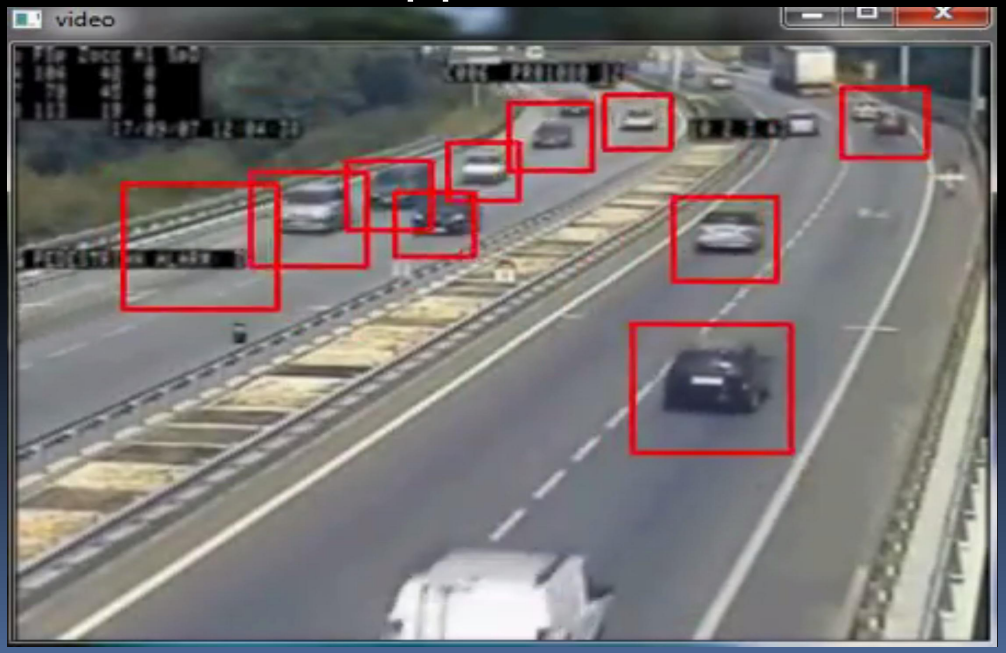
Optimizing Neural Networks using Deformable Part Model

CSS 487 - Computer Vision
Robert Griswold & Ryu Muthui
12/07/2016

Project Goals:

- Inspiration for the project was how objects could be identified within video feeds and image sets with high accuracy and speed.

- For example, a real-world application for this could be a video feed that monitors cars on a stretch of road that can identify each make and model of vehicle.



Project Goals: Cont.

- Focused on researching ways to remove confounding information in a search image for a neural network.
- Our next focus was an optimization of the neural network so that it could be run in real-time.
- Ultimately, our goal evolved into making a neural network robust enough to be able to be used in real-time computer vision applications.

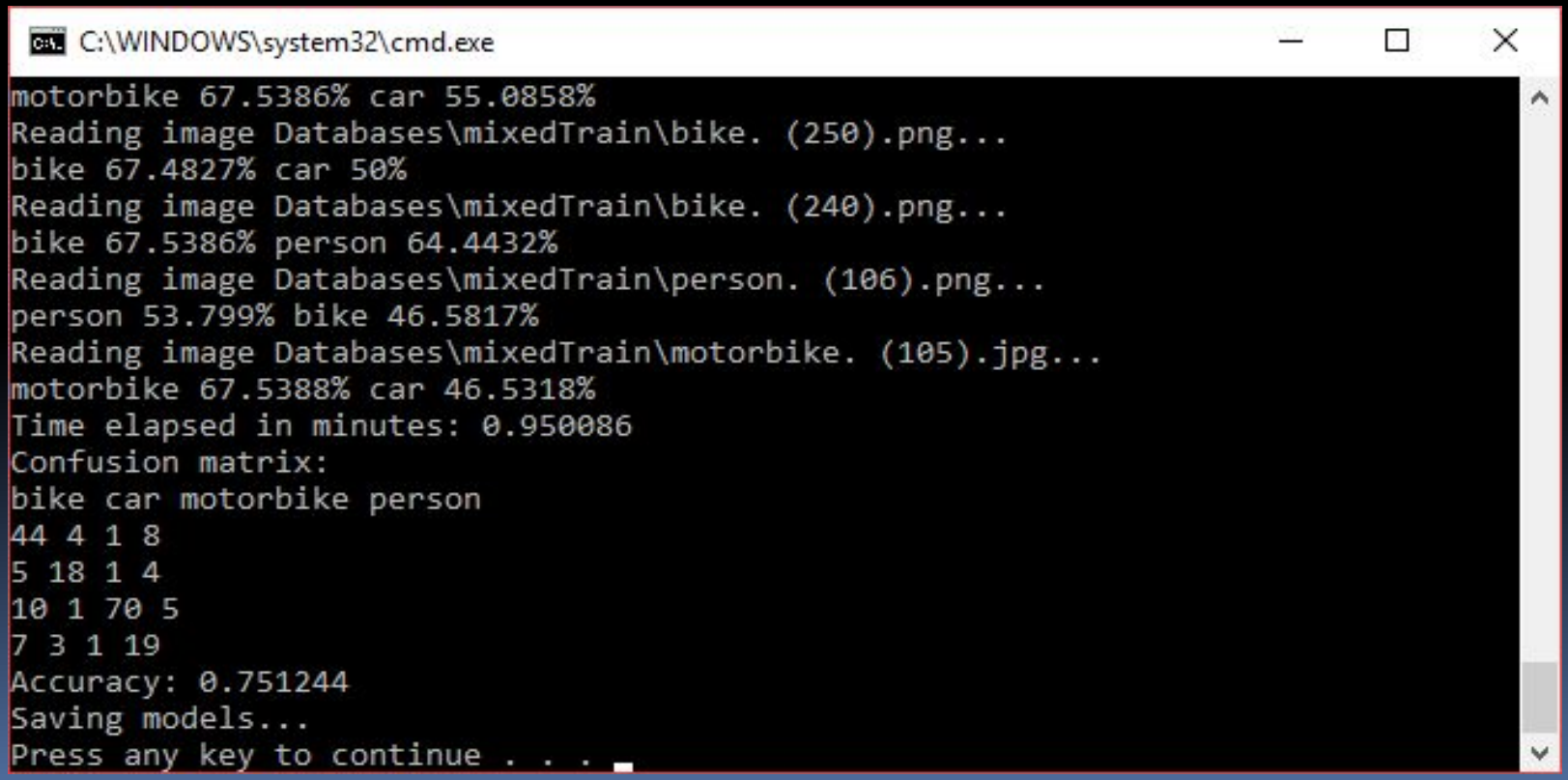
Accomplishments:

Started from an example OpenCV application and implemented the following:

- Rebuilt OpenCV library, enabling to compile in release mode, reducing execution time significantly.
- Functionality to load the serialized neural network from file, allowing us to create a large network without having to rebuild each time.
- Resolved inability to change feature detectors, can now switch between SURF, SIFT, and KAZE.

Accomplishments: Cont.

- Added functionality to the confusion matrix function to support a n x n matrix so that we could train and test a neural network with more than two classes.



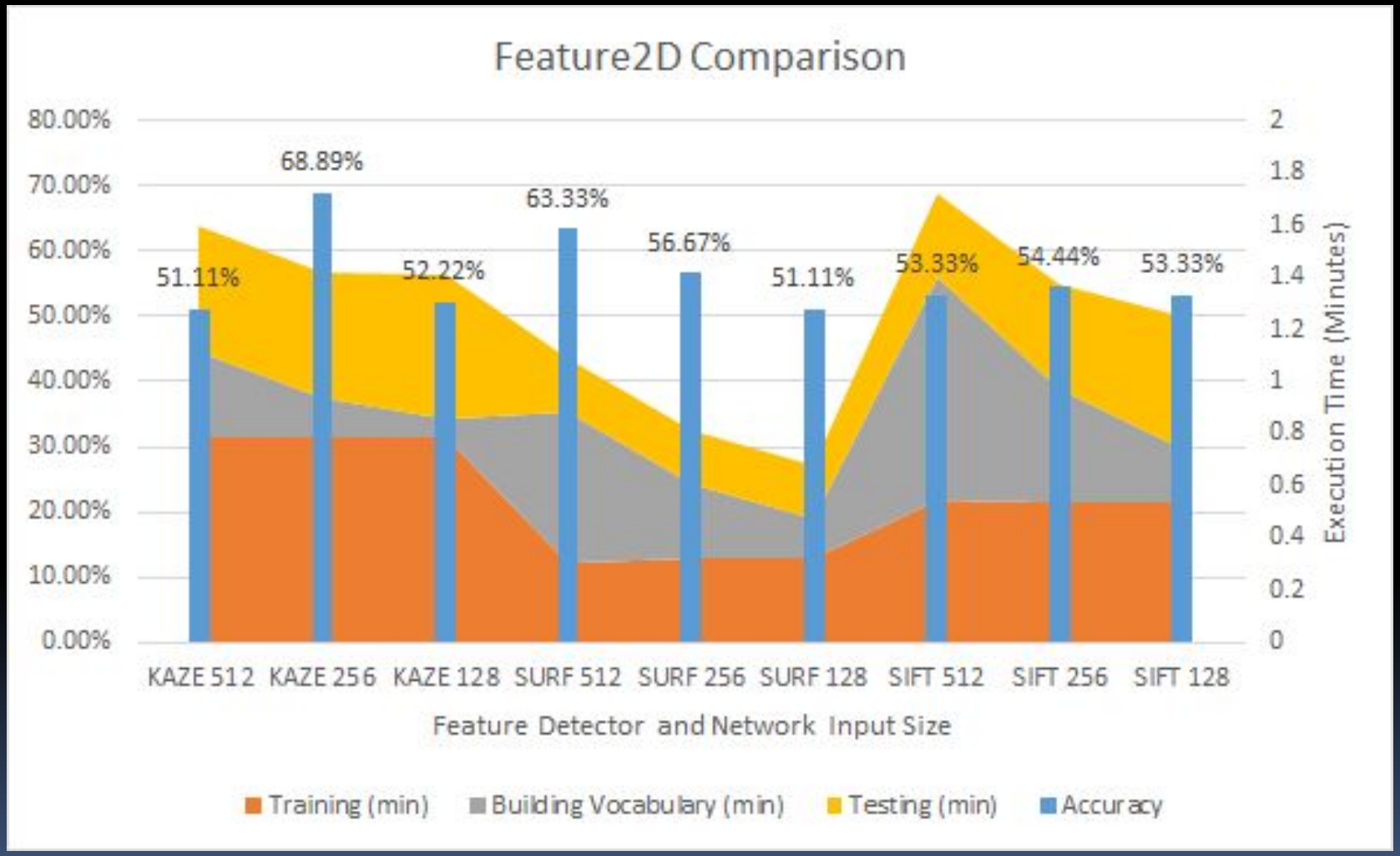
Accomplishments: Cont.

- Evaluate the raw image output of the neural network into a user-readable, color coded, confidence level.

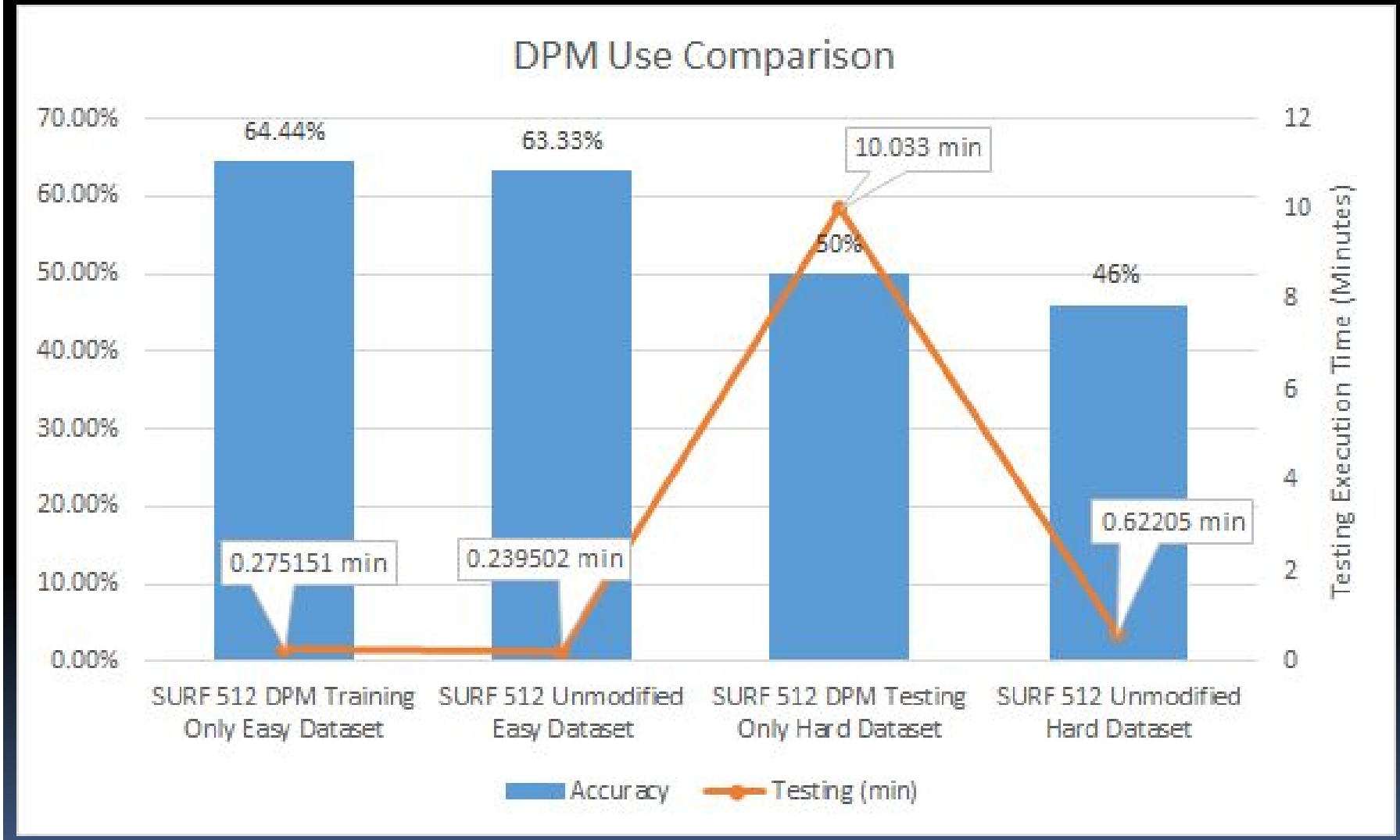
Accomplishments: Cont.

- We implemented functionality to use a Deformable Part Model for use in both training and testing with a provided pre-trained cascade xml model.

Results:


Feature2D Comparison
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Execution Time (Minutes)
51.11%
68.89%
52.22%
63.33%
56.67%
51.11%
53.33%
54.44%
53.33%
KAZE 512
KAZE 256
KAZE 128
SURF 512
SURF 256
SURF 128
SIFT 512
SIFT 256
SIFT 128
Feature Detector and Network Input Size
Training (min)
Building Vocabulary (min)
Testing (min)
Accuracy

Results:


DPM Use Comparison
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
12
10
8
6
4
2
0
Testing Execution Time (Minutes)
64.44%
63.33%
50%
46%
0.275151 min
0.239502 min
10.033 min
0.62205 min
SURF 512 DPM Training Only Easy Dataset
SURF 512 Unmodified Easy Dataset
SURF 512 DPM Testing Only Hard Dataset
SURF 512 Unmodified Hard Dataset
Accuracy
Testing (min)

Lessons Learned:

- We expected that DPM optimization would improve the speed of testing the image sets. However, the DPM algorithm appears to be unoptimized and or the current DPM models are poor.
- Lack of data specific to our needs largely shaped the direction of the project.
- Finding example code that is compatible with the latest version of OpenCV.

Future Work:

Implement more finite detection of features:

- Identify make, model, year, and color of a car.

Questions?

?

Credits

- OpenCV repository with example code for dpm use.
- Git repository by Andrews Sobral, vehicle detection using haar cascades car xml.
- OpenCV documentation on how to use cascades classifier, was useful for video set up.
- Blog by Abner Matheus, our project based on improving this neural network.