OBJECT DETECTION USING MACHINE LEARNING

Cone finding for the Robo-Magellan (or RoboColumbus) contest

Bob Cook
June 22, 2024

ABOUT ME

Background in professional software development with ~30 years experience

Working for an industry leading cybersecurity firm managing development teams in Canada and Germany

Passionate and persistent tinkerer

Love building robots (when I can)

bob.cook@gmail.com
linkedin.com/in/bob-cook-software-exec/

NOTES

- This is my hobby… please understand I'm:
  - not a professional roboticist
  - not a machine vision expert
  - not a data scientist nor an ML / AI expert (just an enthusiast)
- This talk is about classification (object detection) but not other types of ML / AI
- Unless otherwise noted, all content is my own

TODAY'S TOPICS

- Quick intro to the SRS Robo-Magellan contest
- Introducing Ferdy, my cone finding robot
- Detecting orange cones
  - Classic approaches
  - Machine learning approach
  - Detailed how-to
  - My results + other test results

SRS ROBO-MAGELLAN

Inspired by the DARPA Grand Challenge 2004

A small scale autonomous vehicle race in which robots navigate between predefined start and finish points.

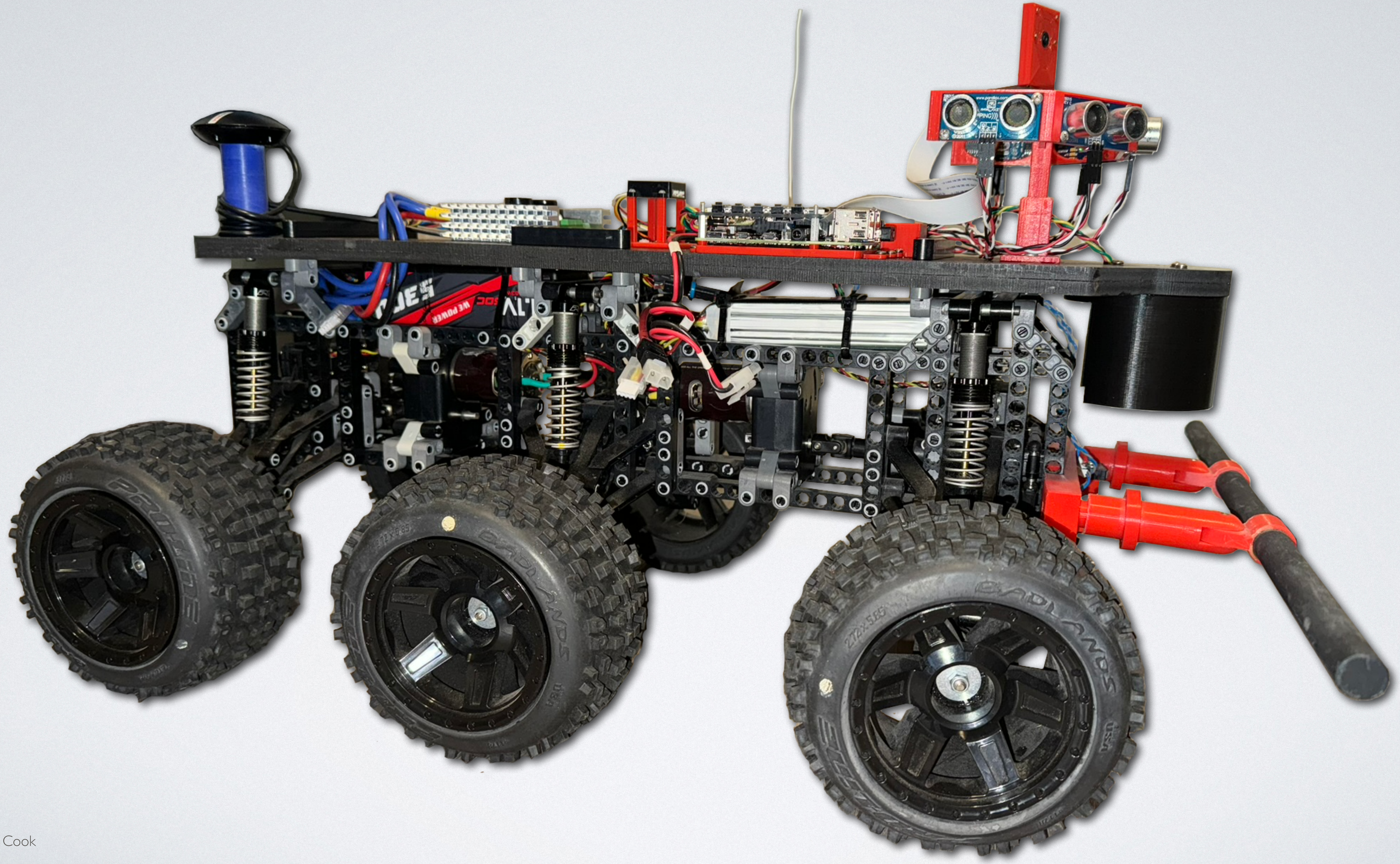
The start and finish points are usually represented as GPS coordinates and marked by orange traffic cones.

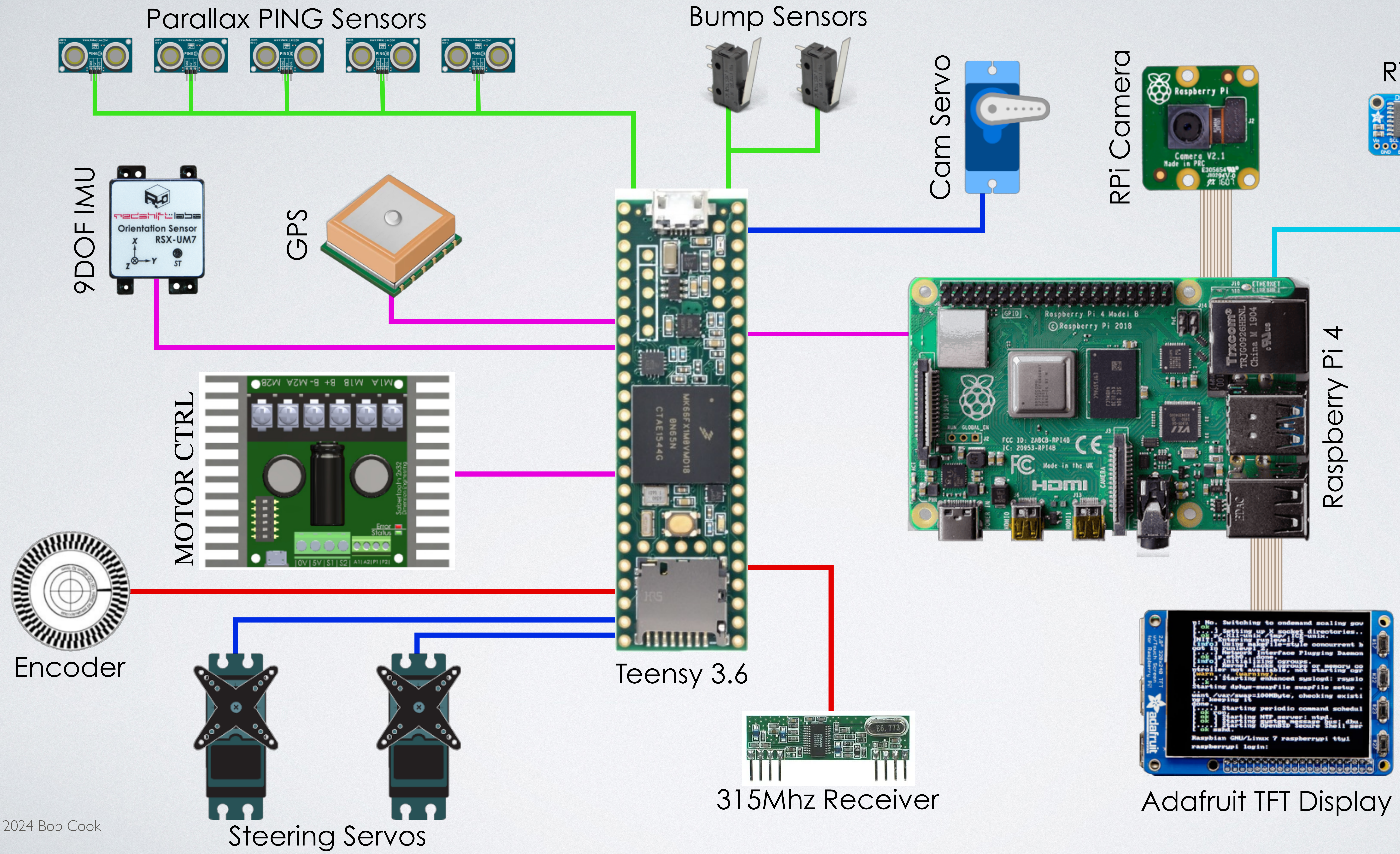
In most versions of the competition there are also optional waypoints that the robot can navigate to in order to earn bonus points.

The race is usually conducted on mixed pedestrian terrain which can include obstacles such as park benches, curbs, trees, bushes, hills, people, etc.

Wikipedia


Robothon 2004




Parallax PING Sensors
Bump Sensors
Cam Servo
RPi Camera
RTC
9DOF IMU
GPS
Orientation Sensor
RSX-UM7
MOTOR CTRL
Raspberry Pi 4
Encoder
Teensy 3.6
Steering Servos
315Mhz Receiver
Adafruit TFT Display

MY OBJECTIVE

- Detect orange cones as defined by the Robo-Magellan contest:
  - 18" orange plastic cones
  - Upright position
  - Terrain may include pavement, dirt, small rocks, grass, hills, gullies, trees, curbs and weeds – cone placement may be in a variety of lighting conditions, backgrounds, etc.

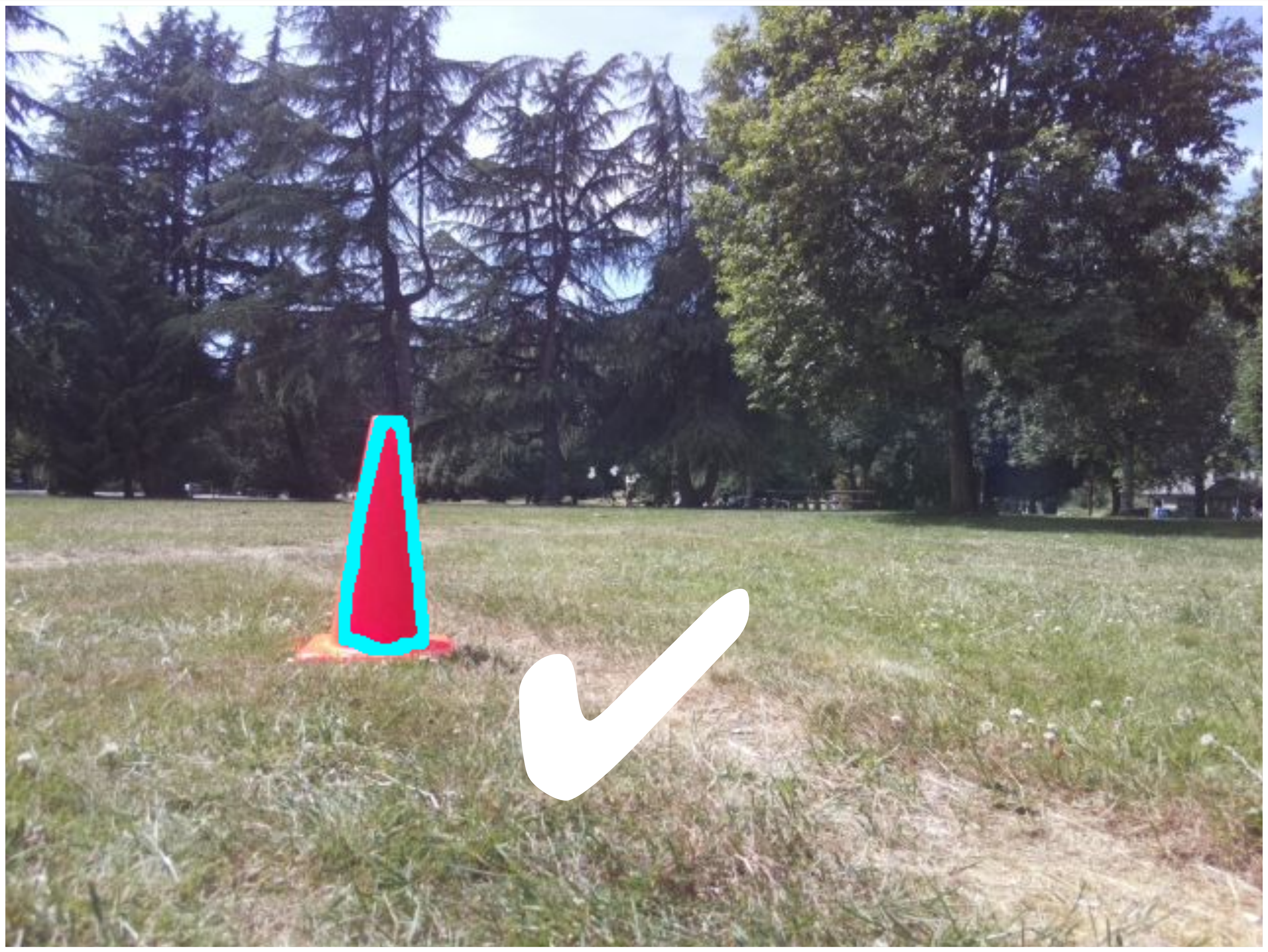



CLASSIC DETECTION TECHNIQUE #1

Look for Blobs of Specific Colors

- Color blob tracking is relatively simple to implement even with small microprocessors
- Find pixels that are vaguely orange, then find adjacent similar pixels
- Add heuristics to filter out false positives e.g. "must be vaguely triangular" or "must have a certain number of pixels/size"
- Many off-the-shelf solutions: AVRcam, CMUcam, PixyCam, OpenMV, etc.
- I've tried most of them… and didn't have success. Why not?

Orange
Blobs



Orange
Blobs

Orange
Blobs

Orange
Blobs

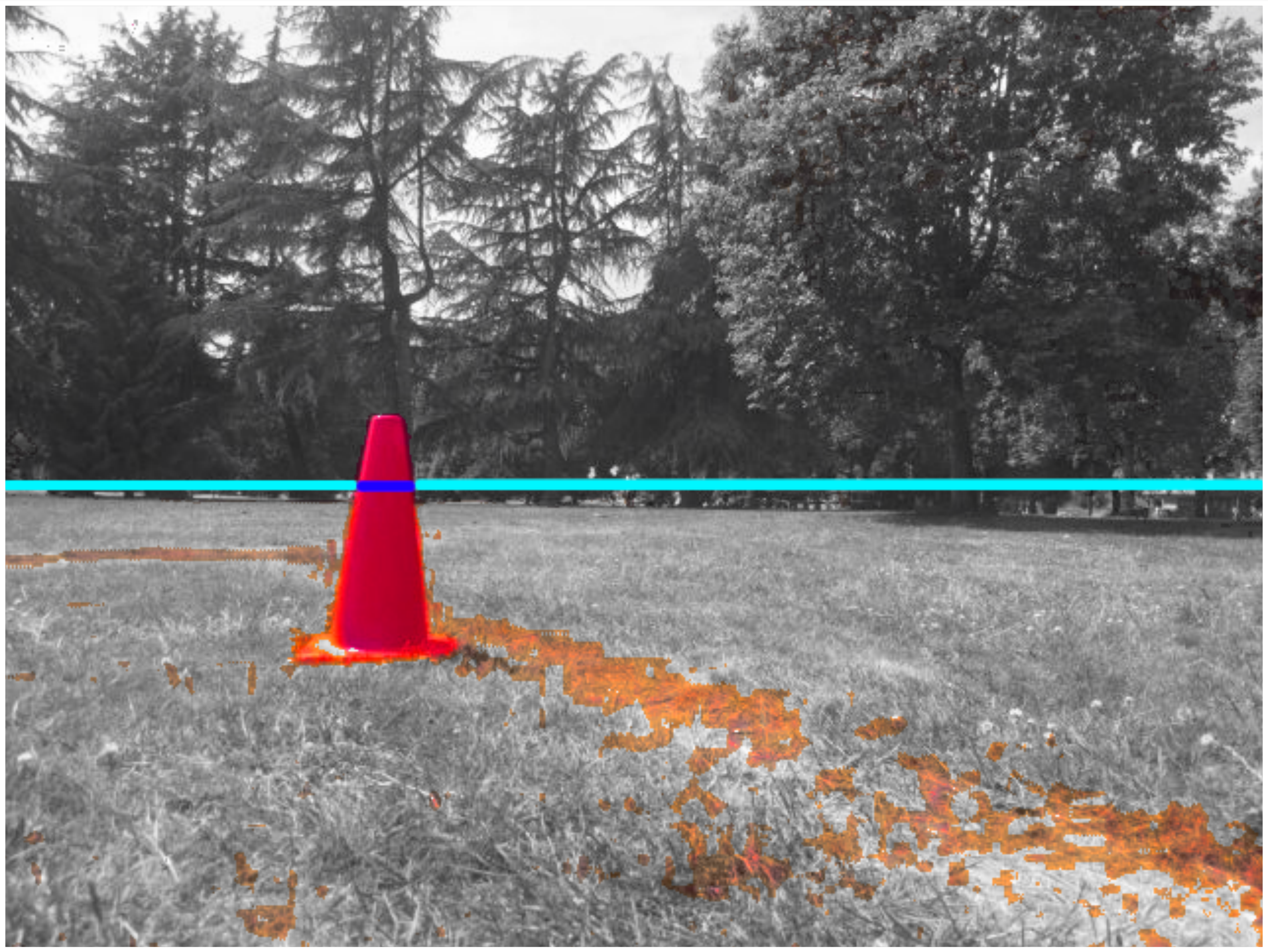
CLASSIC DETECTION TECHNIQUE #2

Advanced Image Filtering

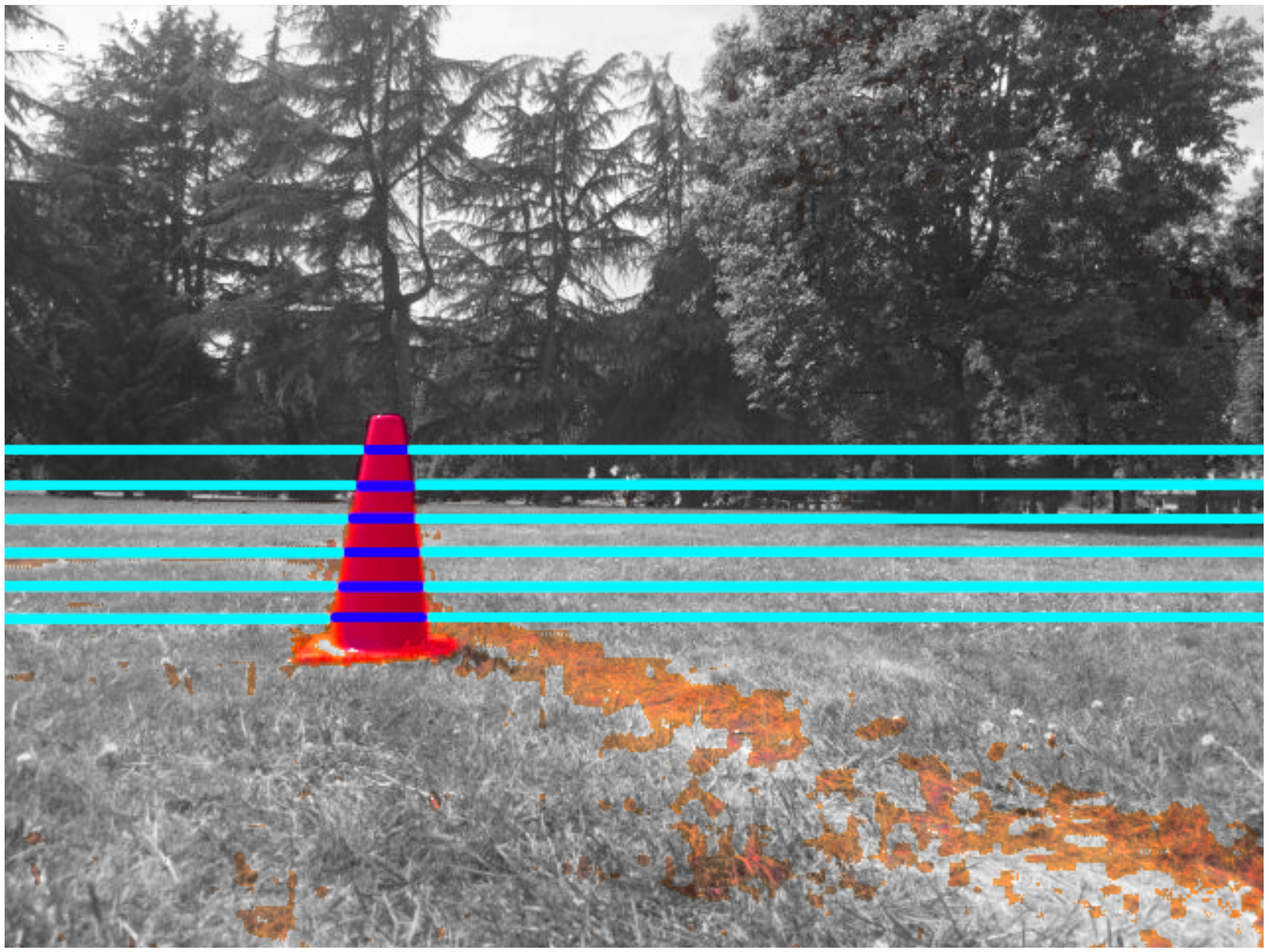
- Manipulation of the image using specific transforms allows better analysis
- Each step requires filters and image processing algorithms
- One approach:
  1. Colorspace transformation to reduce illumination variability
  2. Edge detection to find vertical regions of high contrast
  3. Heuristics to decide whether edges define a "cone-like object"
- OpenCV makes this somewhat approachable

Advanced
Filtering

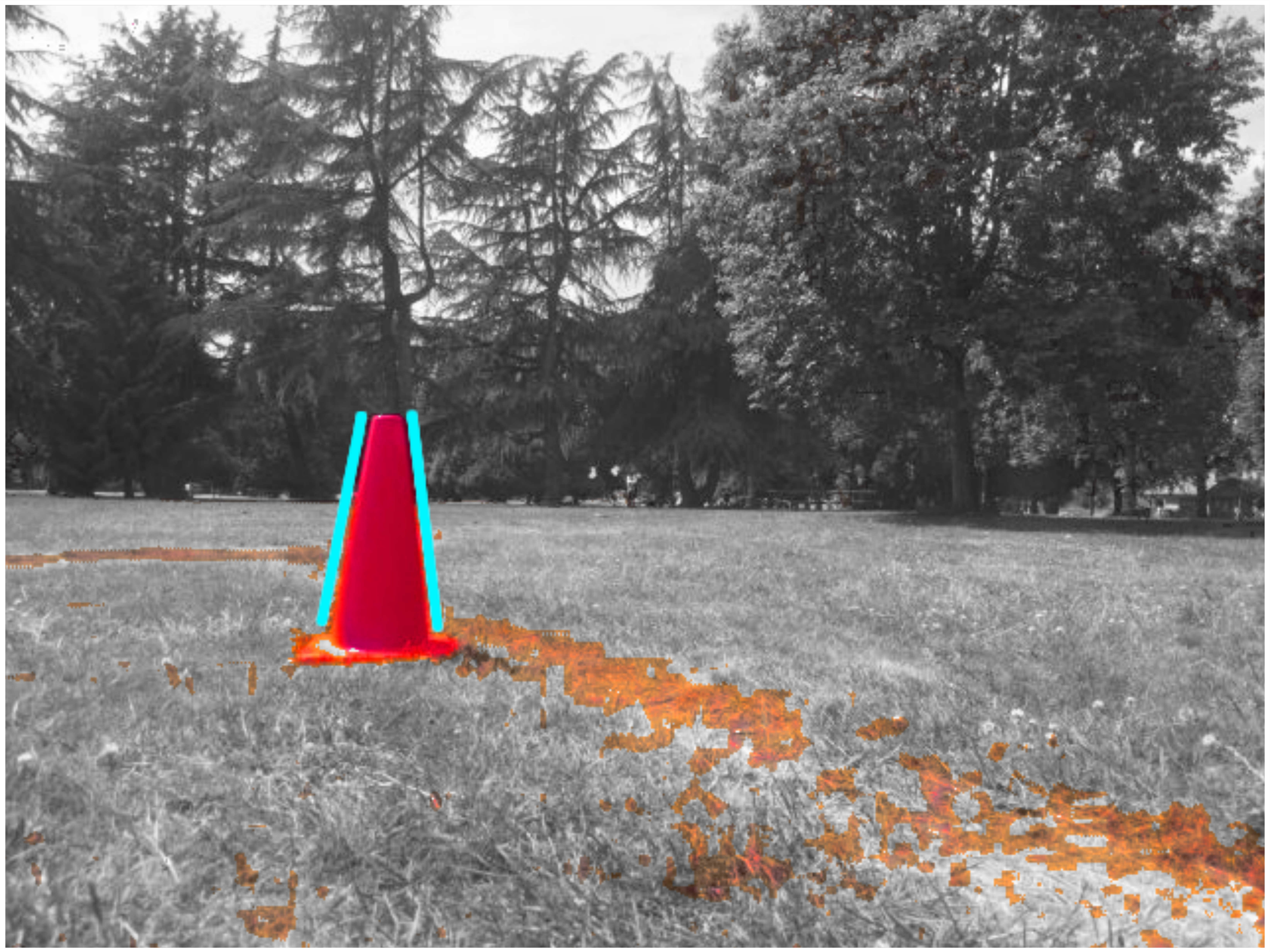
Advanced Filtering



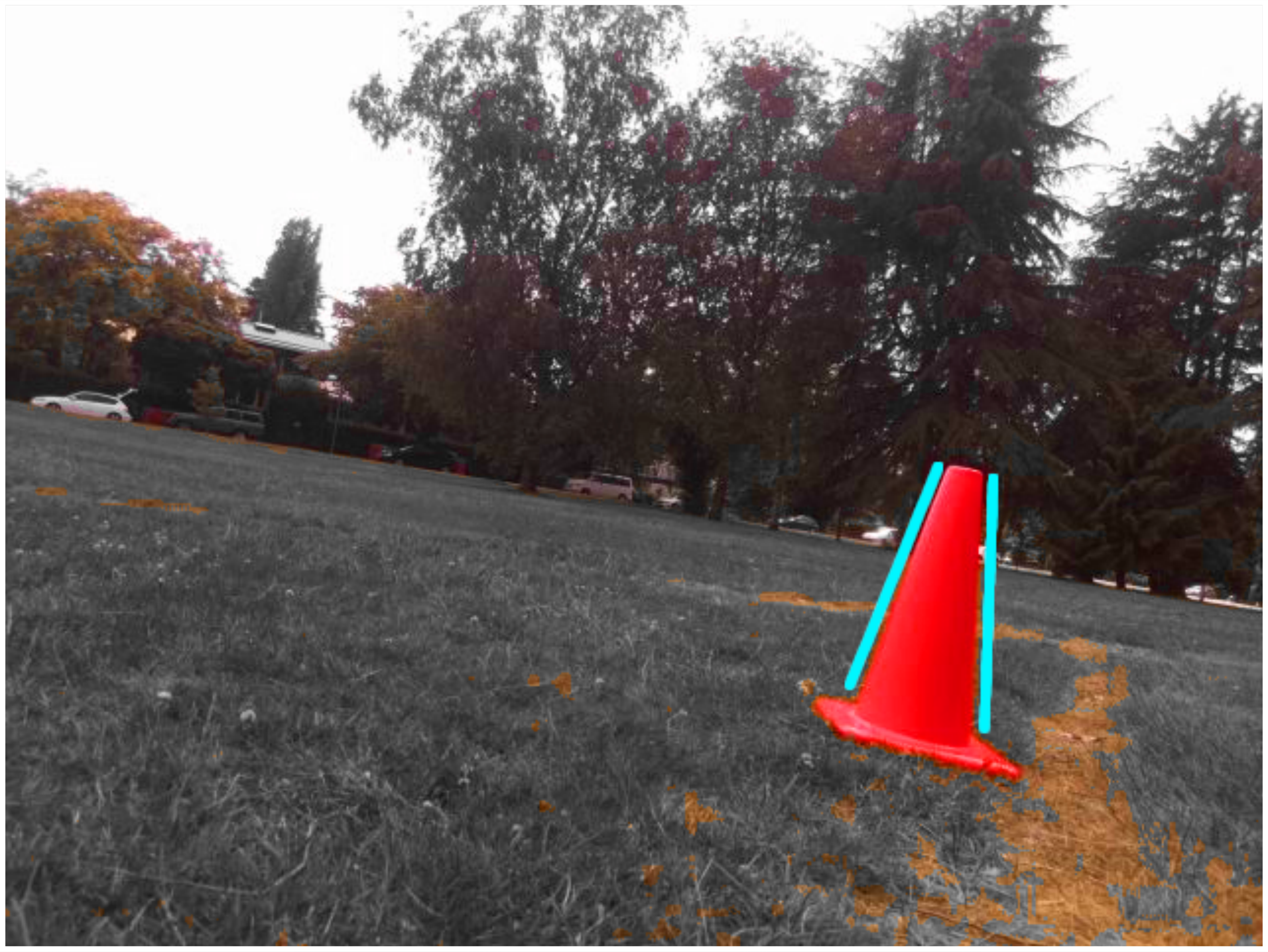
Advanced Filtering



Advanced Filtering



Advanced Filtering



Advanced Filtering

Advanced Filtering

Advanced
Filtering

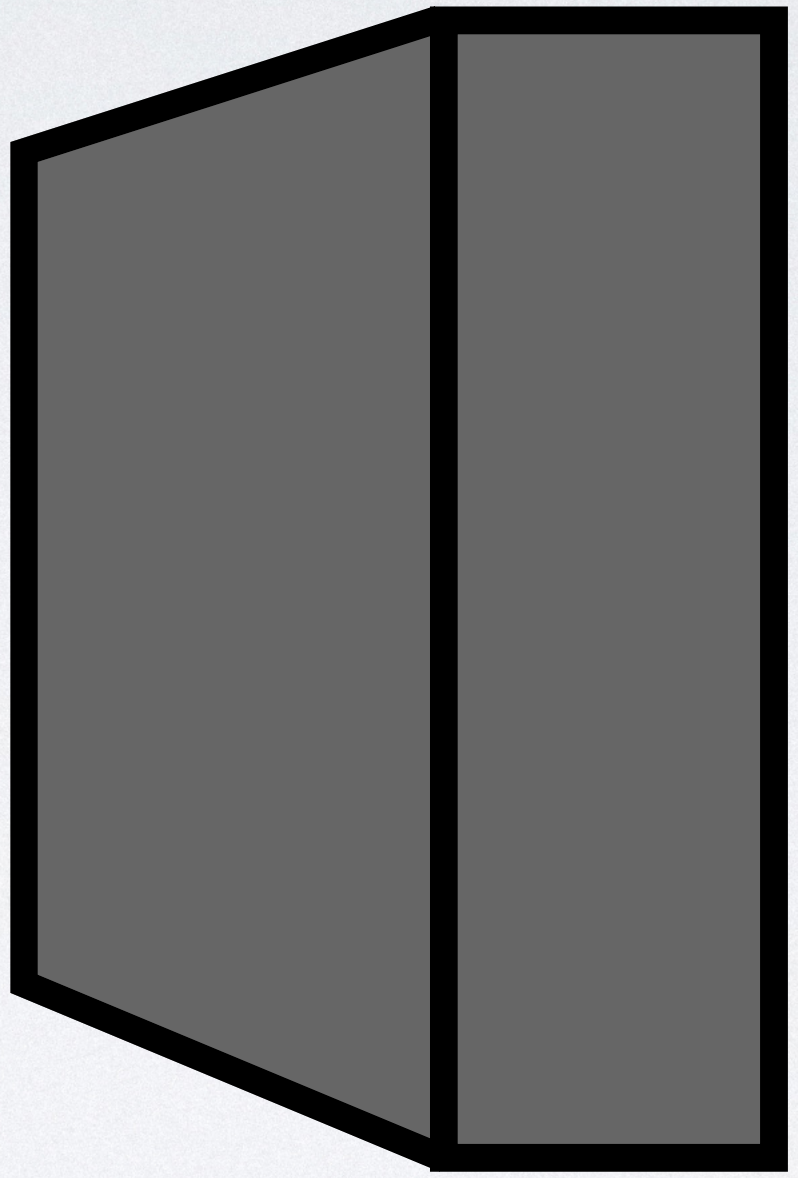
MACHINE LEARNING

Black Boxes and Statistical Outcomes

Machine learning (ML) is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can effectively generalize and thus perform tasks without explicit instructions. – Wikipedia

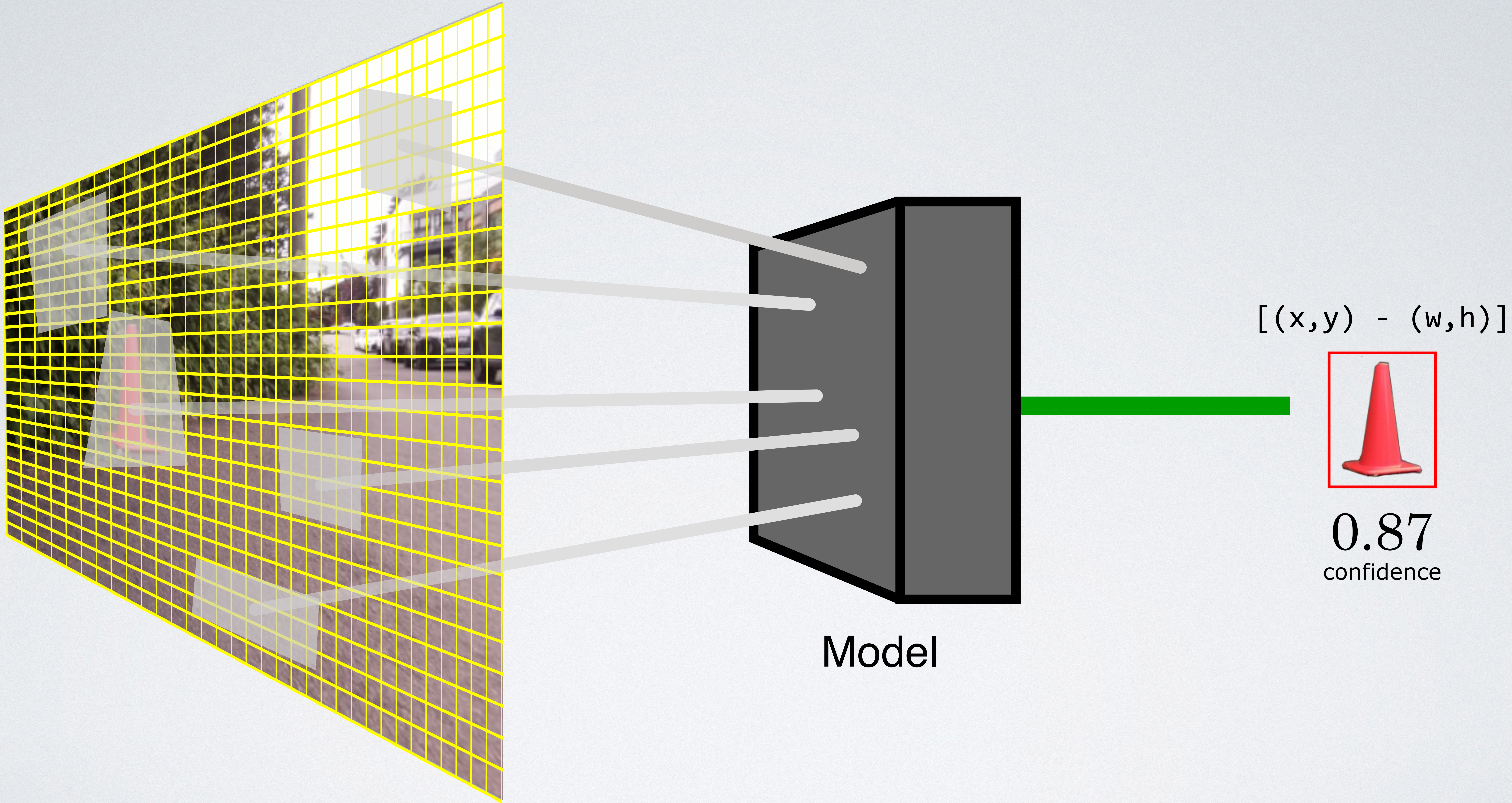
"Statistical algorithms" = Magic Black Box!

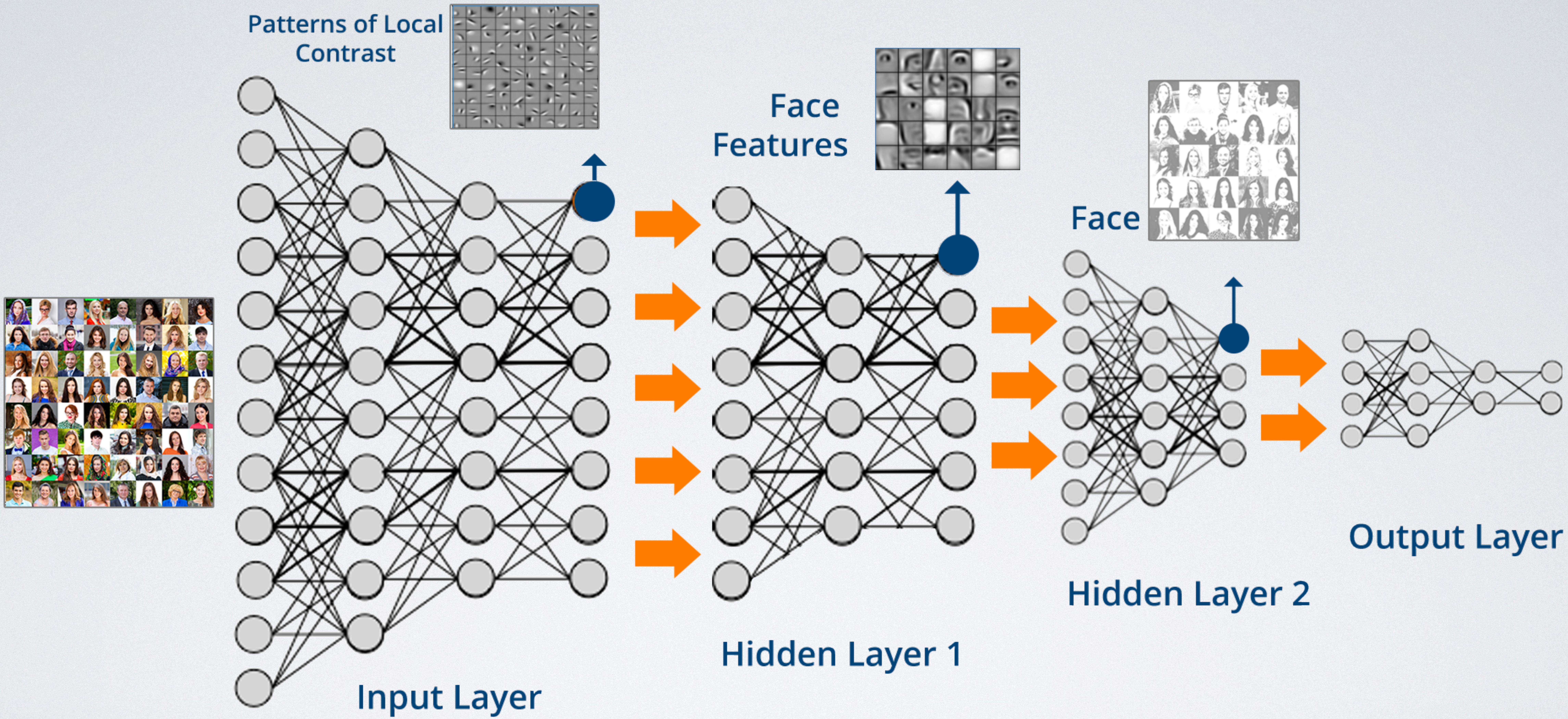
Machine
Learning



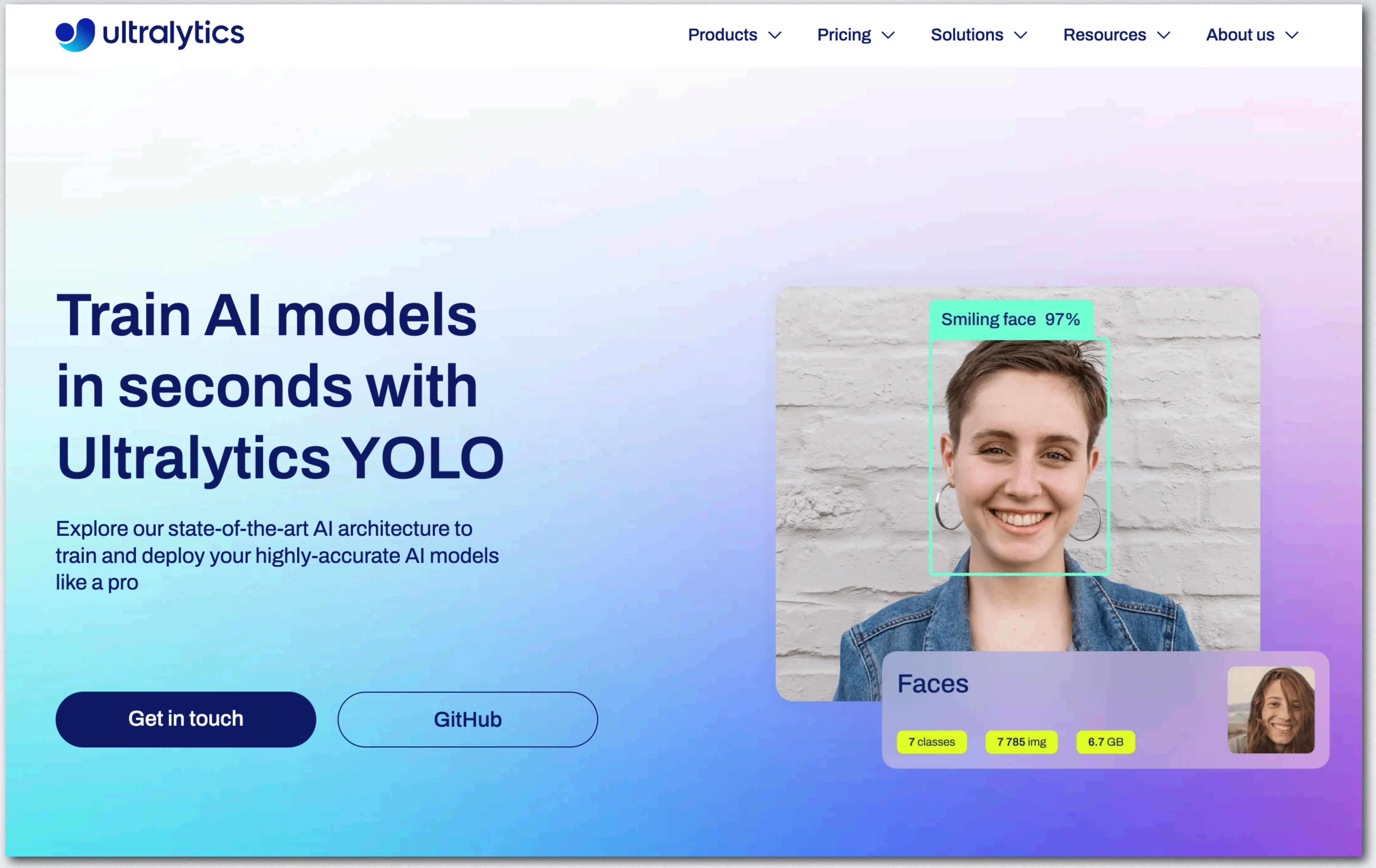
Model

Machine
Learning


[(x,y) - (w,h)]
0.87
confidence
Model



Patterns of Local
Contrast
Face
Features
Face
Input Layer
Hidden Layer 1
Hidden Layer 2
Output Layer



ultralytics.com

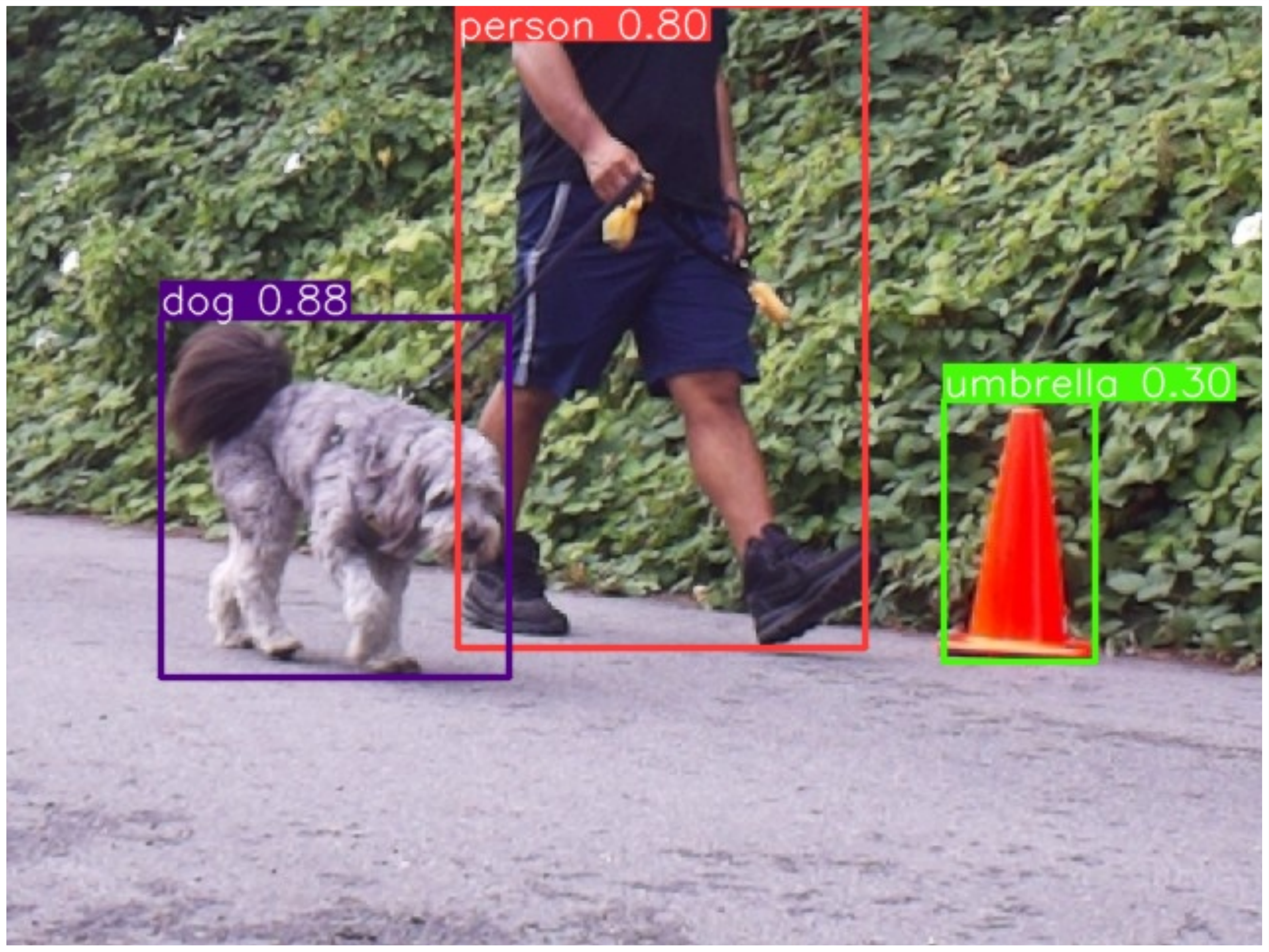
python prediction code

```
from ultralytics import YOLO

# Load a model
model = YOLO('yolov8n.pt')  # pretrained YOLOv8n model

# Run batched inference on a list of images
results = model(['im1.jpg', 'im2.jpg'])  # return a list of Results objects

# Process results list
for result in results:
    boxes = result.boxes  # Boxes object for bbox outputs
    masks = result.masks  # Masks object for segmentation masks outputs
    keypoints = result.keypoints  # Keypoints object for pose outputs
    probs = result.probs  # Probs object for classification outputs
```



Using the original **yolov8n.pt** model without my training

command line to train a new model

```
# Start training from a pretrained *.pt model
yolo detect train data=dataset.yml model=yolov8n.pt epochs=100 imgsz=640
```

command line to train a new model

```
# Start training from a pretrained *.pt model
yolo detect train data=dataset.yml model=yolov8n.pt epochs=100 imgsz=640
```

dataset.yml

```
# ferdy-cone-data dataset description
path: "C:\\Users\\bobcook\\gits\\bob.cook\\ferdy-cone-data"
train: images
val: images
test:  # test images (optional)
names:
  0: cone
```

command line to train a new model

```
# Start training from a pretrained *.pt model
yolo detect train data=dataset.yml model=yolov8n.pt epochs=100 imgsz=640
```

dataset.yml

```
# ferdy-cone-data dataset description
path: "C:\\Users\\bobcook\\gits\\bob.cook\\ferdy-cone-data"
train: images
val: images
test:  # test images (optional)
names:
  0: cone
```

Pre-trained model

(COCO dataset, 80 objects, 150k+ images)

ULTRALYTICS YOLO LICENSING

Know Before You Code

- YOLO is available to enthusiasts and students under the AGPL-3.0

- All models created with and software using YOLO must follow the same license

- https://github.com/ultralytics/ultralytics

MACHINE LEARNING

Training Your Own Model

1. Collect images
2. Label images
3. Train new model
4. Test new model
5. Deploy that model on the robot

MACHINE LEARNING

Training Your Own Model

1. Collect images
2. Label images

Manual steps

3. Train new model
4. Test new model

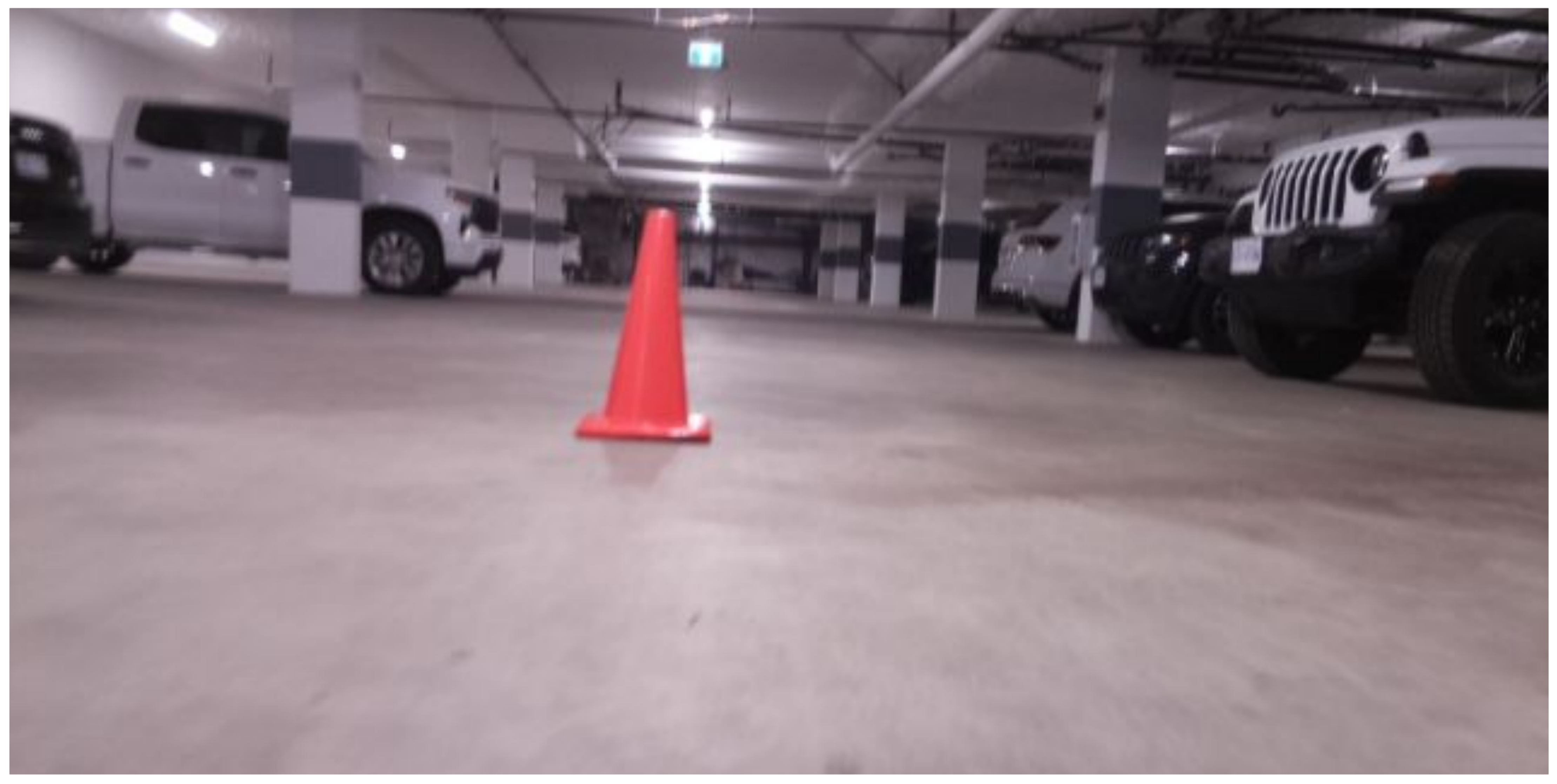
Automated steps

5. Deploy that model on the robot

STEP 1. COLLECT IMAGES

Make Ferdy Do Some Work

video




cones



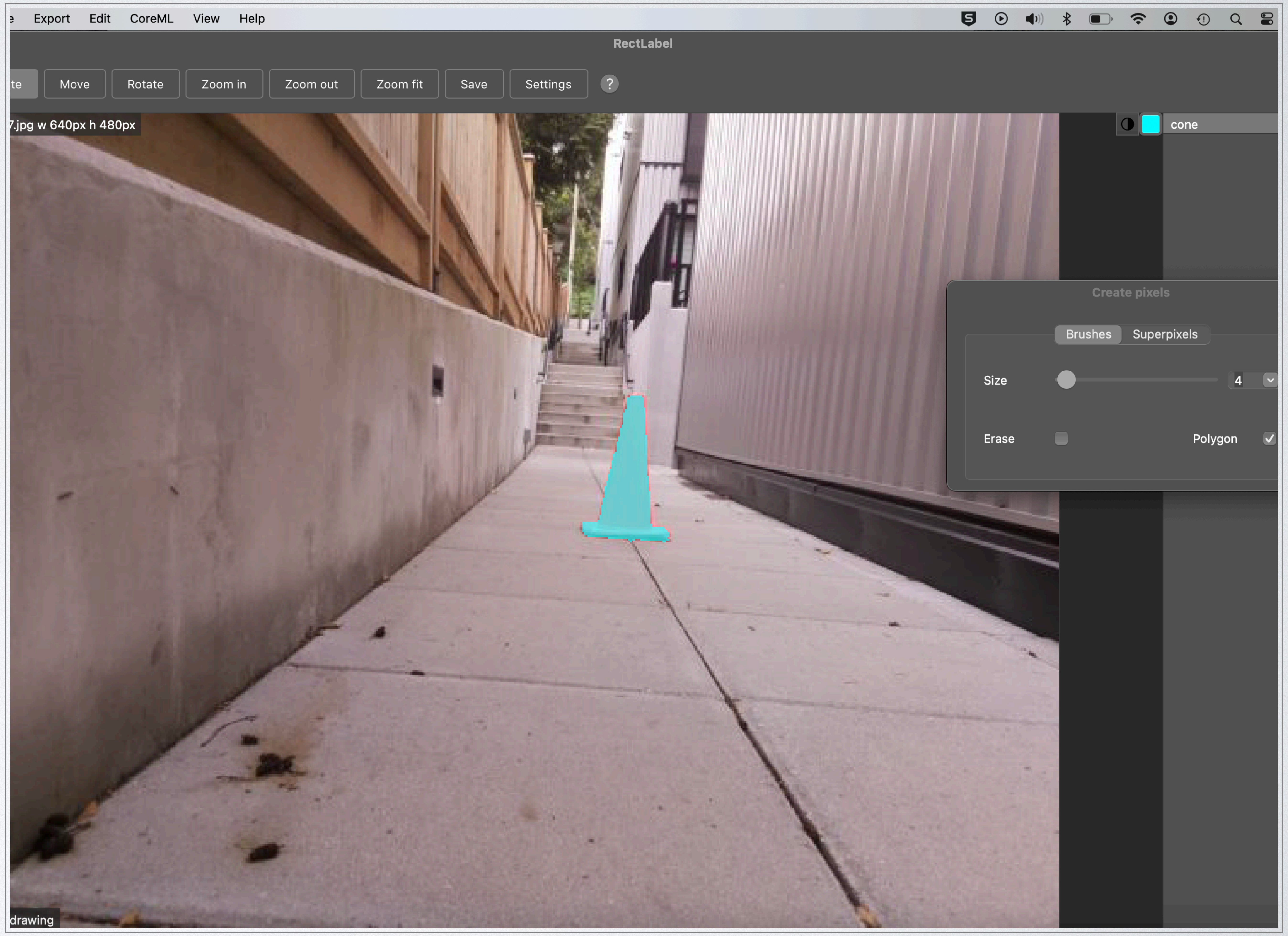

more cones

negative cases

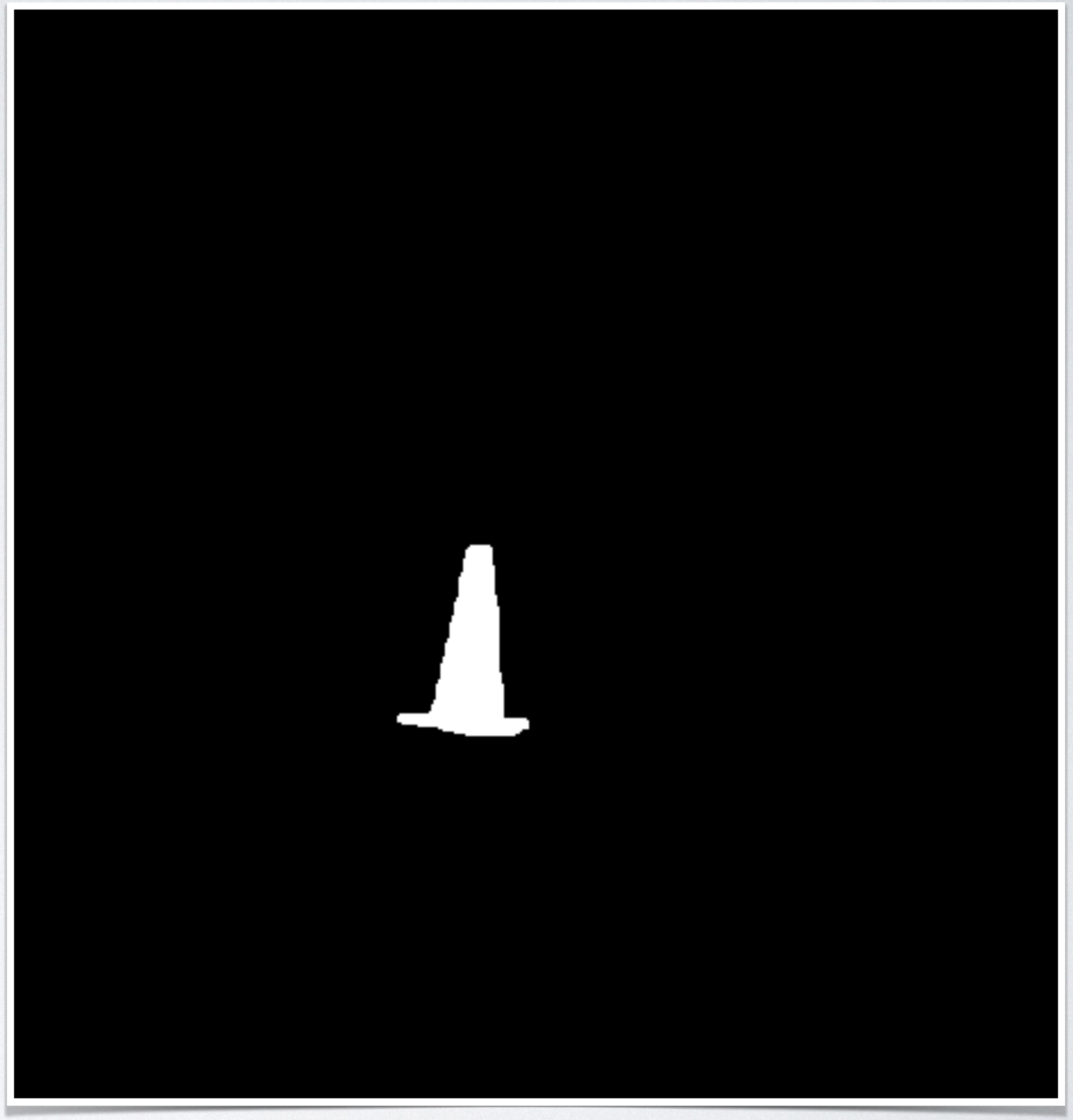
STEP 2. LABEL IMAGES

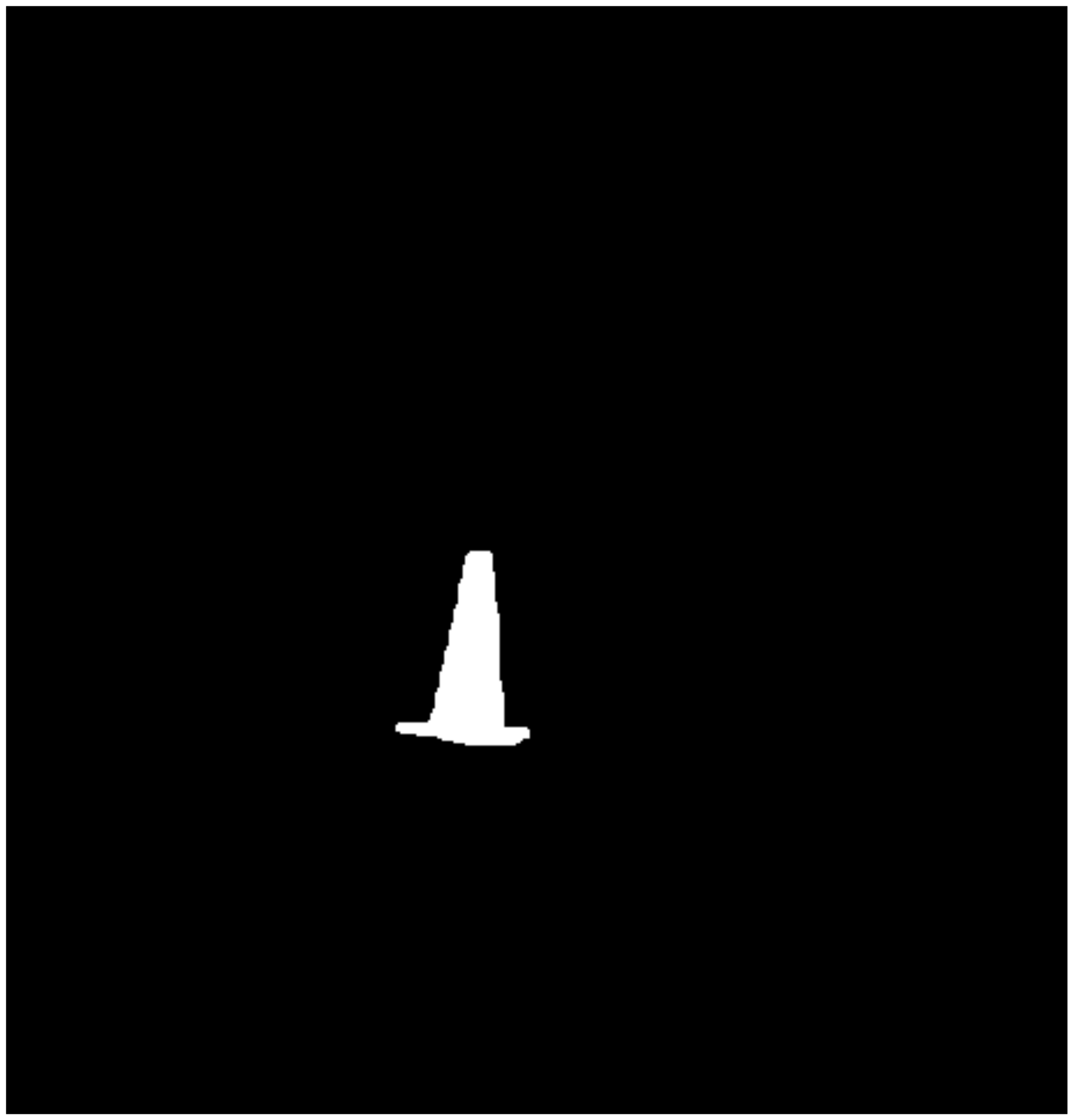
Create Training Dataset

- **RectLabel Pro** software
- Manually annotate images
- Highlight "feature of interest"
- Exports a standard file format used with training program




RectLabel
Open folder
Create
Move
Rotate
Zoom in
Zoom out
Zoom fit
Save
Settings
?
88/356 20230617160435.jpg w 640px h 480px
cone
Create pixels
Brushes
Superpixels
Size
4
Erase
Polygon






```xml
<annotation>
    <folder>images</folder>
    <filename>20230618150315.jpg</filename>
    <size>
        <width>640</width>
        <height>480</height>
        <depth>3</depth>
    </size>
    <object>
        <name>cone</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <pixels>
            <id>0</id>
        </pixels>
        <bndbox>
            <xmin>259.595306</xmin>
            <ymin>236.508331</ymin>
            <xmax>316.506256</xmax>
            <ymax>319.335419</ymax>
        </bndbox>
    </object>
</annotation>
```

Final data saved as a TXT record in YOLO format:

```
0 0.454688 0.491667 0.451562 0.495833 0.450000 0.516667 0.446875 0.522917 0.446875
0.539583 0.443750 0.545833 0.443750 0.556250 0.440625 0.564583 0.440625 0.577083
0.437500 0.583333 0.437500 0.593750 0.434375 0.602083 0.434375 0.614583 0.431250
0.620833 0.431250 0.633333 0.428125 0.643750 0.425000 0.647917 0.404687 0.650000
0.404687 0.654167 0.418750 0.658333 0.431250 0.658333 0.435937 0.662500 0.451562
0.666667 0.484375 0.666667 0.490625 0.660417 0.493750 0.660417 0.492188 0.652083
0.476562 0.650000 0.476562 0.620833 0.473437 0.608333 0.473437 0.550000 0.470313
0.537500 0.470313 0.512500 0.467187 0.491667
```

STEP 3. TRAIN NEW MODEL

Iterate To Improve Detection

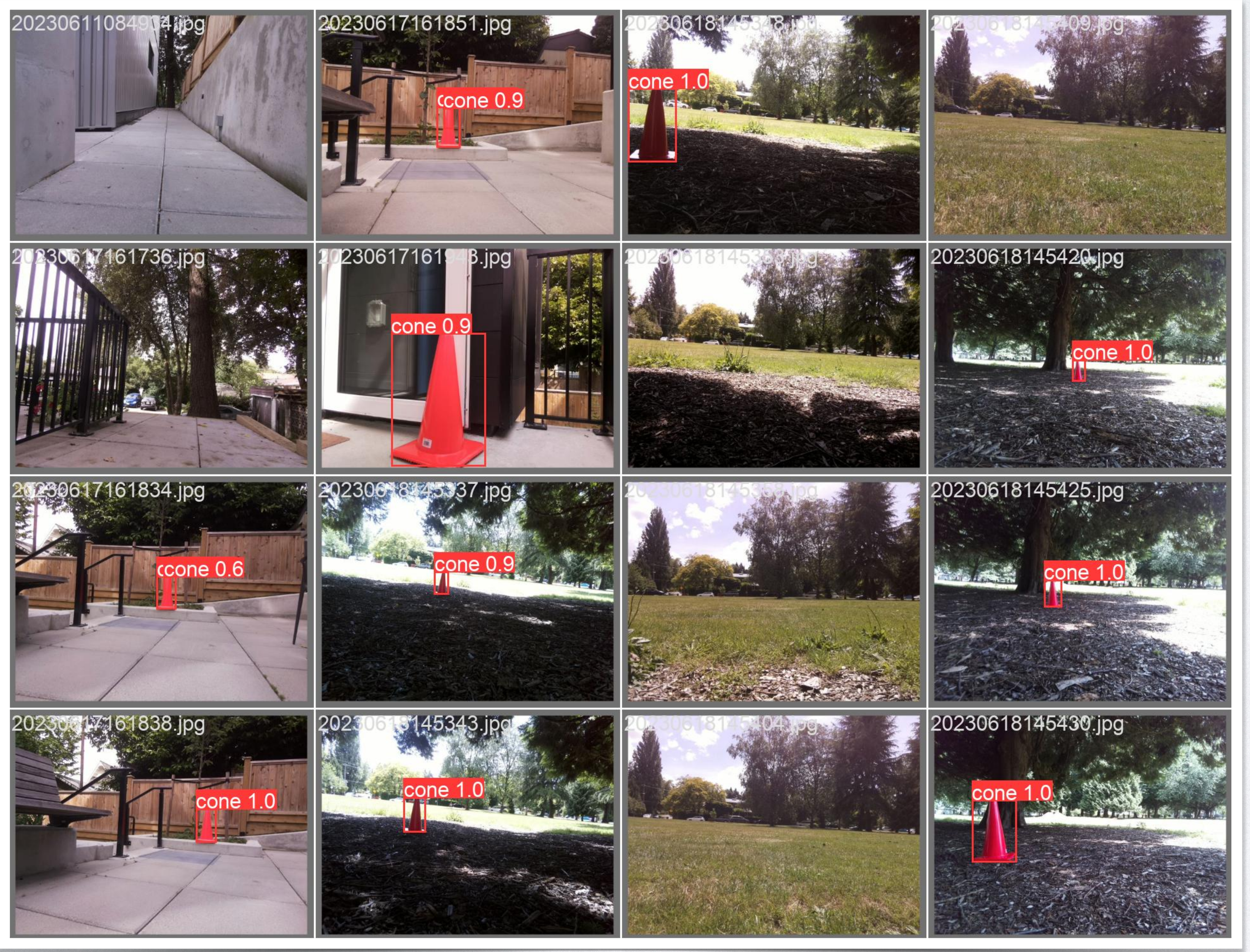
```
# training is simple once you have the data
yolo detect train data=dataset.yml model=yolov8n.pt epochs=3000 imgsz=640
```

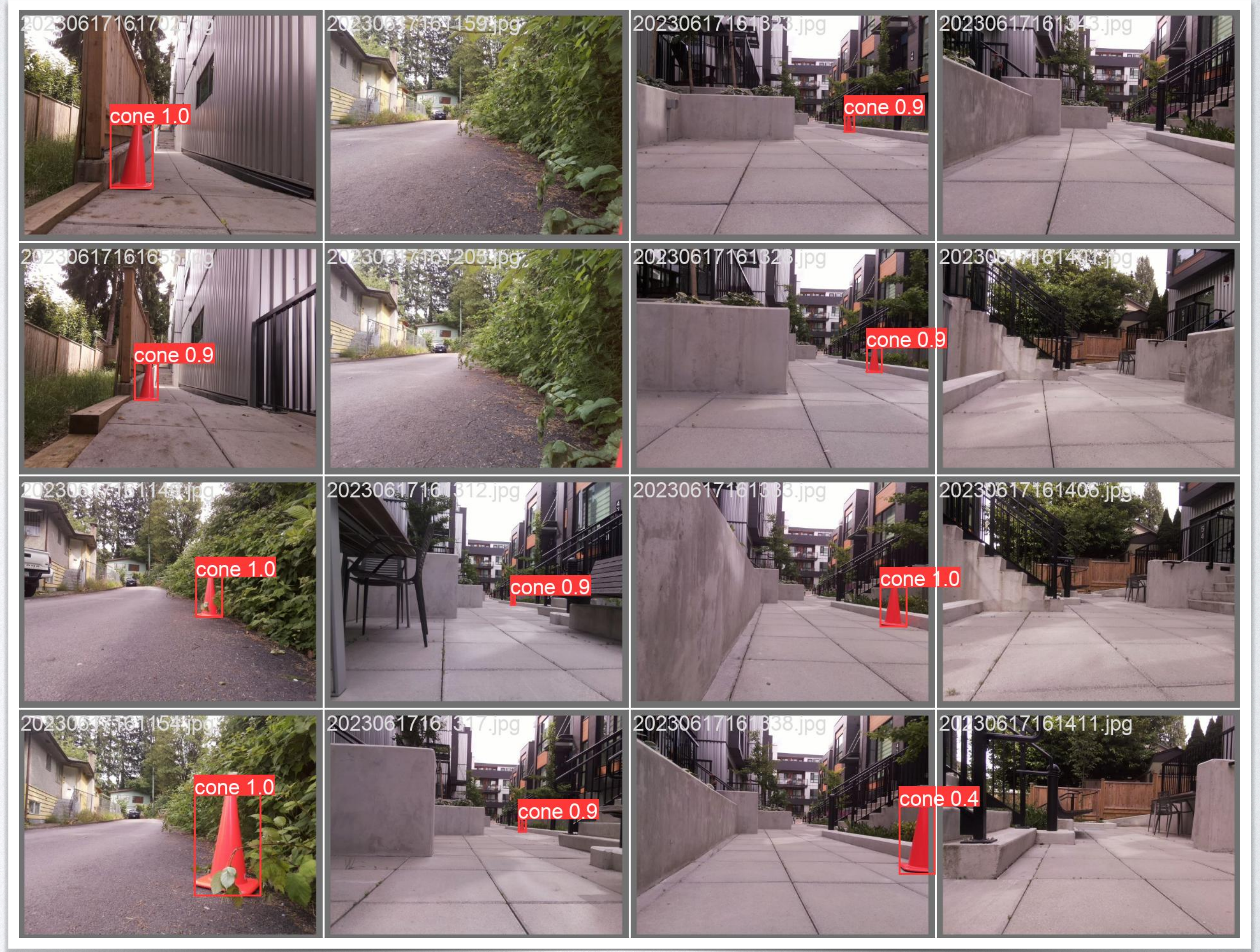
Using a Windows gaming laptop:

- 4 core 3.3 GHz i7, 16 GB RAM, NVIDIA RTX 3070 GPU
- Less than 4 hours for a successful training run

STEP 4. TEST NEW MODEL

But Does It Work?





After my training


cone 0.96

cone 0.97

cone 0.96

Ultralytics recommends:

- 1,500+ images per feature to be identified
- Images from different times of day, different seasons, different weather, different lighting, different angles, different sources
- Separate validation set of labeled images
- +10% additional images without desired feature

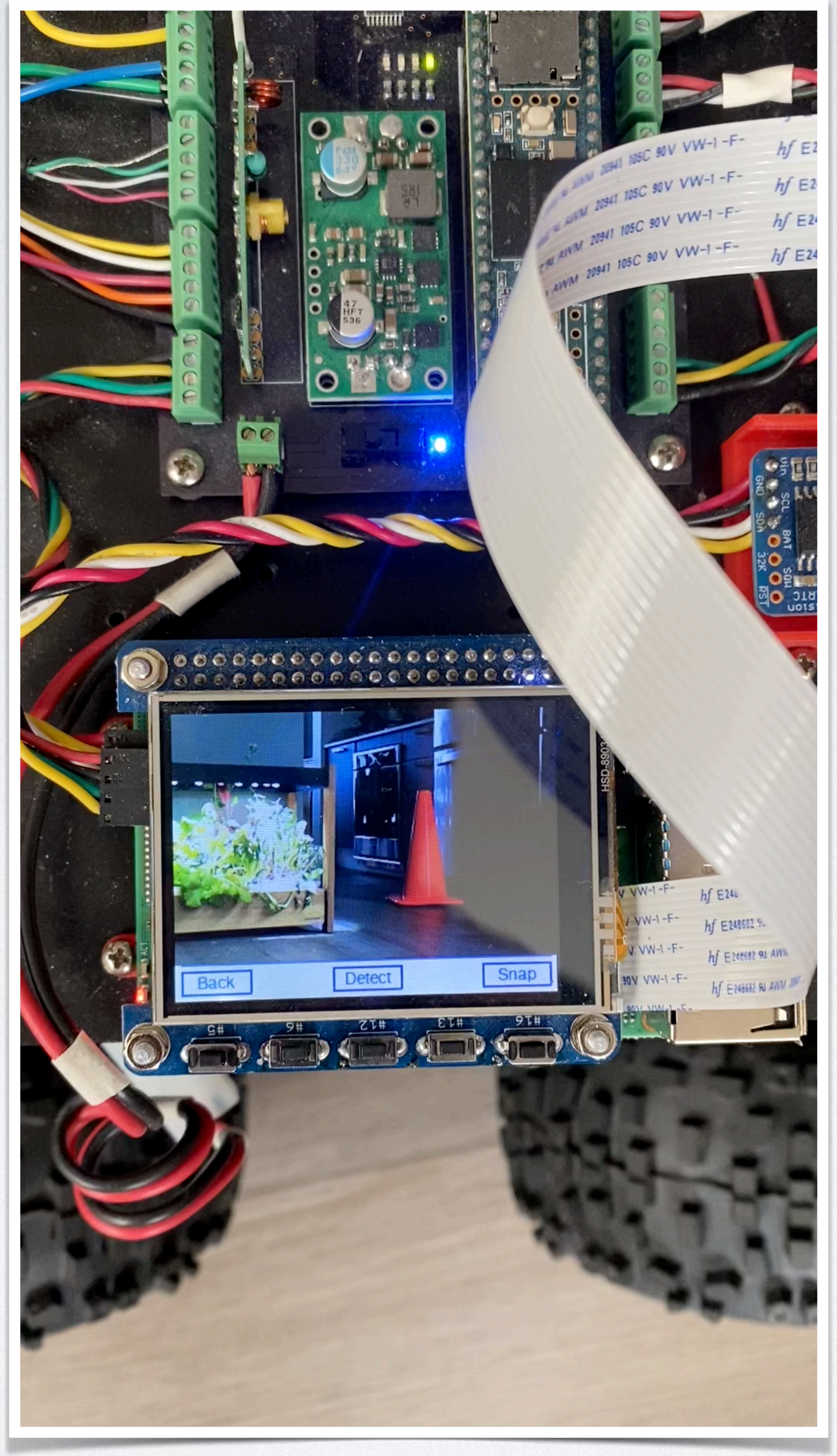
My experience, with pretty good results:

- 357 total images
- 227 positive images (each contains a single labeled cone)
- Training dataset = validation dataset
- Started from pre-trained weights from the COCO dataset

5. DEPLOY ON THE ROBOT

Let's Get Real

- Model = 6.2 MB binary file
- My python code is pretty much like the Ultralytics example
- Every detection is saved (image + metadata)
- PIL package allows image annotation in python
- Test mode on the robot allows easy experimentation



video

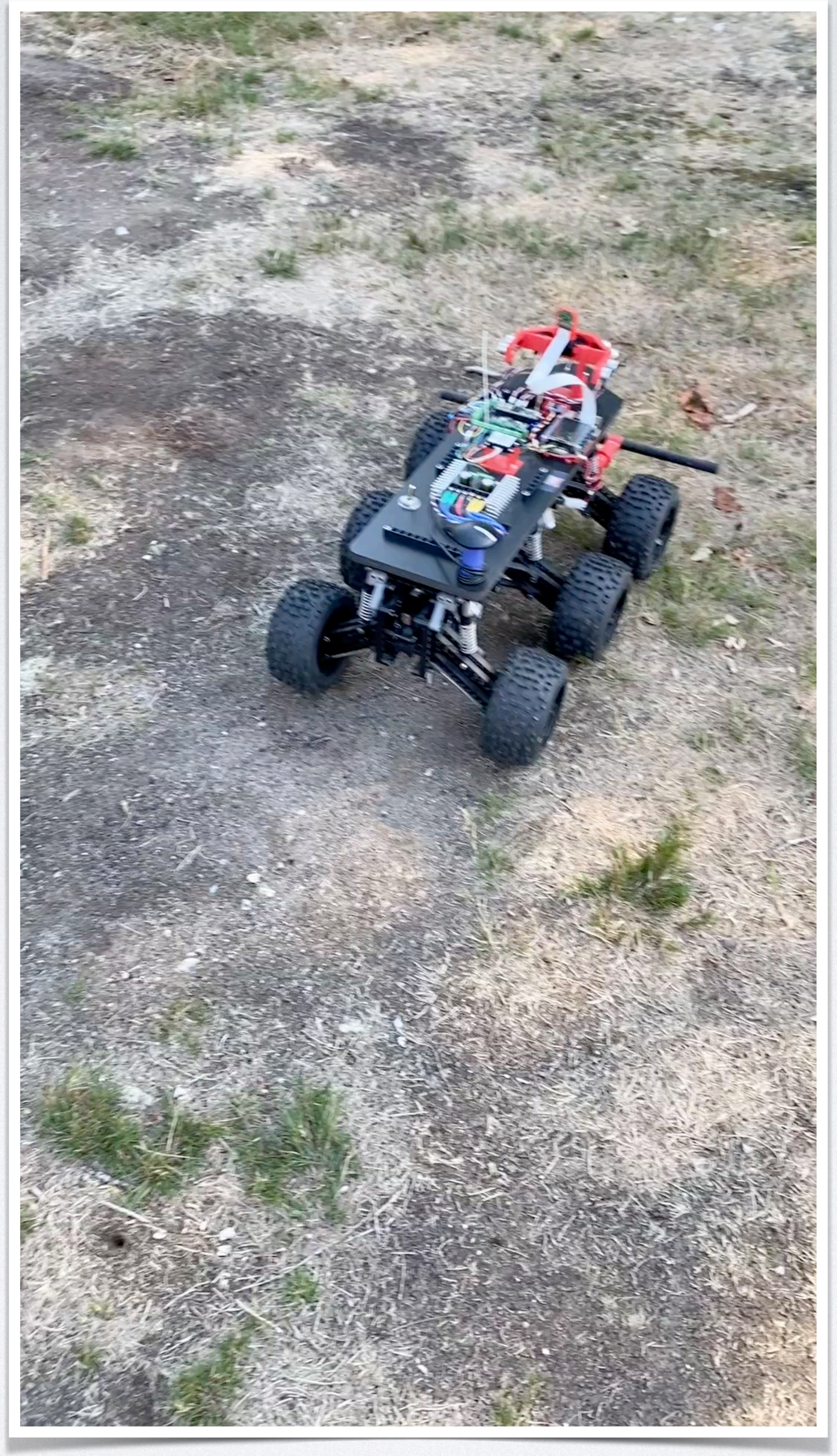
```
confidence: 0.9168590307235718
time:       1648.0088233947754
center:     459, 244
size:       60, 124
zone:       1
```

example saved image + metadata

- Results were excellent
  - Zero false positives in my testing during development
  - Performed well enough on the RPi 4 to be *somewhat practical*

- Prediction on the RPi 4 requires ~1.6 sec

- Not real-time prediction means different strategy required
  - Detect – move briefly – detect again – move briefly again – repeat



video

ACTUAL CONTEST RESULTS

```
confidence: 0.848670244216919

time:       1683.2005977630615

center:     564, 183

size:       37, 89

zone:       2
```

ACTUAL CONTEST RESULTS

```
confidence: 0.8739311099052429

time:       1692.2695636749268

center:     53, 306

size:       44, 101

zone:       -2
```

ACTUAL CONTEST RESULTS

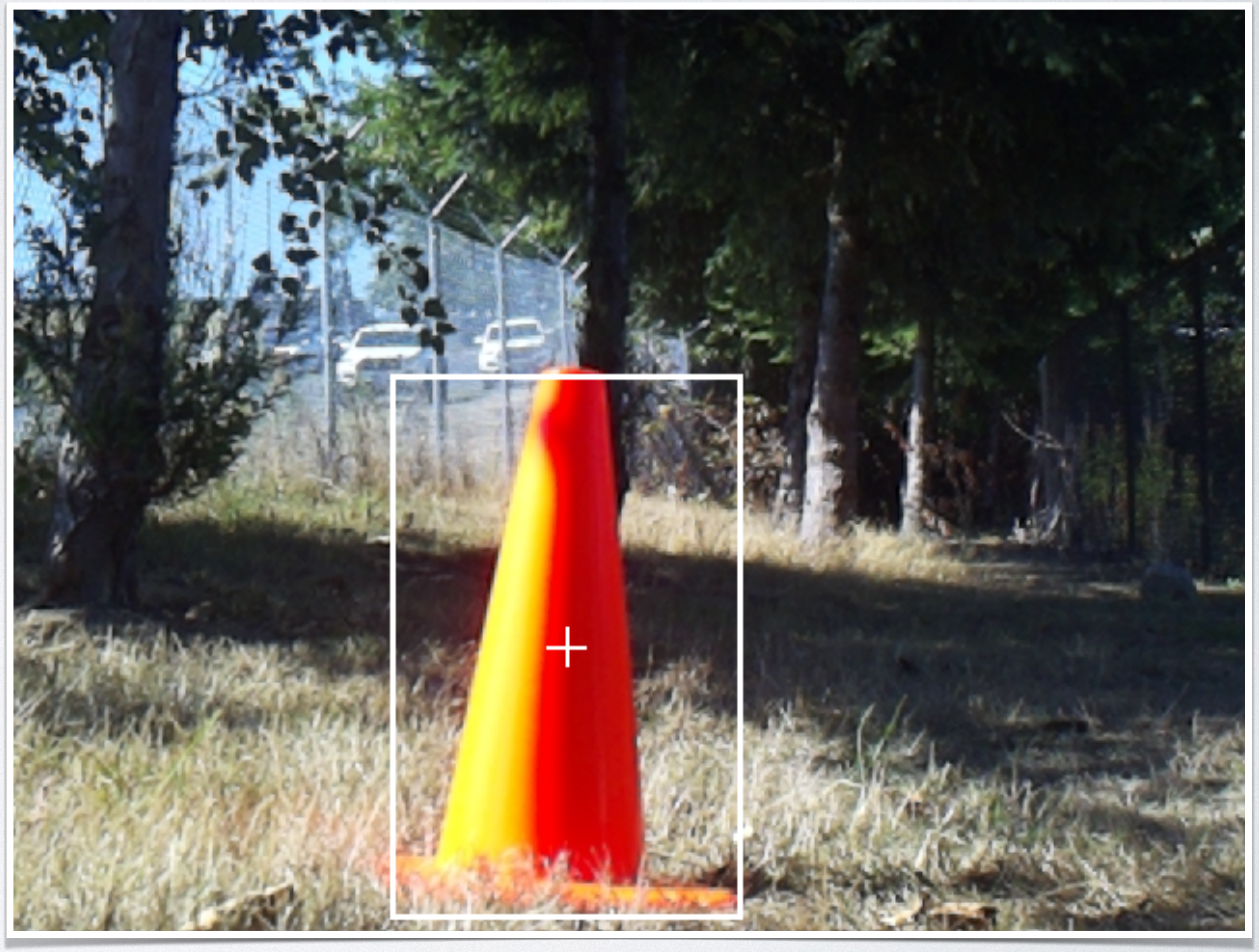
```
confidence: 0.9472693204879761

time:       1559.1990947723389

center:     287, 331

size:       182, 282

zone:       0
```



ACTUAL CONTEST RESULTS

```
confidence: 0.8713879585266113

time:       1704.5116424560547

center:     278, 305

size:       175, 346

zone:       0
```

video

ACTUAL CONTEST RESULTS

```
confidence: 0.9209558963775635
time:       1727.0872592926025
center:     101, 417
size:       59, 124
zone:       -2
```

ACTUAL CONTEST RESULTS

```
confidence: 0.9539856910705566

time:       1634.819746017456

center:     244, 272

size:       85, 136

zone:       -1
```

ACTUAL CONTEST RESULTS

```
confidence: 0.958247184753418

time:       1652.2455215454102

center:     394, 354

size:       139, 199

zone:       1
```

And some unexpected results…

ACTUAL CONTEST RESULTS

```
confidence: 0.2954730689525604
time:       1627.6228427886963
center:     114, 5
size:       93, 10
zone:       -2
```

ACTUAL CONTEST RESULTS

```
confidence: 0.6940656304359436

time:       1663.5806560516357

center:     306, 15

size:       26, 31

zone:       0
```

ACTUAL CONTEST RESULTS

```
confidence: 0.5635690093040466

time:       1737.0731830596924

center:     395, 196

size:       10, 14

zone:       1
```

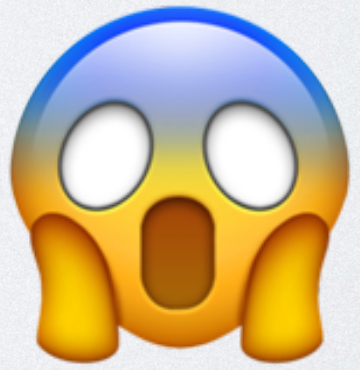
OTHER CONE IMAGES

images courtesy of Doug Paradis

NO CONES DETECTED

???

images courtesy of Doug Paradis



images courtesy of Doug Paradis

Ummm…

https://designer.microsoft.com/image-creator

Ummm…

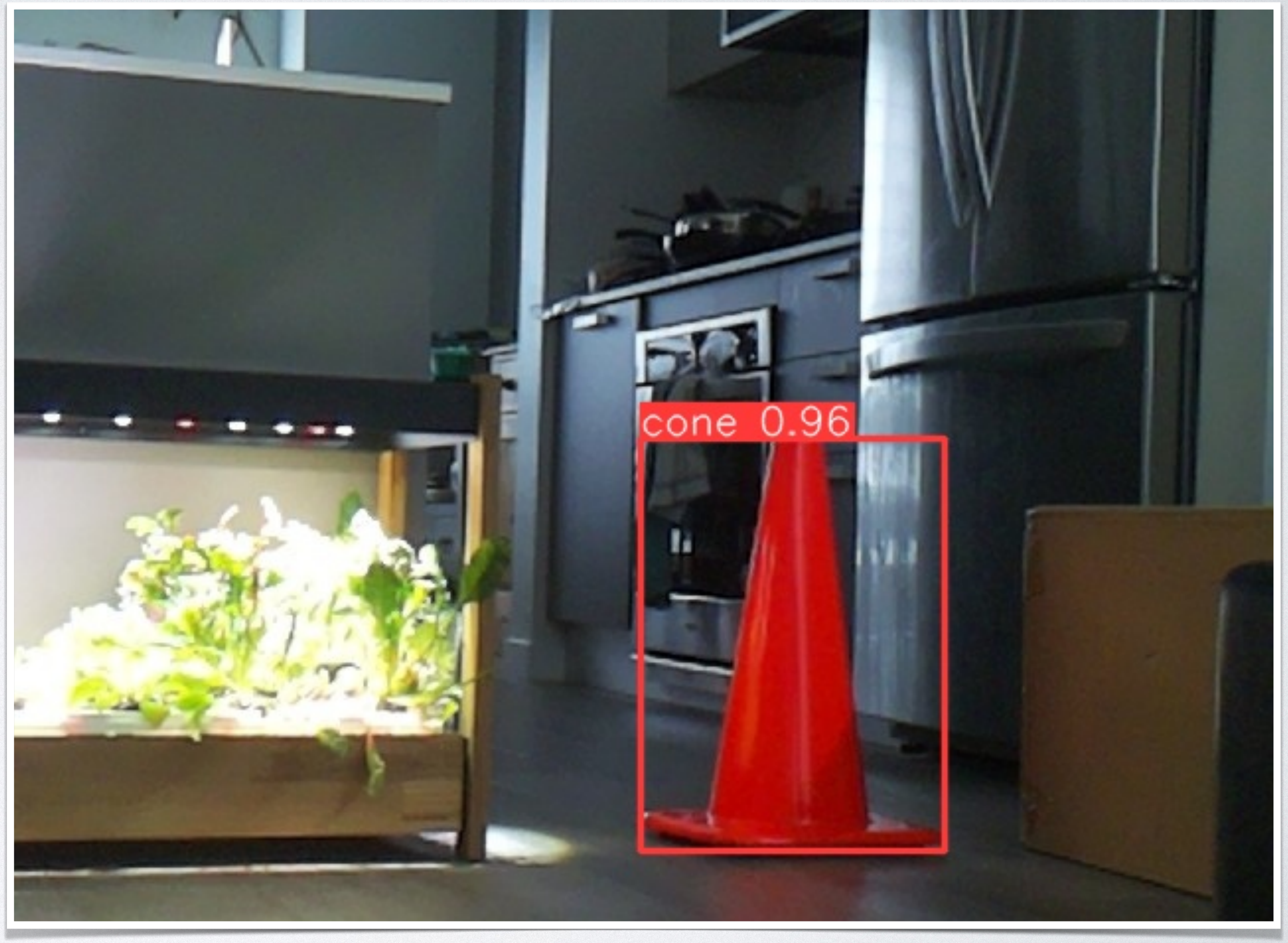
Yep, it's a cone…

https://designer.microsoft.com/image-creator

OUTCOMES

My Theory Why This (Mostly) Worked

- Same camera to capture training dataset as well as predictions
- Highly constrained application
- Similar conditions:
  - Weather conditions
  - Backgrounds
  - Specific target feature



SUMMARY

- Training an ML model for object detection is practical for mere mortals like you and me
- Training data selection is the most important factor
  - Camera quality and image resolution for training data are much less important than you might initially believe
  - Very high resolution images are usually not a benefit
- Your new models can run on a variety of hardware, and you can convert models between different formats e.g. ONNX, TinyML
- Cone detection is only the start… <u>everything</u> is a vision problem to some degree or another

QUESTIONS?

APPENDIX

VARIOUS LINKS

Maybe Inspirational / Useful to You Too?

Ultralytics tutorial for object detection

https://docs.ultralytics.com/tasks/detect/

COCO image dataset

https://cocodataset.org/

Ferdy Cone Training Data & Model

https://gitlab.bronzestarfish.com/bob.cook/ferdy-cone-data

My videos from SRS Robo-Magellan event September 9:

https://www.youtube.com/playlist?list=PLiogHjvupE6BqIUVyylJFg6cnRw15vqYF

DPRG RoboColumbus event November 18:

https://www.youtube.com/playlist?list=PLXixJXO-dNbr2sSJeTX3-IhGZuqffVnHe

END