On-the-fly Image Classification to Help Blind People

Dalal Khalid Aljasem
Dept. of Computer Science
Middlesex University
London, UK
Email: DA716@live.mdx.ac.uk

Michael Heeney, Armando Pesenti Gritti, Franco Raimondi
Dept. of Computer Science
Middlesex University
London, UK
Email: {M.Heeney,A.PesentiGritti,F.Raimondi}@mdx.ac.uk

***Abstract*—In this paper we present an affordable solution to help blind people navigate unknown environments. Our solution performs image classification on a Raspberry Pi and provides feedback to users by means of vibration motors to signal the presence of an obstacle in a given direction. The training phase is performed off-line, while the on-line phase can classify an image in 1.12 seconds on average. We provide an evaluation using several thousands images, showing that we can achieve a precision of 79% and a recall of 79%. All our code and the hardware design files are released open source.**

I. Introduction

Computer vision, and in particular object detection and image classification, are mainstream research areas that have attracted a huge interest in recent year both in academia and in industry. Big corporations like Google have invested heavily in deep learning [1] and other cutting-edge techniques, with applications in nearly all domains, from autonomous cars to robotics and UAVs.

In parallel, the progress in hardware technologies has made it possible to develop compact and powerful platforms, such as the BeagleBone Black (http://beagleboard.org/bone), the Raspberry Pi (https://www.raspberrypi.org) and ODROID (http://odroid.com), whose computing capabilities are comparable to desktop machines of less than 5 years ago. While these platforms may not be able to operate complex unsupervised learning algorithms, they are nevertheless powerful enough to perform a binary classification of images using state-of-the-art classification algorithms.

In this paper we show how it is possible to employ a Raspberry Pi and its dedicated camera to classify images in walkable / non-walkable categories after an initial training phase. A number of previous approaches to support navigation for blind people have employed distance sensors (either sonar or laser), while some recent approaches are investigating the use of mobile phones acting as a bridge between the user and on-line, remote servers. Our aim in this paper is to present a solution that can be implemented using off-the-shelf portable hardware and software libraries; in particular, we connect the Raspberry Pi and its camera to a portable battery and to three vibration motors, assembling all the parts in a wearable belt that can signal the presence of obstacles in the left/right/middle direction. All the software and the models described in this paper are available at https://bitbucket.org/mdxmase/intenv2016.

The rest of the paper is organised as follows: we discuss related work in Section II and we introduce relevant background on classifiers in Section III. We describe the overall architecture of the system in Section IV and we present its evaluation in Section V.

II. Related work

Our work falls in the general area of *image analysis* and in particular we are interested in performing an *image classification* task, that is automatically labelling an image as belonging to one of a specified set of classes.

If we focus on the specific task of helping visually-impaired people, in the past decade several attempts have been made at developing technology to support them. A low-cost solution using a sonar distance sensor is presented in [2]. Infra-red sensors can be used as well, as described in [3]. In all these cases there is not attempt at "understanding" a scene: an alarm is raised when obstacles are detected at a distance below a certain threshold.

The approach presented in [4] is perhaps closer to our work in that it employs a Microsoft Kinect to perform depth analysis and translates depth information to auditory signals, but it is applicable only to indoor environment given the sensor limits. Along similar lines, the tool described in [5] operates on the same principles of translation from depth of field to auditory signals. What differentiates our approach is the use of image classification that can potentially detect also obstacles and non-walkable areas containing dirt that do not present a substantial depth.

An approach using image recognition is presented in [6]. A 2010 survey of wearable devices for obstacle avoidance for the blinds is available in [7]. At the time of writing, the closest approach based on AI techniques seems to be Baidu DuLight [8], which combines a micro camera with a mobile phone and bluetooth headsets. Our solution is less ambitious as it only aims at notifying of the presence of obstacles, instead of performing a full semantic analysis of the scene. This allows to keep the cost of our solution down.

III. Background on classifiers

An image classification task, as the one we are interested to perform, can be carried out using two very different high level

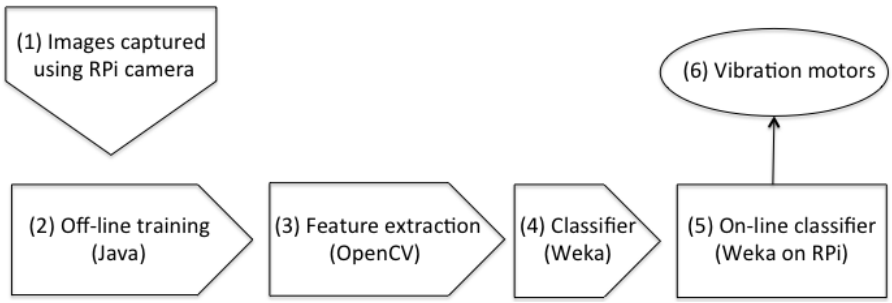

Fig. 1: Overall system architecture

approaches. The first one is to explicitly define a model (e.g. color, shape, luminosity, etc.) of the classes we are considering and then label the image with the best fitting model. This approach is feasible only for the cases when it is possible to easily define such characterising features (e.g. the task of identify a ball). In our context, it is impossible to a priori explicitly define which characteristics of the image can allow us to discriminate walkable/non-walkable parts of the environment. For this reason we decided to adopt a machine learning approach, in the specific a supervised learning approach. Given a sufficient amount of labeled data, the algorithm will learn to discriminate between the two classes we are considering without the need of manually engineering a model.

There exists several supervised learning algorithms with different characteristics. We have decided to focus our attention to three in particular and to compare their performance on our task:

1) *Naive Bayes*: very simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. It is our performance baseline.
2) *AdaBoost* [9]: a meta learning algorithm, that is the output of weak learning algorithms is combined into a weighted sum that represents the final output of the boosted classifier. It is frequently used in image classification tasks as it can be less susceptible to the overfitting problem.
3) *Support Vector Machines (SVM)* [10]: a set of supervised learning methods used for classification, regression and outliers detection. We selected this algorithm since its high expressive power and its effectiveness also in high dimensional spaces, like the 2D image space we are considering.

In order to reduce the dimensionality of the input of our learning algorithm and to obtain a representation invariant to photometric (e.g. illumination, shadowing, contrast, etc.) and geometric transformations, we compute the *Histogram of Oriented Gradients (HOG)* [11] feature on the image acquired by the camera.

IV. Architecture of the system

In this section we describe the overall architecture of the systems, including both hardware and software components and the external libraries employed. The high-level architecture is depicted in Figure 1. All the code, the experimental results, the configuration parameters and the trained model are available from https://bitbucket.org/mdxmase/intenv2016.

At the highest level, the workflow of our approach can be described as follows:

1) In the initial training phase, images are captured on the Raspberry Pi from the same position that will be used in the final product (waist level). We employ the utility `raspivid` to capture a continuous video (full HD) and the utility `ffmpeg` to extract single frames. This is an operation that is performed only once and on the paths that we want to classify, as we expect our application to be used on repeated paths (e.g. home to work).
2) The images are transferred to a laptop machine where we perform manual classification. We have implemented a GUI in Java that automatically reads and splits the images in three sections (vertically) for left-middle-right components using OpenCV (http://opencv.org/), and then asks the user to manually classify them. The sliced images are stored in separate directories for "walkable" and "non-walkable" examples. See the code in `Label.java` for additional details.
3) We use OpenCV to extract HOG (Histogram of Oriented Gradients) descriptors for the images that have been classified in the previous step. We refer to the file `Feature.java` for details on our choice of parameters. The code in `Preprocess.java` extracts the features from each image and it writes them as a line in a CSV file, with an additional field for the walkable/non-walkable classification. The results of the manually classified images are available in the file `code/img/dataset.csv`.
4) The CSV file generated in the previous step is fed into a data mining / machine learning tool. In our work we employ Weka [12], as it is open source and it supports multiple platforms, including the Raspberry Pi. As described in the following section, we have evaluated several classifiers, obtaining the best results with Support Vector Machines (SVM). The output of this step is a trained classifier, called a *model*, that is used in the on-the-fly phase. We make our models available in the files

with extensions `.model`.

5) The on-the-fly classifier is a Java application that interacts with the camera on the Raspberry Pi. Specifically, the application implemented in `Predict.java` loads in memory an image taken with the Raspberry Pi camera. It then uses OpenCV to slice the image in three vertical sections. For each section, the HOG features are computed (see `Feature.java`) and are then used as input when invoking an instance of the Weka library, together with the model generated above. For each of the three slices, the result of the invocation of Weka in a number between 0 and 1, expressing the certainty that a slice is *not* walkable.
6) The final component of our architecture is the connection with vibration motors. The hardware circuit we employ is depicted in Figure 2: three GPIO pins drive three transistors which, in turn, are connected to three vibration motors (model 1201 from Adafruit) using three diodes (notice that the motors need to be powered separately, as they require approximately 100mA each). The code in `Predict.java` drives these motors: when the output from Weka is greater than 0.5, i.e., the picture is classified as non-walkable and the corresponding motor is activated.

Some excerpts from the class `Predict.java` are shown in Figure 3, showing how the various libraries interact in the `run()` method of the class. Please see the comments in the code reported in the figure for additional details.

V. Evaluation

Accuracy of the classification step can be performed using Weka in step (4) above. For our purposes, we have collected images on five public footpaths around Middlesex University (both indoor and outdoor), recording a total of approximately 15 minutes of videos and extracting approximately 3000 still frames in varying conditions of light and for different surfaces. Possible obstacles include: dirt, rubbish bins, potholes, water puddles, etc. We have classified these images manually, a process that required approximately 2 hours. The samples employed comprised approximately 50% of walkable images and 50% of non-walkable images.

We have evaluated three possible classifiers that are available in Weka: a simple Bayesian classifier, Adaboost and SVM. Generating the classifier and running the tests took approximately 10 minutes on a standard laptop machine (Quad-core Intel i7 2.5GHz 8 GB of RAM).

Tables I, II and III report, respectively, the results for these classifiers.

These results show that Adaboost performs slightly better than a naive Bayesian classifier. On the other hand, SVM clearly outperforms all the other approaches. We refer to the files available on-line for more details about these experiments and for all the configuration parameters employed. Figure 4 shows the Receiver Operating Characteristic (ROC) curves for the three classifiers. Each curve plots the True positive rate (i.e., the recall) on the y-axis as a function of the False positive rate on the x-axis. A random guess would correspond to the diagonal, and any curve above the diagonal is an improvement over the random guess, with larger areas corresponding to better classifiers. Figure 4 shows visually that the SVM is clearly superior to the other two options. Notice that in our evaluation we have considered images from *different* paths. While the generated model cannot be taken as a "universal" classifier for paths, nevertheless it is not limited to a single path but to 5 paths with different characteristics (concrete, carpet, off-road path). Accuracy could be improved by focussing on a single kind of path, possibly configurable by the user. We leave this for future work.

Class	Precision	Recall	ROC area
Walkable	0.77	0.57	0.73
Non-walkable	0.55	0.76	0.71
Average	0.68	0.64	0.73

TABLE I: Naive Bayesian classifier

Class	Precision	Recall	ROC area
Walkable	0.79	0.65	0.76
Non-walkable	0.60	0.75	0.76
Average	0.71	0.69	0.76

TABLE II: Adaboost

Class	Precision	Recall	ROC area
Walkable	0.79	0.87	0.85
Non-walkable	0.79	0.67	0.85
Average	0.79	0.79	0.85

TABLE III: SVM

Another important aspect that we have evaluated is the performance of our approach on the actual hardware. We have modified the code capturing and classifying an image on the Raspberry Pi to compute the time elapsed from invocation to successful completion of a full classification. This can be achieved by instrumenting the file `Predict.java`. We ran a sequence of 200 images classifications and we obtained an average classification time of 1.12 seconds with a variance of 0.1 seconds. Finally, our initial experiments with a 9000 mAh battery show that the approach has an autonomy of approximately 3 hours.

VI. Conclusion

In this paper we have presented a solution that allows on-the-fly classification of images in walkable / non-walkable categories using open-source libraries and using Raspberry Pi as the running platform. We have performed an initial training phase using a desktop machine and we have obtained promising results using SVM as a classifier. We then deployed the classifier on the Raspberry Pi and we obtained a more than adequate performance of one image classified per second. The overall cost of the whole solution is below £50. Therefore, we argue that our approach provides an affordable support mechanism for visually impaired people, especially in the case of paths that are repeated daily, such as home to work.

For the future, we plan to investigate more efficient libraries to replace Weka, possibly implementing our own bespoke

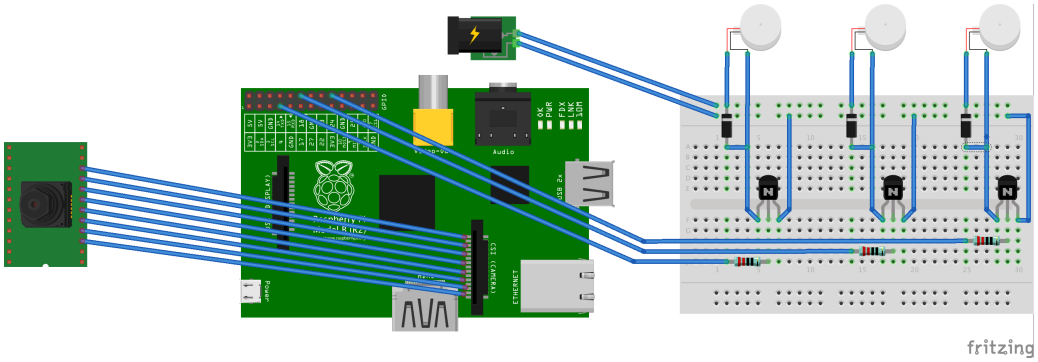

Fig. 2: Hardware circuit

```
// We import libraries for GPIO pins, for the Weka
// classifier and for OpenCV, which is used both to
// capture and to extract HOG features.
import com.pi4j.io.gpio.*
import weka.classifiers.functions.LibSVM;
import java.io.FileInputStream;
import org.opencv.highgui.VideoCapture;
// [...]
public class Predict {
  // [...]
  public void run() {
    // [...] Set the pins for the motors:
    GpioController gpio = GpioFactory.getInstance();
    GpioPinDigitalOutput motor1 =
    gpio.provisionDigitalOutputPin(RaspiPin.GPIO_04);
    // [...] Prepare the camera and read the image
    VideoCapture camera = new VideoCapture(0);
    camera.read(image1);
    // [...] Extract the features:
    Mat feature1 = Feature.computeFeature(image1);
    // [...] Instantiates the model with the file created
    // off-line:
    LibSVM model = LibSVM.read(
       new FileInputStream("svm.model")));
    // [...] Classify the image:
    double[] P1 =
            model.distributionForInstance(feature1);
    // [...] Activate vibration if appropriate.
    if (P1[0] > 0.5) {
      System.out.println("Not walkable");
      motor1.high();
    }
}
```

Fig. 3: Excerpts from the class Predict.java

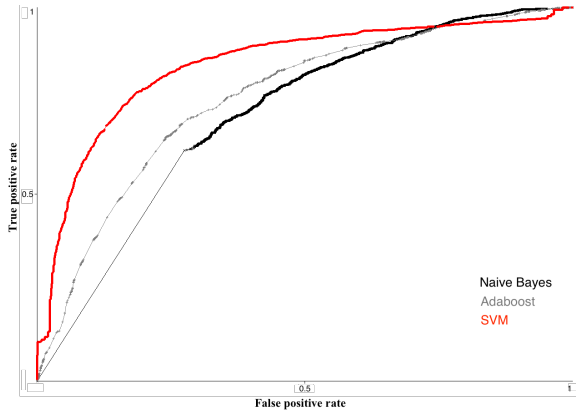

Fig. 4: ROC curves

classifier. We are also investigating more energy-efficient solutions using different kinds of motors and additional input mechanisms so that a user can pre-select the kind of path to be walked and by increasing the number of motors so that each image could be classified in 5 or more areas, instead of 3.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

[2] "Smartcane device," http://smartcane.saksham.org/, accessed: 2016-03-10.

[3] A. S. Al-Fahoum, H. B. Al-Hmoud, and A. A. Al-Fraihat, "A smart infrared microcontroller-based blind guidance system," *Active and Passive Electronic Components*, vol. 2013, 2013.

[4] M. Brock and P. O. Kristensson, "Supporting blind navigation using depth sensing and sonification," in *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, ser. UbiComp '13 Adjunct. New York, NY, USA: ACM, 2013, pp. 255–258. [Online]. Available: http://doi.acm.org/10.1145/2494091.2494173

[5] "artificialvision," http://www.artificialvision.com/, accessed: 2016-03-10.

[6] G. Sainarayanan, R. Nagarajan, and S. Yaacob, "Fuzzy image processing scheme for autonomous navigation of human blind," *Applied Soft Computing*, vol. 7, no. 1, pp. 257–264, 2007.

[7] D. Dakopoulos and N. G. Bourbakis, "Wearable obstacle avoidance electronic travel aids for blind: a survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 1, pp. 25–35, 2010.

[8] "Baidu dulight," http://www.wired.com/2016/01/2015-was-the-year-ai-finally-entered-the-everyday-world/, accessed: 2016-03-10.

[9] Y. Freund and R. Schapire, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[10] 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:1022627411411

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, June 2005, pp. 886–893 vol. 1.

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278