**Human pose estimation project work**

**Tutorial 1.1 Assignment**

Mukul C Yadav

mchand.yadav@ufl.edu

UFID: 7585 9623

**Task 1**

**Canny Edge Detection snippets:**

A large brick building

Description automatically generated

**Original image**

A picture containing black, white, computer

Description automatically generated

**High threshold canny edge detection**

A picture containing building, photo, sitting, sign

Description automatically generated

**Low threshold canny edge detection**

**New concepts learned:**

* Sobel filter-based edge detection
* Gaussian filter implementation

The given tutorial assignment uses Sobel flavored gaussian kernel based convolutional filter implementation to perform canny edge detection with user configurable threshold levels.

**Resources:**

* [Online tutorial code URL](https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html)

**Source Code:**



**Task 2**

**Object tracking snippets:**

A person standing on a beach

Description automatically generated

**Object tracking using Boosting tracker**

A picture containing person, man, cellphone, looking

Description automatically generated

**Object tracking using CSRT tracker**

A person standing in front of a beach

Description automatically generated

**Object tracking using KCF tracker with known intermittent tracking error**

A person looking at the camera

Description automatically generated

**Object tracking using MedianFlow tracker with known intermittent tracking error**

A screen shot of a person

Description automatically generated

**Object tracking using MIL tracker**

A person standing on a beach

Description automatically generated

**Object tracking using Mosse tracker with known intermittent tracking error**

A person that is standing in the dark

Description automatically generated

**Object tracking using TLD tracker**

**GOTURN Tracker was still unstable in OpenCV 3.4.10**

**New concepts learned:**

* Object tracking flow with OpenCV API on video feeds with different tracker type algorithms

The given tutorial assignment involved tracking objects in a video with 8 different type of algorithms to follow through moving object, namely,

* Boosting,
* MIL(multiple instance learning),
* KCF(Kernelized correlation filter),
* TLD(tracking, learning & detection),
* GOTURN(Generic Object Tracking Using Regression Networks), **[Unstable on 3.4.10]**
* MOSSE(Minimum Output Sum of Squared Error),
* CSRT(Discriminative Correlation Filter with Channel and Spatial Reliability), and
* Median flow.

Static image snippets given above illustrate performance for each of these given algorithms used for object tracking.

**Source Code:**

1. #include <opencv2/opencv.hpp>
2. #include <opencv2/tracking.hpp>
3. #include <opencv2/core/ocl.hpp>
4. #include "ObjectTracking.h"
6. **using** **namespace** cv;
7. **using** **namespace** std;
9. // Convert to string
10. #define SSTR( x ) static\_cast< std::ostringstream & >( \
11. ( std::ostringstream() << std::dec << x ) ).str()
13. **int** main(**int** argc, **char** \*\*argv)
14. {
15. // List of tracker types in OpenCV 3.4.1
16. string trackerTypes[8] = { "BOOSTING", "MIL", "KCF","TLD", "GOTURN", "MOSSE", "CSRT","MEDIANFLOW" };
17. // vector <string> trackerTypes(types, std::end(types));
18. **bool** retflag;
19. **int** retval;
20. **for** (**int** i = 0; i < 8; ++i) {
21. // Create a tracker
22. string trackerType = trackerTypes[i];
24. Ptr<Tracker> tracker;
26. #if (CV\_MINOR\_VERSION < 3)
27. {
28. tracker = Tracker::create(trackerType);
29. }
30. #else
31. {
32. **if** (trackerType == "BOOSTING")
33. tracker = TrackerBoosting::create();
34. **if** (trackerType == "MIL")
35. tracker = TrackerMIL::create();
36. **if** (trackerType == "KCF")
37. tracker = TrackerKCF::create();
38. **if** (trackerType == "TLD")
39. tracker = TrackerTLD::create();
40. **if** (trackerType == "MEDIANFLOW")
41. tracker = TrackerMedianFlow::create();
42. **if** (trackerType == "GOTURN")
43. tracker = TrackerGOTURN::create();
44. **if** (trackerType == "MOSSE")
45. tracker = TrackerMOSSE::create();
46. **if** (trackerType == "CSRT")
47. tracker = TrackerCSRT::create();
48. }
49. #endif
51. retval = performObjectDetection(tracker, trackerType, retflag);
52. **if** (retflag) **continue**;
53. **else** **break**;
54. }
55. **return** retval;
56. }
58. **int** performObjectDetection(cv::Ptr<cv::Tracker> &tracker, std::string &trackerType, **bool** &retflag)
59. {
60. retflag = **false**;
62. // Read video
63. VideoCapture video("C:\\Users\\vyomkesh\\Documents\\UF\\S4\\Individual Study\\HumanPoseEstimationProjectWork\\objectTracking\\videos\\final4.mp4");
65. // Exit if video is not opened
66. **if** (!video.isOpened())
67. {
68. cout << "Could not read video file" << endl;
69. **return** 1;
70. }
72. // Read first frame
73. Mat frame;
74. **bool** ok = video.read(frame);
76. // Define initial bounding box
77. Rect2d bbox(287, 23, 86, 320);
79. // Uncomment the line below to select a different bounding box
80. // bbox = selectROI(frame, false);
81. // Display bounding box.
82. rectangle(frame, bbox, Scalar(255, 0, 0), 2, 1);
84. imshow("Tracking", frame);
85. tracker->init(frame, bbox);
87. **while** (video.read(frame))
88. {
89. // Start timer
90. **double** timer = (**double**)getTickCount();
92. // Update the tracking result
93. **bool** ok = tracker->update(frame, bbox);
95. // Calculate Frames per second (FPS)
96. **float** fps = getTickFrequency() / ((**double**)getTickCount() - timer);
98. **if** (ok)
99. {
100. // Tracking success : Draw the tracked object
101. rectangle(frame, bbox, Scalar(255, 0, 0), 2, 1);
102. }
103. **else**
104. {
105. // Tracking failure detected.
106. putText(frame, "Tracking failure detected", Point(100, 80), FONT\_HERSHEY\_SIMPLEX, 0.75, Scalar(0, 0, 255), 2);
107. }
109. // Display tracker type on frame
110. putText(frame, trackerType + " Tracker", Point(100, 20), FONT\_HERSHEY\_SIMPLEX, 0.75, Scalar(50, 170, 50), 2);
112. // Display FPS on frame
113. putText(frame, "FPS : " + SSTR(**int**(fps)), Point(100, 50), FONT\_HERSHEY\_SIMPLEX, 0.75, Scalar(50, 170, 50), 2);
115. // Display frame.
116. imshow("Tracking", frame);
118. // Exit if ESC pressed.
119. **int** k = waitKey(1);
120. **if** (k == 27)
121. {
122. **break**;
123. }
125. }
126. retflag = **true**;
127. **return** 0;
128. }

**Resources used:**

* [**http://www.planetb.ca/syntax-highlight-word**](http://www.planetb.ca/syntax-highlight-word) **(Copy & paste from IE for usage in word)**