HCI Course Exercise: Level 10 & 11
Image labelling application

Michal Dziemianko and Mark Wright

September 21, 2012

1 Introduction

In this practical exercise you will be designing, building and evaluating an application in which users will identify, mark and label objects in images. This application is meant to be a help for researchers and developers to quickly produce datasets of fully annotated images.

To be effective the application should allow quick editing and reviewing of image labels. It has to be easy to use, so volunteers can easily learn to use it, and become productive with only minimal training. You should complete the exercise in pairs. This will allow you to divide the work, and discuss design decisions. Please find a suitable partner (at the same course level), and contact us immediately if you cannot.

The demo should be implemented in Java (preferably using Swing). We provide a simple example which you might extend. In case you do not know Java or want to use another language please contact us immediately. Remember however that the application has to compile and run in DICE environment (therefore please no C# as it does not run well on DICE).

There are 2 parts to the practical:

- *design and implementation* - levels 10 and 11, deadline: 19th October at 23:59 (11:59PM) BST;
- *evaluation* - levels 10 and 11, deadlines: 23rd November at 16:00 (4:00PM) BST for UG4, 30th November at 16:00 (4:00PM) BST for MSc;

You should submit your working application and a report explaining the design process and decisions, and an evaluation report using the submit command. The details are given below.

2 Background

By image annotation we mean segmentation of an image into a set of regions (also called blobs) and attaching tags or labels (either being object names or abstract terms) to them. We will also use the term labelling as a synonym of annotation. *LabelMe* (http://labelme.csail.mit.edu) is an effort to build a large database of annotated images by collecting contributions from many people. You can contribute yourself by visiting the annotation tool at http://labelme.csail.mit.edu/tool.html (please remember it is an ongoing project so read the guidelines at http://labelme.csail.mit.edu/guidelines.html first, and do not leave any trash behind). However, the LabelMe tool has a certain number of usability problems.

In this assignment you will build a more user-friendly, standalone application, that can be used to produce similar image databases. You can utilise an example interface we provide, or build your own from scratch. *Remember that your application needs to be easy to compile and run directly from the command line in Linux DICE environment.* It is best to provide a simple run.sh script in the main directory. README files are welcome in case something more needs to be done. Please do not use any external libraries or software that is not installed as standard on DICE machines. Contact us in doubt.

2.1 Example interface

You can download the example interface by executing the following command on any DICE machine:

```
#> cp -r /afs/inf.ed.ac.uk/group/teaching/hci/2011-12/hci-practical .
```

Do not forget the final dot. This will copy the practical directory into your home directory. The directory contains several files:

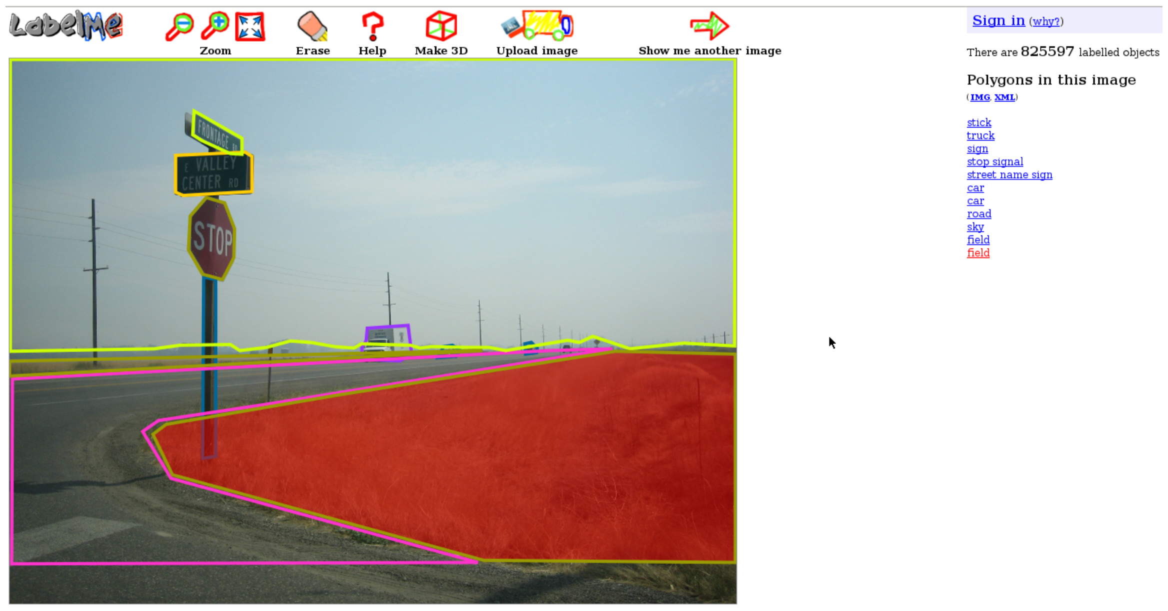

Figure 1: LabelMe tool window showing an image, assigned labels and highlighted area corresponding to one of the labels. Please note that it might be hard to tell if an object (such as sky) is already labelled. It is also impossible to browse images, and then resume your work on the last image. Instead a random image is shown every time the tool is open.

- *run.sh* - a simple script that compiles and run the application.
- *hci* - the main directory of the Java sources, with:
  - *ImageLabeller.java* - main class that sets up the GUI and executes the program.
  - *ImagePanel.java* - class handling the editing of the image labels.
  - *utils* - a directory with very simple *Point.java* class to handle points.

The program is very simple. ImageLabeller sets up a GUI - the main window containing an image panel (instance of ImagePanel class) and toolbox (currently implemented directly in ImageLabeller - you might want to move it to a separate class if it gets too crowded). The ImagePanel handles display of the loaded image, and editing capabilities by capturing all mouse events on its area. The mouse clicks are added as vertices to the current polygon, and upon clicking the new object button this polygon is transferred to the list of completed polygons, and a space for a new one is reserved. As for now only one image can be loaded at a time, and has to be specified as a command line parameter to the program (see run.sh script).

3 Exercise

3.1 Part 1: design and implementation

In this part you will build the application for labelling the images. The requirements are to:

- Design a user friendly interface, that is:
  - Perform task analysis;
  - Use techniques presented in the course to design the interface;
  - Explore alternative designs and perform brief comparable analysis.
- Implement the designed interface, that allows:
  - create, edit, delete and review image labels along with their corresponding areas;
  - open any image from a specified directory;
  - save and load labels generated for a given image.

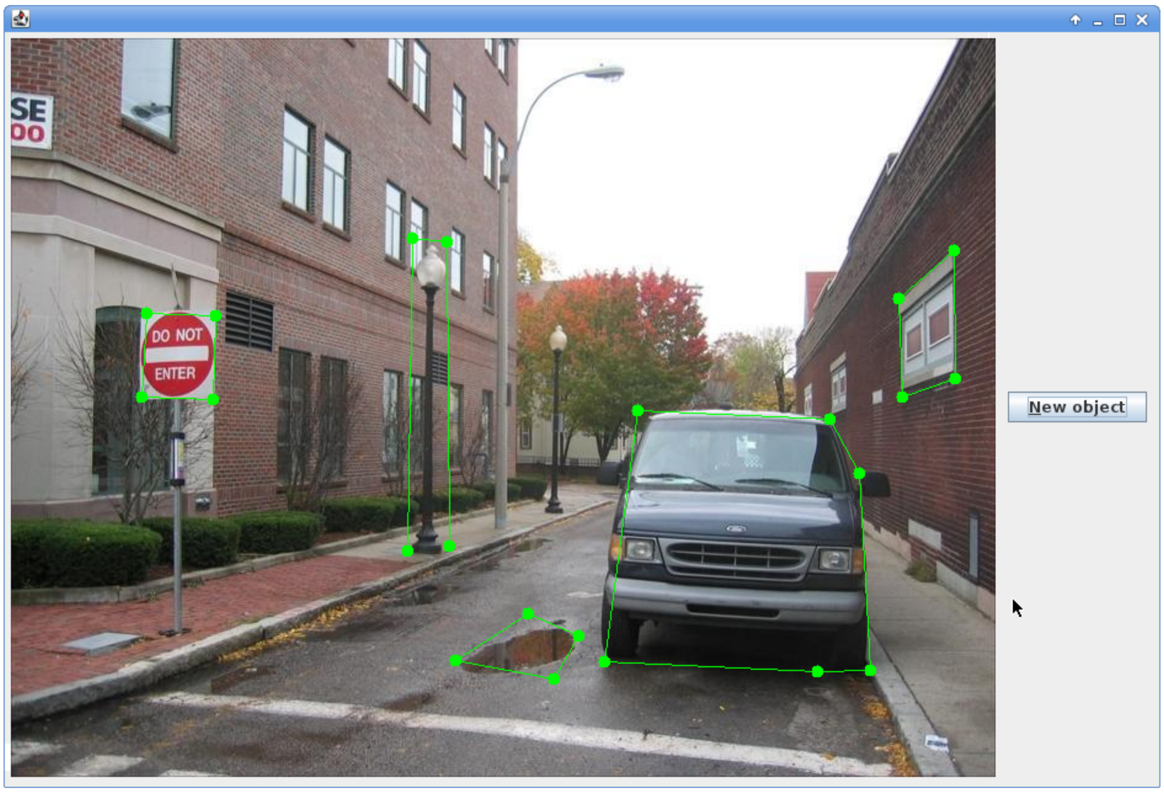

Figure 2: Example interface showing the image, some completed polygons, and currently edited polygon (lamp post in the center).

This is a minimum you must do for a pass mark for part 1. You can extend your application in many ways such as easy browsing of images in a given directory. All extensions that will ease users life are welcome (and give you some extra points). **Please note that the provided example does not full-fill all of these requirements**.

You should submit the whole code needed to run the application, any scripts and a report of around 3000 words (this is not a strict limit, but any report substantially longer might be penalised). This should describe the design process, justify your decisions (e.g. explaining features or solutions you considered), and assess the strengths and weaknesses of what you have produced. In a short report it is critical to write concisely, so do not waste words on re-describing the assignment. Presenting information in point form is acceptable, and where possible let diagrams or pictures provide the information. Please note that we expect MSc students to produce more formal reports showing evidence of some background reading and formal assessment of the provided sample interfaces.

To submit, *one of the pair* should use the electronic submission command like this:

```
submit hci 1 hci-practical.zip
```

The zip file must contain all the JAVA code which is needed to launch your GUI. All other necessary files must also be contained within the zip file (but please do not submit directories with sample images). Ideally compilation will be as simple as executing javac *.java or running run.sh script in the unzipped directory - detailed instructions must be provided in form of README file for anything more complex.

The report must be in PDF or DOC format and on the first page should give the matriculation number of both members of the pair.

We will normally allocate both people the same mark. To notify us of an alternative allocation you have agreed, please put a table on the first page of the report with a percentage split. However, please only do this in extraordinary cases, and be aware that a clever design idea in this practical may have more impact on the mark than many hours of coding, so allocation by hours spent may not be the most appropriate.

3.2 Part 2: User Evaluation

We will randomly assign each pair of students with the interface system designed by another pair. The task is then to evaluate this interface with users in one but preferably both (mandatory for MSc) of the following ways:

- A think-aloud or cooperative evaluation (see Dix et al 9.4.3) with at least one naive user, who to the extent possible, should closely resemble a member of the target audience (i.e. someone from outside informatics). In this you should gather detailed feedback on how the system is used and any problems that arise. You should give the user a set of structured tasks and prompt them so as to ensure they explore the whole system.

  This is the minimum you must do for a pass mark on Part 2 for UG4.

- Design a set of objective measures and closed questions and apply them to quantify the results for subjects who freely interact with the system (i.e. as might happen if they were volunteers, so possibly only for a few minutes or not using all features). For example, how long do subjects play with the system? How many of the features do they discover? How many errors do they make? How do they rate it? Have they learned anything? Ideally these subjects should at least be naive to the aims of the practical (i.e. not fellow students on the HCI course). You should aim to make this evaluation fairly short and easy for the subjects to complete (e.g. a questionnaire that they might actually be willing to fill in if they were on-line).

  This part is optional for UG4, but mandatory for MSc level.

- Formally evaluate sample interfaces, and compare them with your application. Prepare a short plan for the next development iteration of the evaluated application reflecting the changes and improvements to the issues discovered as a result of the evaluation.

  This part is optional for UG4, but mandatory for MSc level.

Moreover remember that **MSc students are required to produce their report alone** in contrast to UG4 students who are allowed to write in pairs. However, we do encourage collaboration in the form of discussion (but please write your report on your own!).

Note: your evaluation of the design will in no way affect the mark that the other group receive on part one.

To submit:

```
submit hci 2 report.pdf
```

4 Marking guidelines for part 1

Both students receive the same mark, unless otherwise specified.

1. Basic criteria for marking the interface
   - Interface is complete and working
     - User friendly way of opening an image
     - User friendly way of editing, adding and deleting image labels and selecting corresponding image areas
     - User friendly way of saving and loading the label set
   - Interface is appropriate for target users
     - The interface is easy to use
     - The interface allow fast processing of large number of images
   - Interface should follow basic HCI principles and do not break major rules
     - The interface has a clear layout
     - Usage of affordances
2. Additional criteria
   - Design rethought from scratch (rather than using/copying provided examples)
   - Attempt to provide some form of help

    - Attention paid to guiding the user
    - Some thought of how to make the system easy to extend
    - Design extended by additional functionality
    - Anything special (e.g. ingenious ideas)

3. Report
    - Clearly presented, well structured
    - Formal requirements analysis/task analysis/interaction analysis
    - Clearly presented design process (both engineering and interaction-centred approach are acceptable)
    - Critique of LabelMe and/or sample interface
    - Evidence of considered alternative solutions
    - Evidence of iterative development
    - Evidence of explicit and consistent use of usability guidelines and/or choice of good metaphor
    - Evidence that users were involved in design stage
    - Evidence of decisions being grounded in HCI principles
    - Formal analysis of strengths and weaknesses of developed system

5 Marking guidelines for part 2

1. Basic criteria
    - Cooperative/think-aloud evaluation (level 10 & 11)
        - Justification of tasks to be performed;
        - Clear presentation of presented data;
        - Analysis of the data.
    - Quantitative analysis (optional for level 10, mandatory for level 11)
        - Justification of evaluation metrics;
        - Presentation of collected data;
        - Analysis of the data.
    - Interface refinement (optional for level 10, mandatory for level 11)
        - Proposed changes and their justification;
        - Discussion of impact;
        - Discussion of possible alternatives.
2. Additional criteria
    - Critical comparison with solutions available in other applications (e.g. your interface, drawing tools etc.);
    - Formal analysis of the interface (e.g. with regards to design heuristics);
    - Analysis of impact of design errors, if any;
    - Anything special (e.g. ingenious ideas).
3. Report
    - Clearly presented, well structured;
    - Evidence of proper background reading.