$:

        if $M(x, y) = F$:

            $\mathcal{O} \leftarrow \mathcal{O} + 1$

            BFS $(\hat{\mathcal{I}}, (x, y))$

        end if

    end for

end for

end function

Breadth-First Search - Part 2

function BFS (Image $\hat{\mathcal{I}}$, Pixel (x_0, y_0))

01. *Initialize a queue of pixels*
02. $\mathcal{Q} \leftarrow \emptyset$
03. *Add the first pixel into the queue*
04. $\mathcal{Q} \leftarrow \{(x_0, y_0)\}$ and $M(x_0, y_0) \leftarrow T$
05. *The color of this object*
06. $c \leftarrow \mathcal{I}(x_0, y_0)$
07. *Search for all pixels in the connected component*

Breadth-First Search - Part 3

while $\mathcal{Q} \neq \emptyset$:

Pop a pixel (i, j) from $\mathcal{Q}$

for $\Delta_x = -1 \to 1$:

for $\Delta_y = -1 \to 1$:

$x \leftarrow i + \Delta_x$

$y \leftarrow j + \Delta_y$

if $1 \leq x \leq n$, $1 \leq y \leq m$, $\mathcal{I}(x, y) = c$, $M(x, y) = F$:

$M(x, y) \leftarrow T$

$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(x, y)\}$

end if

end for

end for

end while

All pixels that has been in the queue belongs to this object

end function

Random Acceleration Model - Part 1

Consider N objects moving in a 2D plane. Let $x_t^{(i)}$ and $y_t^{(i)}$ be the horizontal and vertical locations of object $i \in \{1, .., N\}$ at time t, and $\Delta x_t^{(i)}$ and $\Delta y_t^{(i)}$ be the corresponding velocity. We can represent this as a state vector $\boldsymbol{x}_t \in \Re^{4N}$ as follows:

$$\boldsymbol{x}_t^T = \begin{bmatrix} x_t^{(1)} y_t^{(1)} & \dots & x_t^{(N)} y_t^{(N)} & \Delta x_t^{(1)} \Delta y_t^{(1)} & \dots & \Delta x_t^{(N)} \Delta y_t^{(N)} \end{bmatrix}$$

Random Acceleration Model - Part 2

$$\boldsymbol{x}_t = A_t \boldsymbol{x}_{t-1} + \boldsymbol{\epsilon}_t$$

$$\begin{bmatrix} x_t^{(1)} \\ y_t^{(1)} \\ \vdots \\ \Delta x_t^{(1)} \\ \Delta y_t^{(1)} \\ \vdots \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_{t-1}^{(1)} \\ y_{t-1}^{(1)} \\ \vdots \\ \Delta x_{t-1}^{(1)} \\ \Delta y_{t-1}^{(1)} \\ \vdots \end{bmatrix} + \boldsymbol{\epsilon}_t$$

where I is the identity matrix of size $2N \times 2N$ and

$$A_{11} = I \quad A_{12} = \Delta \cdot I \quad A_{21} = I \quad A_{22} = I$$

and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, R)$ is the system noise, and Δ is the sampling period.

Random Acceleration Model - Part 3

Suppose that we can observe the location of the object but not its velocity. Let $\boldsymbol{z}_t \in \Re^{2N}$ represent our observations, which we assume is subject to Gaussian noise.

$$\boldsymbol{z}_t = C_t \boldsymbol{x}_t + \boldsymbol{\delta}_t$$

$$\begin{bmatrix} \hat{x}_t^{(1)} \\ \hat{y}_t^{(1)} \\ \vdots \end{bmatrix} = \begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} x_t^{(1)} \\ y_t^{(1)} \\ \vdots \\ \Delta x_t^{(1)} \\ \Delta y_t^{(1)} \\ \vdots \end{bmatrix} + \boldsymbol{\delta}_t$$

where $C_1 = I$, $C_2 = \mathbf{0}^{2N \times 2N}$ and $\boldsymbol{\delta}_t \sim \mathcal{N}(\mathbf{0}, Q)$ is the measurement noise.

Kalman Filter Algorithm

function Kalman Filter

$\mu_0 \leftarrow \boldsymbol{x}_0$

$\Sigma_0 \leftarrow I$

for $t = 1 \to \infty$:

Prediction

$\bar{\mu}_t \leftarrow A_t \mu_{t-1}$

$\bar{\Sigma}_t \leftarrow A_t \Sigma_{t-1} A_t^T + R_t$

Get a new measurement $\boldsymbol{z}_t$

Update

$K_t \leftarrow \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$

$\mu_t \leftarrow \bar{\mu}_t + K_t(\boldsymbol{z}_t - C_t \bar{\mu}_t)$

$\Sigma_t \leftarrow (I - K_t C_t)\bar{\Sigma}_t$

$\boldsymbol{x}_t \leftarrow \mu_t$

end for

end function

Maximum Matching on Bipartite Graph

We need the Hungarian matching algorithm on bipartite graph. The bipartite graph has two sides X and Y. Each vertex of side X represents for an object position. Each vertex of side Y represents for a measurement. The cost of matching a vertex of X to another vertex of Y is the Euclidean distance between the corresponding object position and the corresponding measurement.

In the case that we do not have enough measurement, $|X| > |Y|$, we add some more virtual vertices to Y and the Euclidean distances from X to these new vertices are set to be infinity. For simplicity, we only consider the case when $|X| = |Y|$.

Maximum Flow Minimum Cost

We can solve the Hungarian matching problem efficiently by Kuhn-Munkres algorithm. In this paper, we introduce another way of solving it by Maximum Flow Minimum Cost. First of all, we construct our flow graph by as following:

- Create a virtual source vertex s
- Create a virtual sink vertex t
- Connect s to all vertices in X with cost 0 and edge capacity 1
- Connect all vertices in Y to t with cost 0 and edge capacity 1
- Connect all vertices in X to all vertices in Y with the cost as the Euclidean distance and edge capacity 1

Time complexity

- Bellman-Ford algorithm with Rounded-Queue data structure: $O(|V|.|E|)$. In practice, it will run much **faster** than the original version.
- Number of iterations in Ford-Fulkerson algorithm: $O(|V|.|E|)$.
- Total complexity: $O(|V|^2|E|^2)$.

Hy T. Son

Thank you very much for your attention!