

Project 1 vision and objects recognition

Christian Dansereau

Abstract—The objective of Project 1 is to explore the various tools found in the context of vision and object recognition. During the project we are face with various problems typical of vision that need to be address and that are discuss in the following pages.

Keywords—Vision, Object recognition, IEEE.

1 INTRODUCTION

THE objectives of the projects are the following: 1) identify the objects in motion in a scene and extract some meaningful features. 2) classify the nature of the moving objects namely if they are pedestrians or cars. 3) Be able to track the moving objects and there respective trajectories.

1.1 Public code and data

The code and data used in this experiment are available on a GitHub repository at the following URL: https://github.com/cdansereau/vision_or. A IPython Notebook is also provided with most of the figure generation scripts.

2 METHOD

The project was realized in python with the following libraries: scipy [1], scikit-learn [2], matplotlib [3] and scikit-image [4].

2.1 Identification of objects in motion

The dataset used for classification was composed of 150 images (webcam at <https://www.cs.dal.ca/cams/university.jpg>) of those we use 30 images to tune the parameters of the feature extraction algorithm. The median of the first 30 images was used to compute the background.

Algorithm 1 was use to extract the moving objects from the background and remove artefact of the image. We found that repeting the Algoritm 2 multiple time was improving the mask substantially in order to remove small objects fused the relevant objects and make more consistent blobs for object.

```
Data: rgbImg, background,
Result: imgLabels
im = mean(rgbImg) %Grayscale image;
imDif = im - background;
mask = adaptativeThreshold(imDif);
mask = binaryClosing(mask, square(2));
mask = binaryDilation(mask, square(4));
imgLabels = getLabels(mask);
imgLabels =
removeSmallObj(imgLabels,100);
for i=range(1,5) do
    | imgLabels =
    | maskCorrection(imgLabels);
end
return imgLabels;
Algorithm 1: extractAllLabels
```

2.2 classification

The features used in the classification are based on resized RGB images 32x32 pixels extracted from the the centroid of the detected object. Two example of the resulting images from Algorithm 3 can be seen in Figure 1.

We choose to use SVM (Support Vector Machine) [5] as our classifier since it's a fast and

• Mr. Dansereau is with the Department of Computer science, University of Montreal, Montreal, CA.
E-mail: christiandansereau@gmail.com

Data: imgLabels
Result: imgLabels
mask = $imgLabels > 0$
binaryDilation(mask, square(10));
mask = putBorderZero(mask);
binaryErosion(mask, square(10));
imgLabels = getLabels(mask);
imgLabels =
removeSmallObj(imgLabels, 100);
return imgLabels;
Algorithm 2: maskCorrection

Data: originalImage,
Result: imagePatch
limg = largest side of originalImage;
sqImage = square image limgXlimg;
imagePatch = resample(sqImage);
Algorithm 3: Features extraction

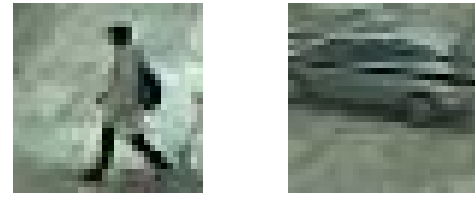
robust state of the art classification algorithm. Some basic preprocessing was performed on the patch image of 32x32 pixels. We first vectorized the RGB images and removed the mean and variance from each of the features.

In order to find the most adapted hyper-parameters (C and Gamma) for our classification problem we choose to use a grid search approach. The grid search and the final classification were evaluated using a stratified cross-validation due to the fact that the number of example available were unbalanced.

2.3 Tracking

Data: objects, image,
Result: mask of the image with object in motion
initialization;
read current;
if understand **then**
| go to next section;
| current section becomes this one;
else
| go back to the beginning of current section;
end

Algorithm 4: Tracking objects



(a) Pedestrian

(b) Car

Fig. 1. 2 features side by side of 32x32 pixels in RGB. a) is a pedestrian and b) a car

3 RESULTS

3.1 Part1: feature extraction

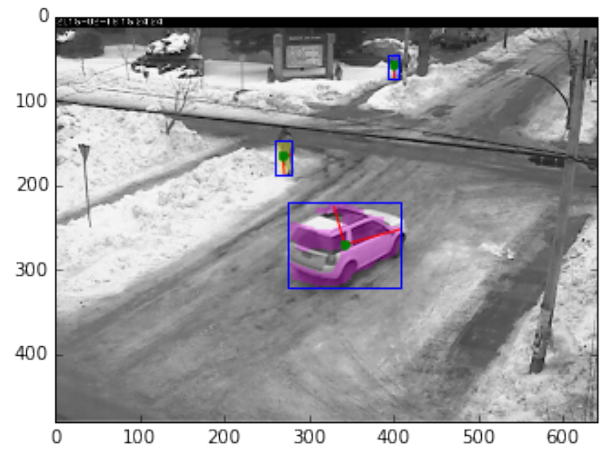


Fig. 2. Features extraction: the box represent the moving object identification and the some of the feature extracted (angle of the the major/minor axis of the object and centroid).

From the 150 images analysed the algorithm was able to identify 238 moving objects (93 cars and 145 pedestrians). Figure 2 show the object identified by the algorithm and there respective bounding box. We also include an overlay of the actual object detected, has we can see from the car some areas are not completely covered.

3.2 Part2: classification

We were able to achieve an accuracy of 90% (+/- 0.18) estimated with a 10 fold cross-validation with stratified test and validation sets. Figure 3 show an example of the classification in a scene with 3 moving objects 2 pedestrian (in red) and one car (in yellow).

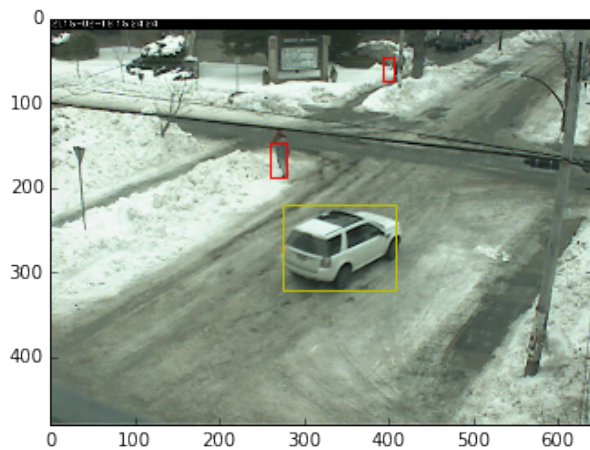


Fig. 3. Classification: the box represent the moving object identification, the color represent the classification result of the object nature (yellow=car, red=pedestrian).

3.3 Part3: tracking

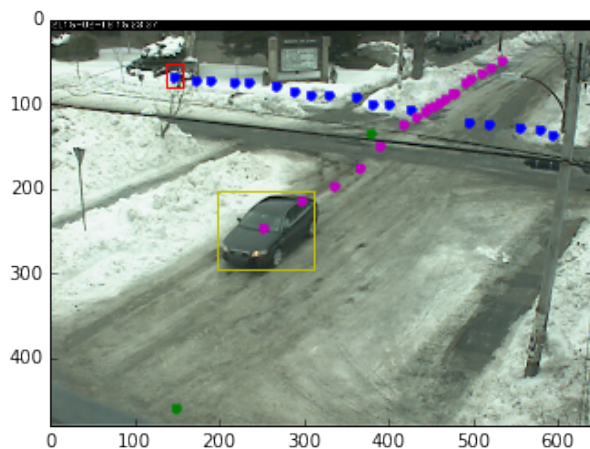


Fig. 4. Final tracking result: the box represent the moving object identification, the color represent the classification result of the object nature (yellow=car, red=pedestrian) and the color dots represent the unique path of a particular moving object in the scene.

Figure 4 show the successfully tracking of two moving objects in a scene, in this particular case the two object cross the path and fused at the intersection point but the algorithm was able to overpass this difficulty by looking at previous frames and the angle of the previous trajectories.

4 CONCLUSION

This project introduce us to the various problematic than one can face in the context of vision and tracking. We were able to achieve the three requirements of the project namely 1) the identification of mobile object and there features, 2) the classification of the object in two class (pedestrian or a car) and finally 3) the tracking of the object across the scene. We also made our software able to identify a trajectory and be able to resolve interference due to two objects crossing path.

REFERENCES

- [1] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed 2015-03-15]. [Online]. Available: <http://www.scipy.org/>
- [2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [3] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [4] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014. [Online]. Available: <http://dx.doi.org/10.7717/peerj.453>
- [5] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: <http://dx.doi.org/10.1023/A%3A1022627411411>