

# Temporal Video Tracking

By John Drogo

# Overview

- Problem Statement
- Background
- General Concepts
- Solution Architecture
- Results

# Problem Statement

- Track an object through a video sequence.
  - Occlusion
  - Complex Background
  - Object motion/rotation
  - Multiple targets
  - Noise artifacts
  - Changing features



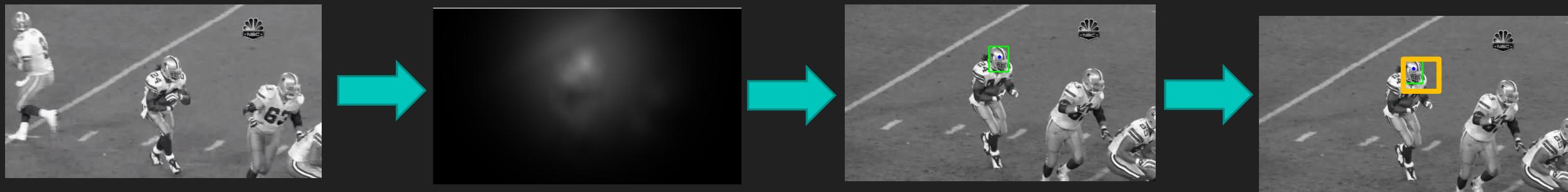
# Background

- Literature survey to understand existing methods
- Widely varied, examined approaches include:
  - Structured Output Tracking with Kernels (STRUCK)
  - Kernelized Correlation Filter (KCF)
  - Minimum Output Sum of Squared Error Correlation Filter (MOSSE)
- Common ground
  - Identify features of interest
  - Predict how the features change
  - Look for the features in new frames

# Solution Architecture

- Three stage approach
  - Feature Detection, Classification, and Prediction
  - An initial truth frame of the target is provided to prime the tracker
  - For subsequence frames the tracker draws a boundary box around the target
- Developed with OpenCV
  - High performance video manipulation framework

# How to Build a Tracker



Input Data

Feature Detection

Classifier

Prediction

# Feature Detection

- Correlation Filter
  - Detects similarity between the video frame and a “kernel”
  - Correlation in the spatial domain becomes multiplication in Fourier domain
- Two kernels are processed
  - A “positive” kernel based on last known match
  - A “negative” kernel based on surrounding known bad tiles
  - Uses the linearity of convolution to combine both into one operation
  - The separation helps the tracker to suppress complex background clutter
- Raw correlation scores are weighted and used for classification

# Feature Classification and Prediction

- The highest score is classified as the target
  - The updated position is fed into the smoother
  - The smoothed location adjust the Gaussian weights in the detector
  - Matches closer to the predicted location are weighted heavier
- Kalman Smoother
  - Estimates the state of the tracked object (position, velocity)
  - A constant acceleration two-dimensional Kalman filter was used
  - The smoother permits prediction of the target's position in the next frame
  - Tunable between smoother and more responsive tracks



Weighted Correlation Score  
(White is Higher)

# Kalman Prediction

- The Kalman filter is used to smooth detected position
  - This position is used to focus in on the predicted search area helping to discern background clutter in complex environments
  - The smoother also permits coasting during track break
  - Prevents track steals when target is partially covered
- The filter is implemented in two stages
  - Predict – Uses the internal state to generate a prediction for the next frame
  - Correct – Updates the internal state based on the observed measurements
- The filter must be calibrated to the expected motion of the scene
  - If the process noise is too low the smoother will not lock
  - Too high and track steals become frequent

# Verification

- Literature survey revealed a tracker testbench
- Several example clips were extracted to validate algorithm
- A set of simpler test clips were used to assist development



# Hello Word – v0.0.0

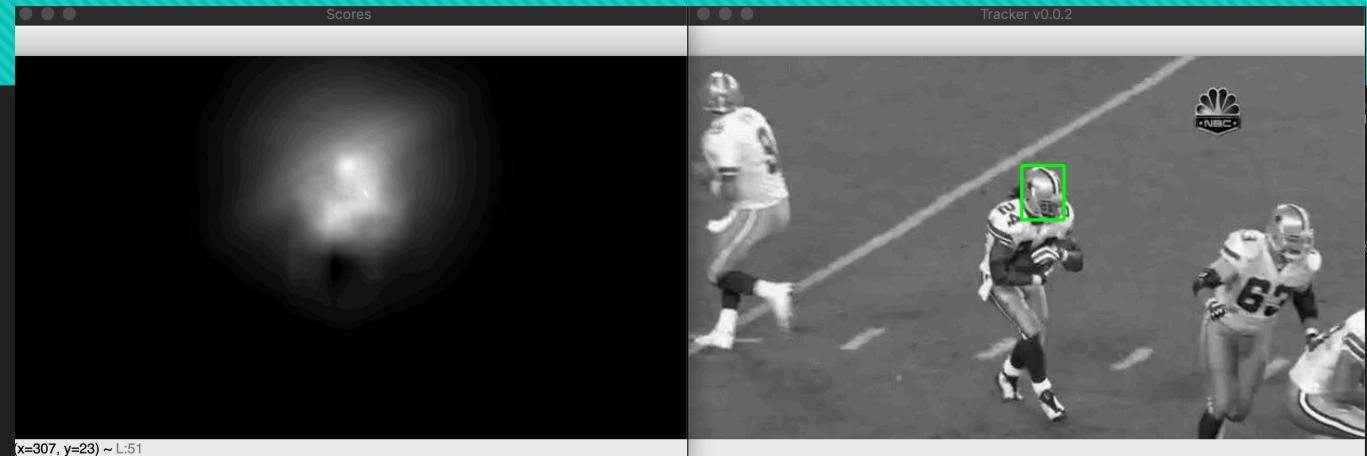
The image shows a Mac OS X desktop environment. The desktop background is a photograph of a bright blue sky with scattered white clouds. In the foreground, there are three windows:

- A terminal window titled "proj — bash — 129x37" showing the command "python main.py". The output shows the script is reading a video file named "ball.mp4".
- A terminal window titled "main.py (~/Documents/SchoolWork/2019/Image/proj) — bash" showing the Python script code.
- A Vim editor window titled "main.py (~/Documents/SchoolWork/2019/Image/proj) - VIM" showing the same Python script code.

```
1 import cv2 as cv
2 import time
3
4 if __name__ == "__main__":
5     capture = cv.VideoCapture(cv.samples.findFileOrKeep("ball.mp4"))
6     if not capture.isOpened():
7         print("Open failure.")
8
9     cv.waitKey(1000)
10    color = (0, 255, 128)
11    size = 2
12    while True:
13        ret, frame = capture.read()
14        if ret is False:
15            print("\nVideo ended")
16            break
17
18        target = (frame > 100).nonzero()
19        target_x = target[1].mean()
20        target_y = target[0].mean()
21        width = target[1].ptp()
22        height = target[0].ptp()
23
24        bounds_first = (int(target_x-width/2), int(target_y-height/2))
25        bounds_second = (int(target_x+width/2), int(target_y+height/2))
26        cv.rectangle(frame, bounds_first, bounds_second, color, size)
27        print(f"\r X: {target_x:7.3f} Y: {target_y:7.3f} W: {width:4} H: {height:4}", end="")
28
29        cv.imshow("Tracker v0.0.0", frame)
30        if cv.waitKey(1) == ord('q'):
31            break
```

# Algorithm Refinements

- Learning Rates
  - Update kernels based on observed samples.
- Kalman Tuning
  - Calibrate process noise and weight distribution.
- Two stage focus
  - Wide area adjusting based on correlation lock.
  - Gain for predicted region.
- Lost lock fallback
  - Auto target re-acquisition



After Kalman Tuning



Before Kalman Tuning

Tracker v0.0.2

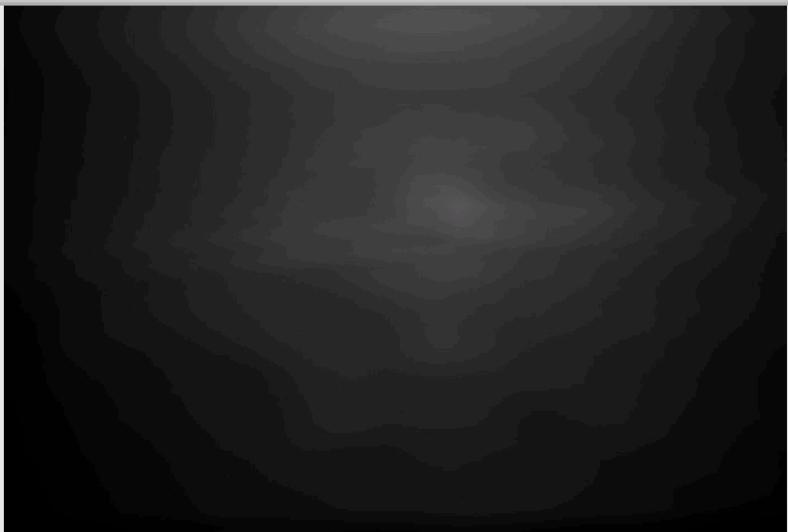


iCloud

clip\_v0.0.0\_small.mp4  
clip\_v0.0.0.mp4

Dec 4,  
Dec 4,

Scores



Positive Kernel



Negative Kernel



```
184      #start_pos = [0, 0, 40, 40]
          ● proj — Python main.py — 62×23
          ○ ~Documents/Scho...
          ○ ...yton main.py ...
          ○ eam — -bash
[.venv] quartz:proj John$ python main.py
X: 203.000 Y: 89.000 W: 38 H: 18 score: 88.44
```

# Summary

- Successful tracking
  - Reasonable performance with common configuration
- Sensitive to camera motion
  - Affects Kalman Estimation
- Profile changes are difficult
  - Faster learning rate makes system susceptible to clutter



# Questions?

- <https://github.com/skhobahi/Kalman-Filter-Object-Tracking/blob/master/Object%20Tracking%20Using%20Kalman%20Filter.pdf>
- <https://www.opencv-srf.com/2018/03/gaussian-blur.html>
- <https://opencvexamples.blogspot.com/2014/01/kalman-filter-implementation-tracking.html>
- [http://www.robots.ox.ac.uk/~joao/publications/henriques\\_tpami2015.pdf](http://www.robots.ox.ac.uk/~joao/publications/henriques_tpami2015.pdf)
- <http://setosa.io/ev/image-kernels/>
- <https://ieeexplore-ieee-org.proxy1.library.jhu.edu/stamp/stamp.jsp?tp=&arnumber=7360205&tag=1>
- [https://www.cs.colostate.edu/~vision/publications/bolme\\_cvpr10.pdf](https://www.cs.colostate.edu/~vision/publications/bolme_cvpr10.pdf)
- [https://faculty.ucmerced.edu/mhyang/papers/cvpr13\\_benchmark.pdf](https://faculty.ucmerced.edu/mhyang/papers/cvpr13_benchmark.pdf)