Multi Object Tracker Using Kalman Filter & Hungarian Algorithm (DSCI-6008 Final Project)

Author: Srini Ananthakrishnan

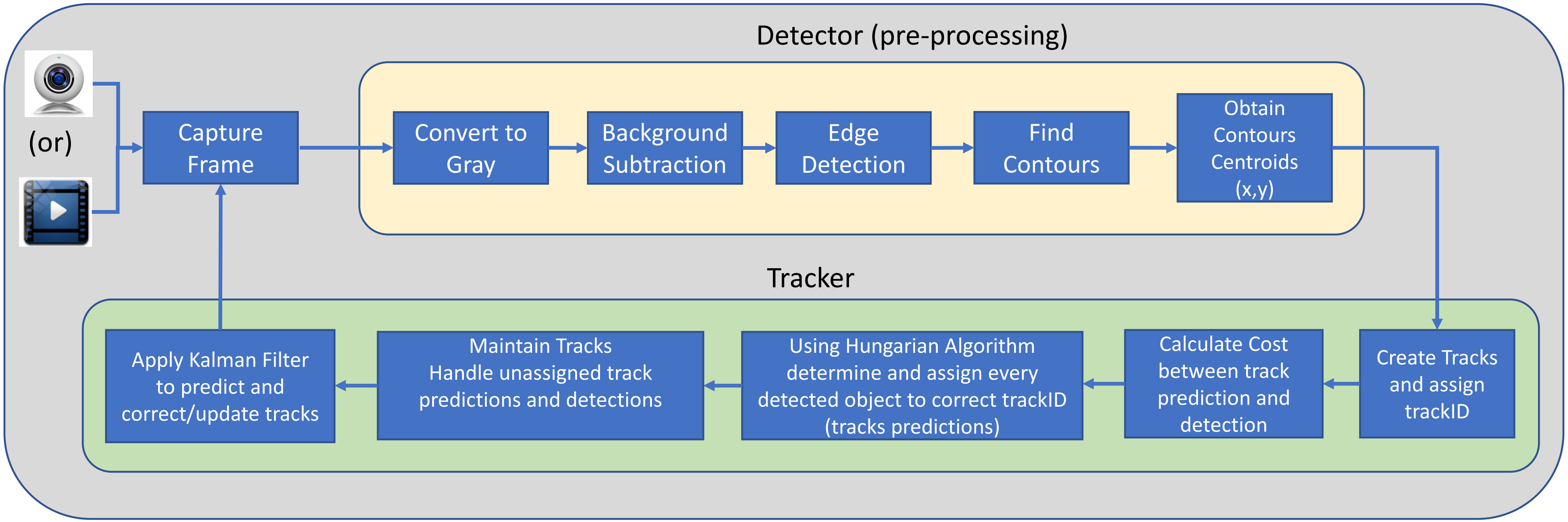
Date: 07/17/2017

Agenda

- Object Tracking Design
- Pipeline Outputs
- Kalman Filter Algorithm
- Pre-requisites
- References
- Q & A

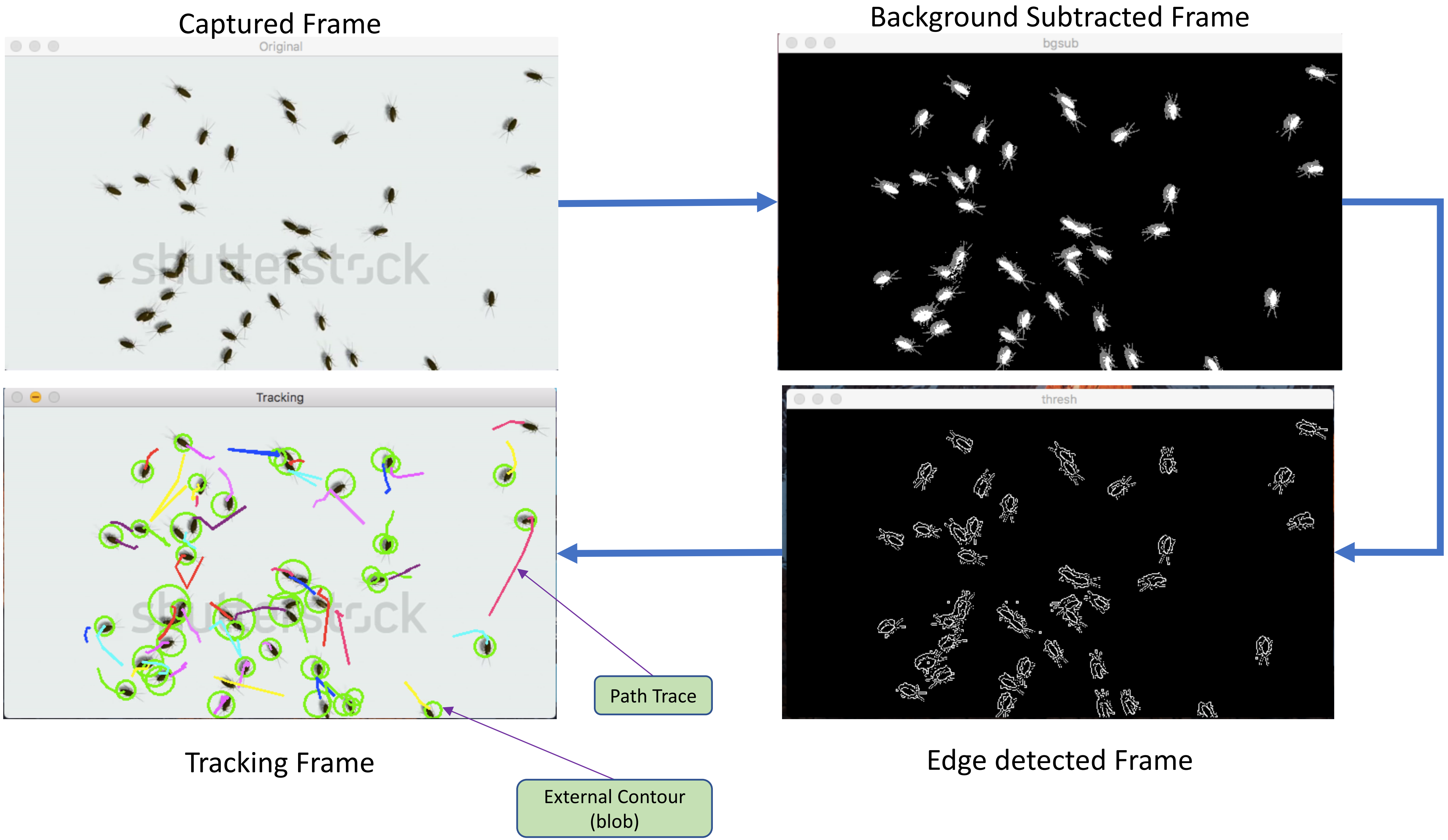
Object Tracking Design

Object Tracking Pipeline



Notes:
*For Hungarian Algorithm SciPy linear_sum_assignment
*For pre-processing used OpenCV libraries
*Kalman Filter is implemented based on various online resources like wikipediea etc.,

Pipeline Output



Kalman Filter Algorithm

- Prediction:
  - Predict state vector u and variance of uncertainty P (covariance)

$u'_{k|k-1} = Fu'_{k-1|k-1}$

$P_{k|k-1} = FP_{k-1|k-1} F^T + Q$

Where,

u: previous state vector

P: previous covariance matrix

F: state transition matrix (captures state transition from one time step to another)

Q: process noise matrix

Above estimation is recursive for each frame and finds best estimate of final state.

Kalman Filter Algorithm (cont.,)

- Correction or Update:
  - Correct or update state vector u and variance of uncertainty P (covariance).

$$C = AP_{k|k-1} A^T + R$$

$$K_k = P_{k|k-1} A^T C^{-1}$$

$$u'_{k|k} = u'_{k|k-1} + K_k \ (b_k - Au'_{k|k-1})$$

$$P_{k|k} = P_{k|k-1} - K_k CK^T$$

Where,
u: predicted state vector
A: matrix in observation equations
b: vector of observations
P: predicted covariance matrix
Q: process noise matrix (or covariance matrix of system error)
K: Kalman Gain matrix (based on recursive least squares)
C: covariance weighting matrix (with variance along diagonal)
R: observation noise matrix

Example of calculating u':

$u'_{100} = u'_{99} + 1/100 \ (b_{100} - u'_{99})$

$u'_{new} = u'_{old} + K \ (b_{new} - u'_{old})$ which is linear combination

Pre-requisite

- Python2.7
- Numpy
- SciPy
- Opencv 3.0 for Python - Installation

References

- Excellent MATLAB tutorial by Student Dave on object tracking
- OpenCV Tutorial: Multiple Object Tracking in Real Time by Kyle Hounslow
- Kalman Filter

Thank You