WPI

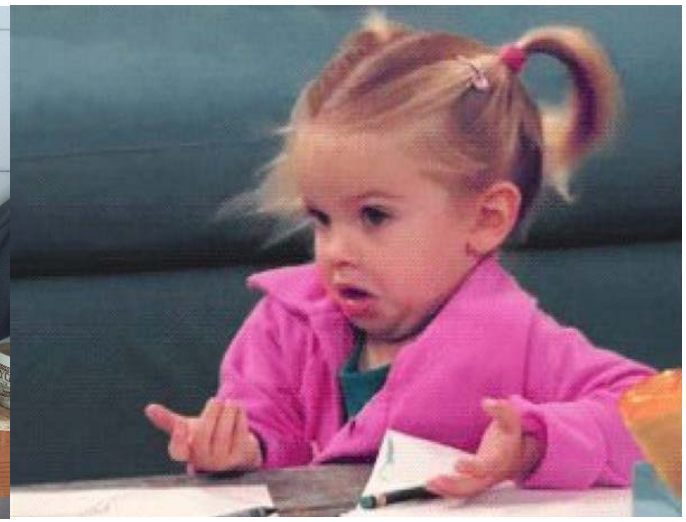
Desktop Object Identification and Tracking

Kevin DuCharme &
Max Li

Introduction

Motivation

The main motivation behind this project is the struggle of finding items on a heavily used desk. On an active workbench, tools and other useful items are frequently getting picked up, used, and placed down somewhere else. While good standard operating procedures will make sure all the parts end up where they're expected to be, it's very easy to quickly lose track of items.



Introduction

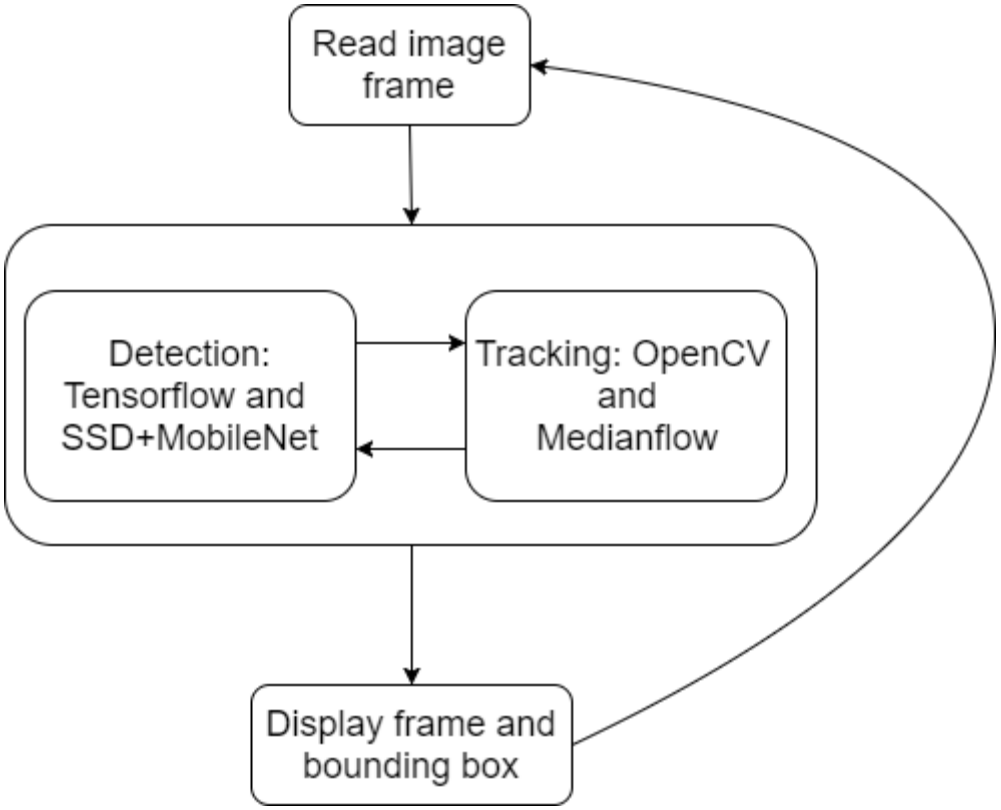
- Starting the framework for a lab assistant robot
- Detect and track pre-trained common workbench items

Desk Camera Configuration

Controlled Environment

- Keep setup simple
- Top-down viewing camera
- Kept well-lit
- Reduced clutter for initial implementation

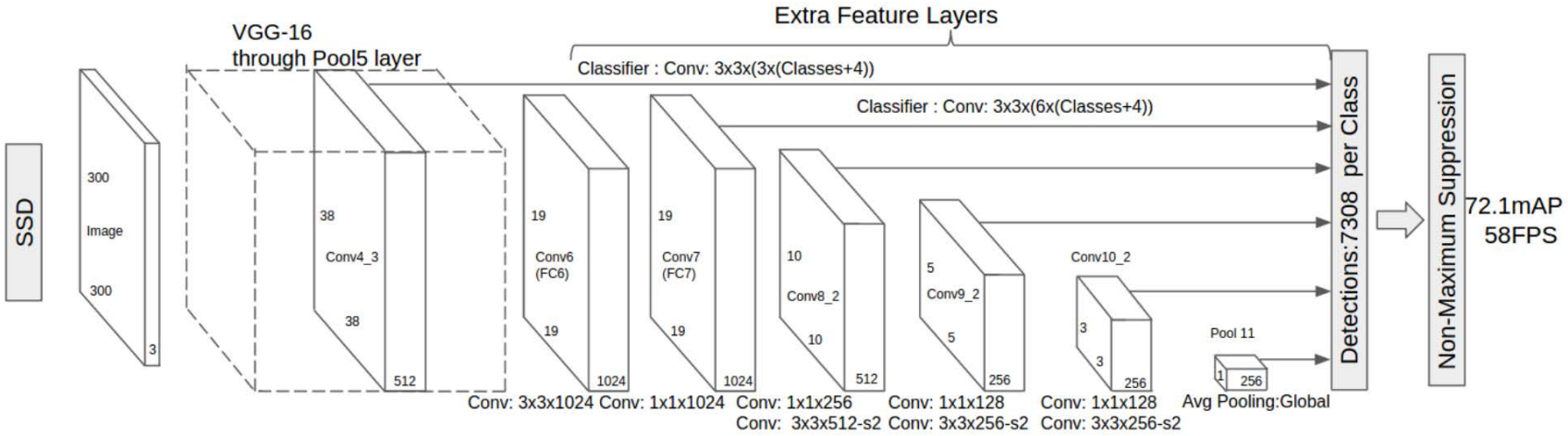
Code Structure



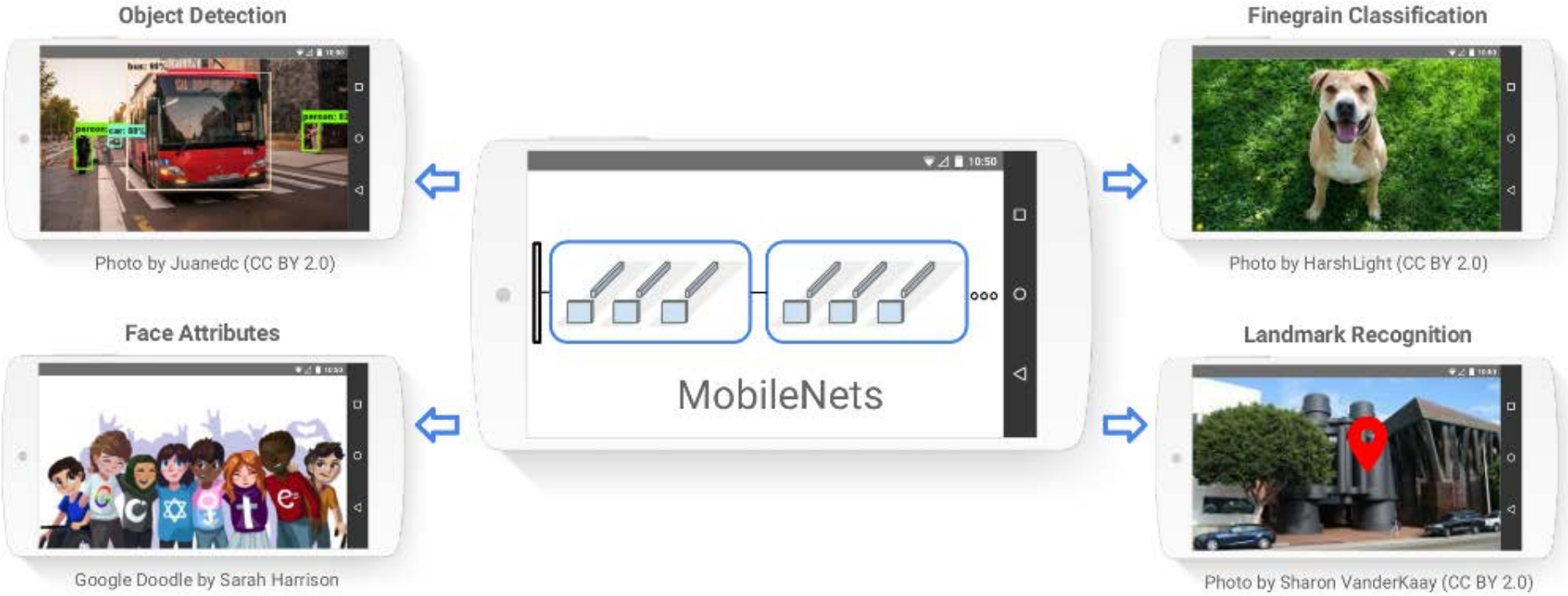
- Python 3.5
- Tensorflow 1.4
- OpenCV 3.3 with extra modules (opencv-contrib)

Object Detection

Single-Shot Detectors

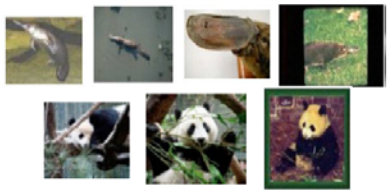


MobileNets

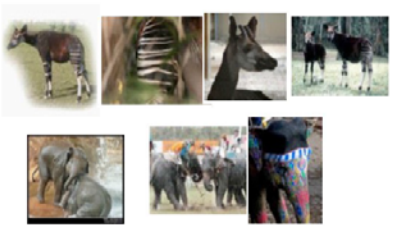


A. G. Howard *et al*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.

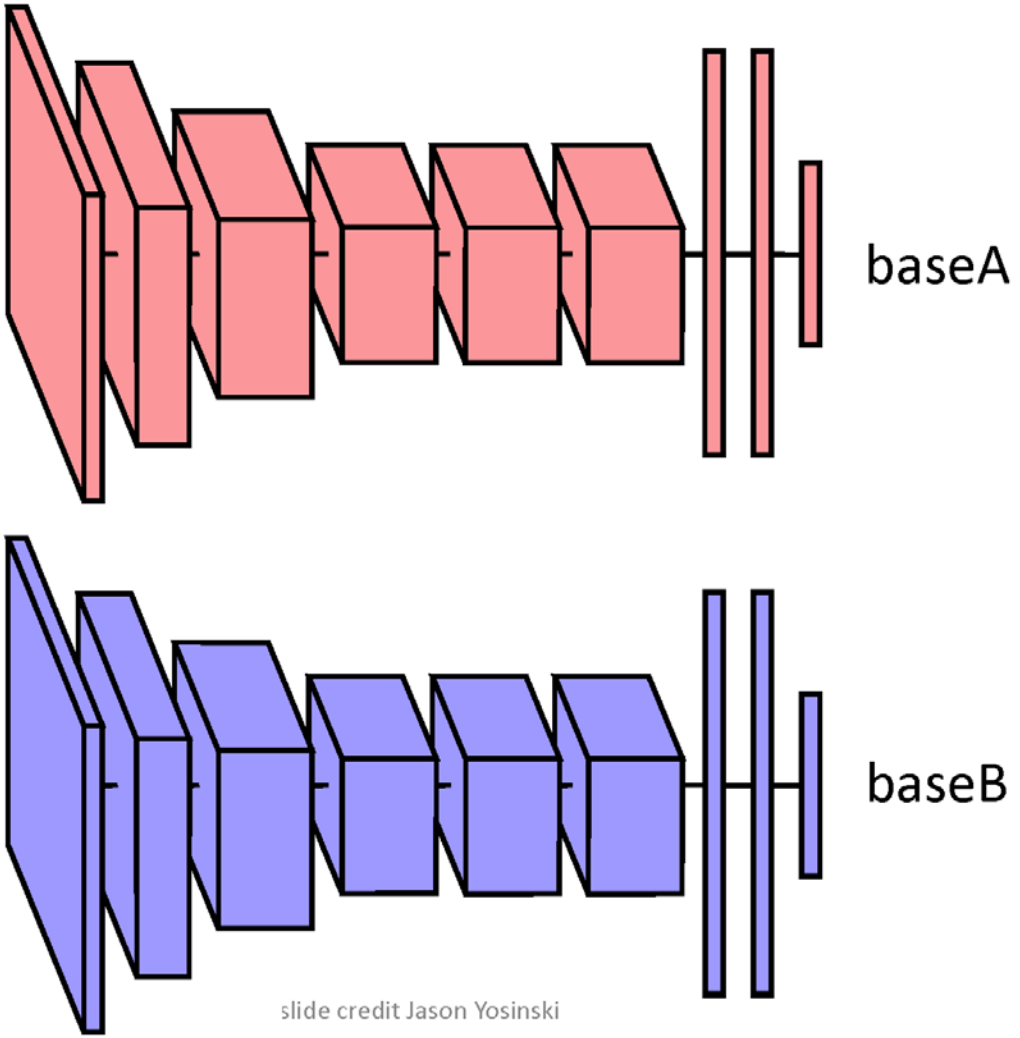
Transfer Learning



A Images



B Images

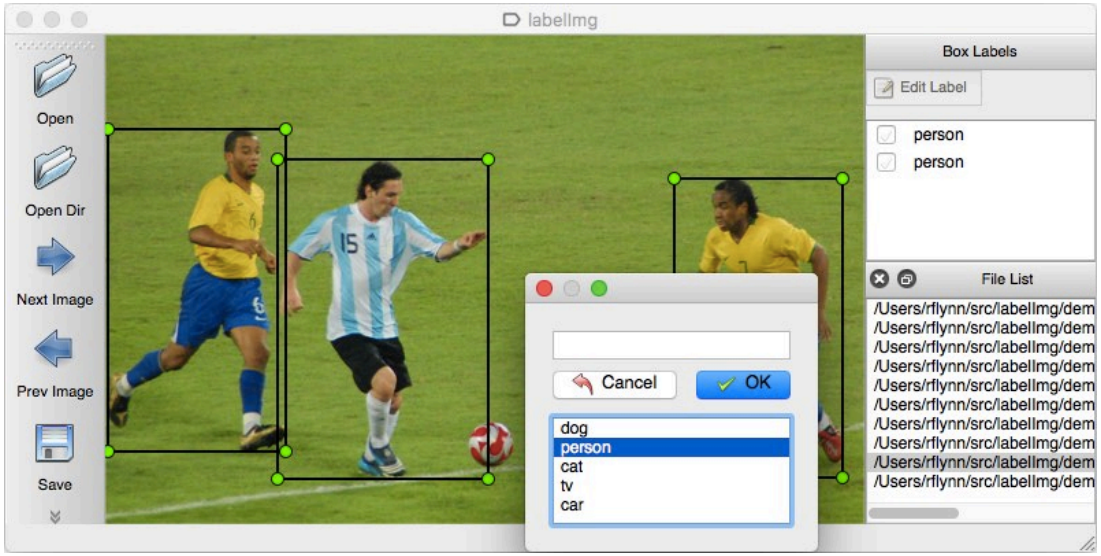


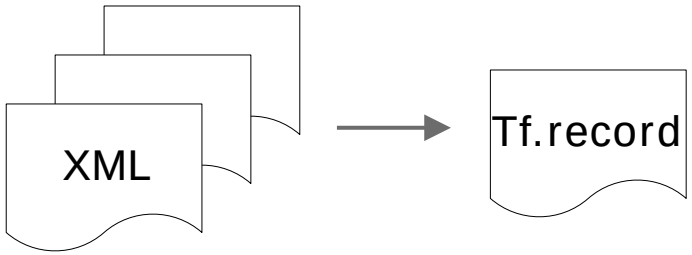
slide credit Jason Yosinski

Gathering Training Set

- The project was limited to three classes: Wrenches, Pliers, and Screwdrivers.
- Each category contains ~150 sample images
- Images are of various resolutions
- Only data augmentation was scaling down high resolution images

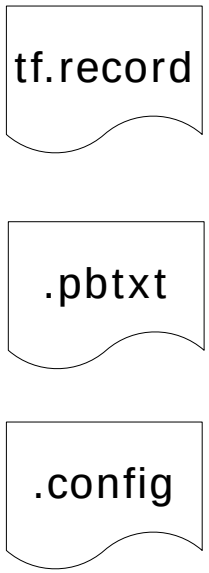
Bounding Box Labels





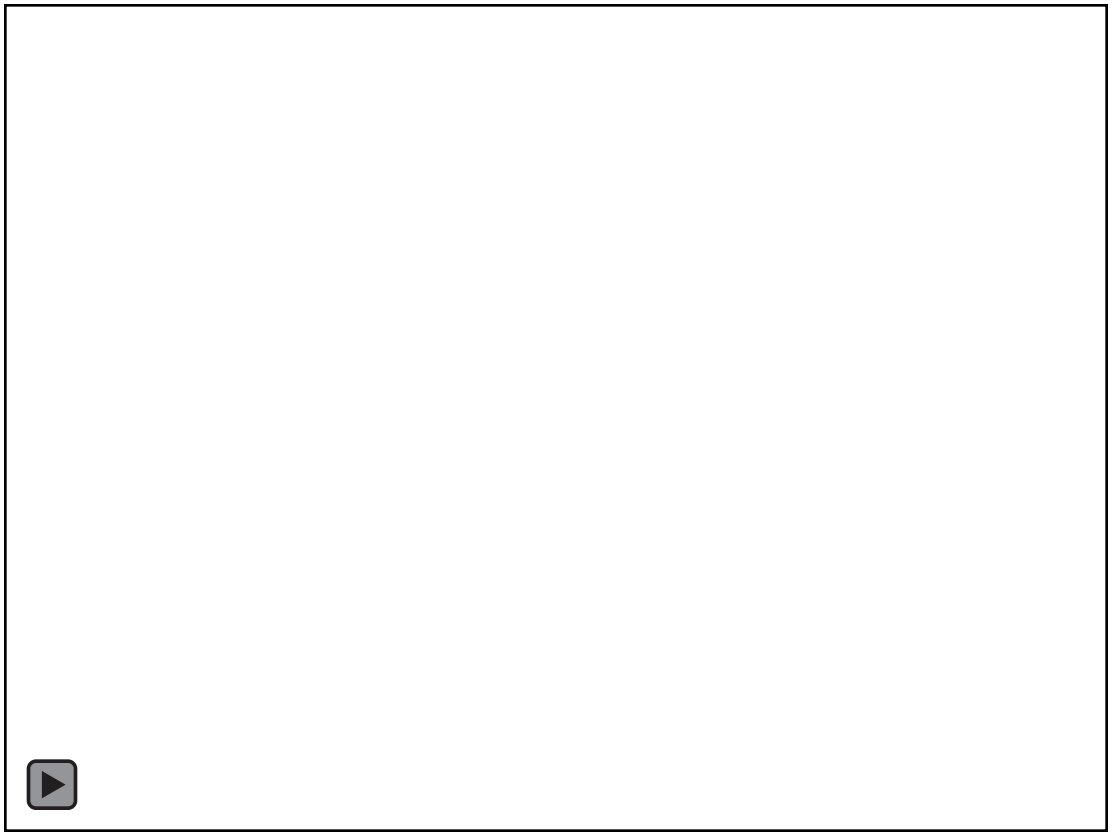
- LabelImg (github/tzutalin)
- Boundary box information was added for each item
- Label was applied to the boundary box
- Resulting XML files are converted to a tensorflow record file.

Training New Model

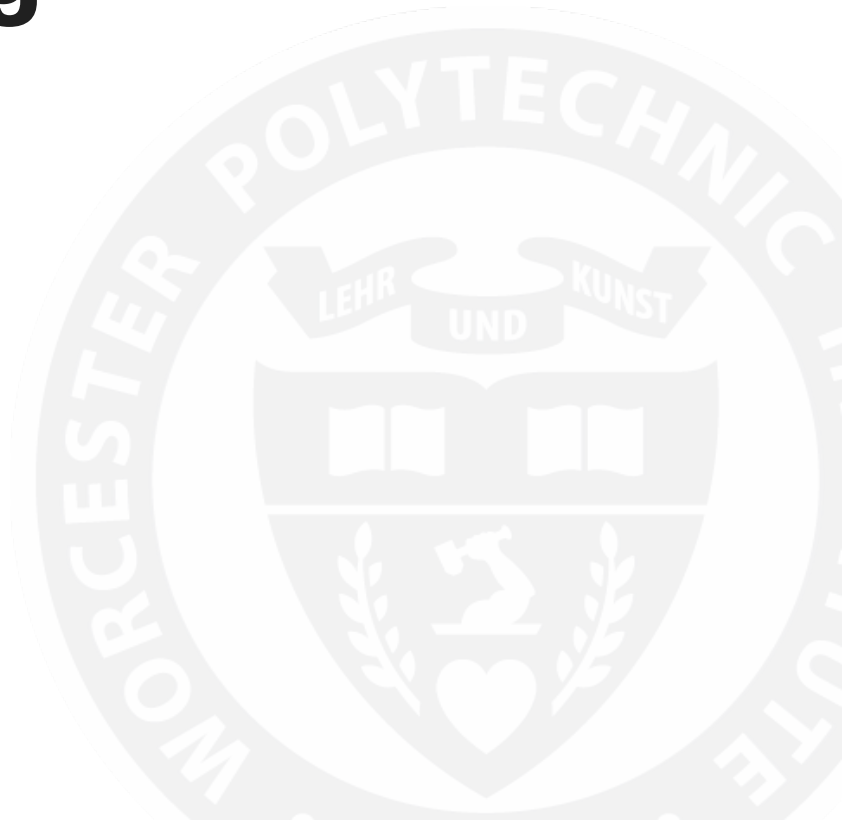


- To train the model with the new data set four files were needed
  - tf.record file for each the training and testing dataset
  - .pbtxt file which maps the id number to a classification label
  - .config which contains the architecture for the neural network
- Configuration file was based off of ssd_mobilenet_v1_pets.config with alterations
  - Changes number of classes from 90 to 3
  - Batch sized changes from 24 to 15
  - Redirected to our tf.record files and label map pb.txt file
- Training ran for 13+ hours and completed 30,857 steps

Real-time Detection (No Tracking)



Object Tracking



What is Object Tracking?

- Object tracking is successively locating a known object in each frame of a video
  - Predefined bounding box is used as input to the tracker
- Tracking is based on combining a motion model and an appearance model
  - Motion model is modeling the location and velocity of the object, like a Kalman filter
  - Appearance model is the model of the object to be tracked.
    - Frequently a classifier that is trained online

Why Track?

- Tracking algorithms are usually faster than detection algorithms
  - + Detection algorithms have to start with no prior information about an object.
  - + Tracking algorithms know the previous location and velocity of an object.
  - – Tracking algorithms start to drift over time from accumulating errors
- Tracking can frequently handle occlusion better
- Tracking preserves the identity of an object

Tracking Methods

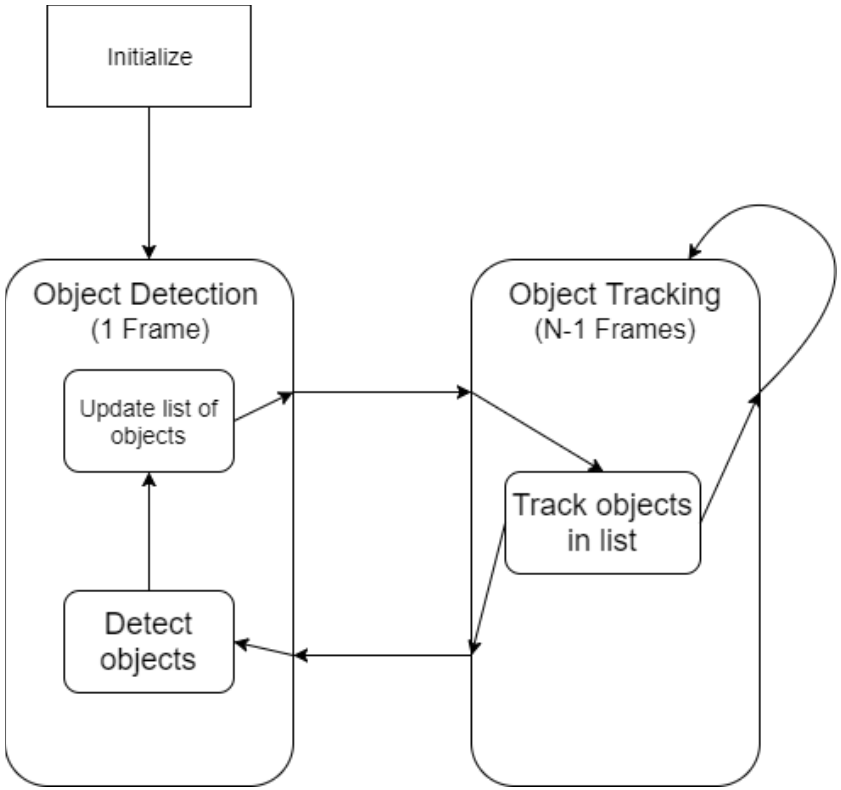
- OpenCV 3.3 has 6 types of online trackers:
- AdaBoost-based:
  - Boosting: Uses AdaBoost, an ensemble learning method, to estimate the location.
  - Multiple Instance Learning (MIL): Similar to Boosting, but looks at a wider range of locations as positive identifications.
  - Kernelized Correlation Filters: Builds on MIL's strategy of creating a large number of overlapping positive identifications.

Tracking Methods, Continued

- Other:
  - **Medianflow:** Looking at a set of points in a bounding box, uses Lucas-Kanade optical flow estimator for a motion model, filtering out bad results. Remaining boxes used to get the final bounding box.
  - Tracking, Learning, Detection (TLD): A different tracking algorithm is combined with a Fern Ensemble classifier, which updates the tracker when it veers off-track
  - Goturn: Uses neural networks pretrained on a variety of labeled tracking videos to track novel objects

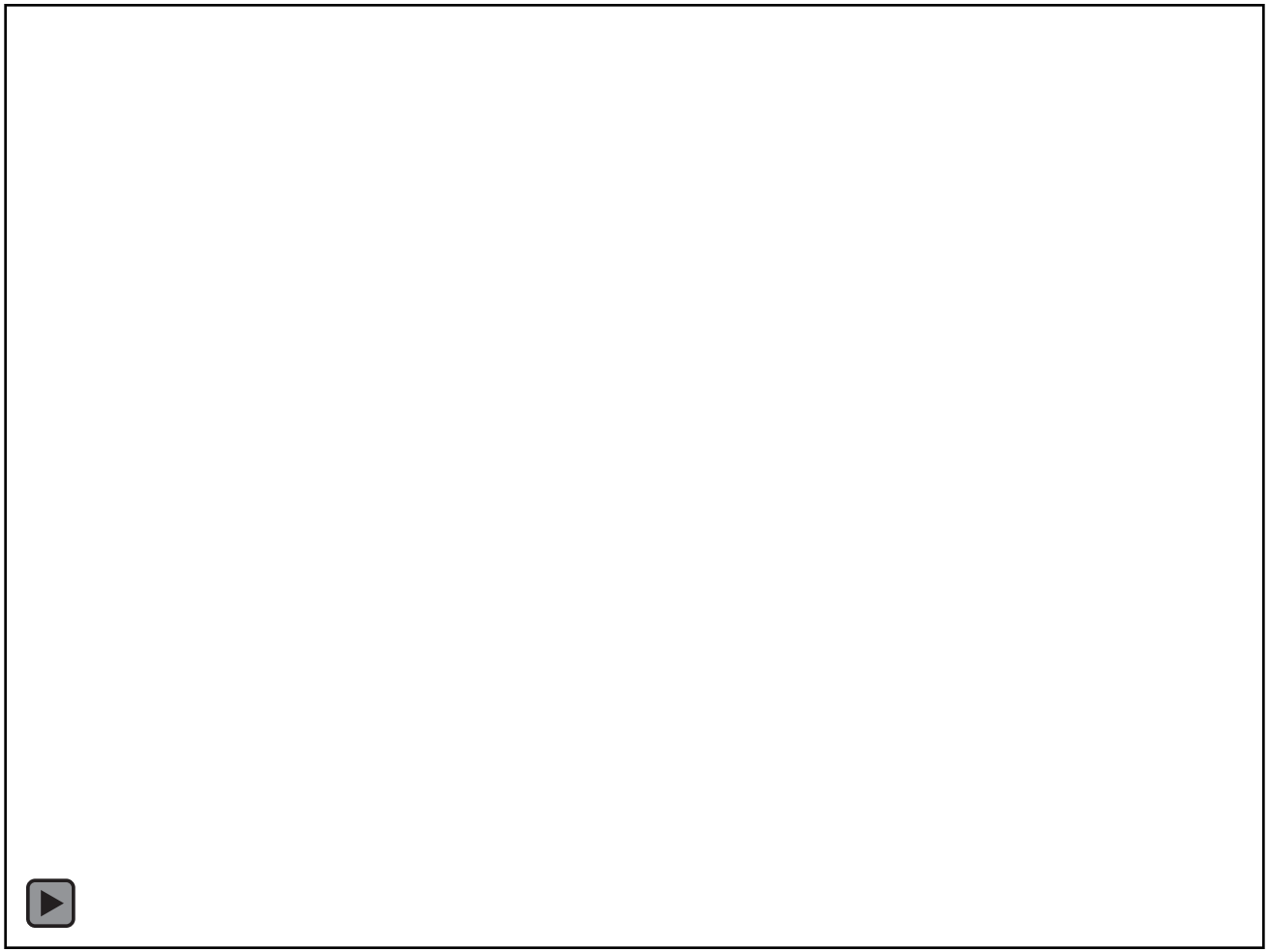
Incorporating Tracking

- With OpenCV, tracking is fairly straightforward.
- Preservation of ID is more complicated.
  - In detection phase, compare detections to list of objects.
  - If object doesn't exist, add to new list.
  - Remove trackers that don't match up with any detected objects.



Results and Conclusions

Real-Time Detection (Tracking)



Results

- Collected and labeled new, tool-specific dataset
- Used transfer learning to modify MobileNet-SSD classifier trained on the COCO dataset, for our tool dataset.
- Implemented real-time object tracking using self-trained model.
  - 50 FPS average
- Incorporated medianflow-based tracking to improve fps and incorporate IDs.
  - 110 FPS average, one object tracked.

Places to Improve

- Detection
  - Artificially grow dataset with data augmentation
    - Apply rotations, translations, flipping, and rescaling
  - Wider variety of training images
  - More quantitative evaluation techniques
    - Precision-recall curve
    - F1 score
- Tracking
  - Utilize more complex methods for tracking
    - Only managed to get KCF, medianflow working
  - Improve ID incrementing logic.
    - A number of parameters to tune that may make for more consistent IDs.

Future Work

- Create a more complete training dataset.
  - More classes of tools
- Explore alternate detection frameworks.
  - Inception V2 + Resnet
- More exploration of tracking methods.
  - Goturn, TLD, KCF in practice
  - Hyperparameter tuning
- Develop a user-facing frontend
  - Allow user to train new tools, train on their own tools.

Any Questions?


WORCESTER POLYTECHNIC INSTITUTE
LEHR UND KUNST
1865