HCI Project Report

Andrei Manolache (s092233) and Keith Robertson (s0940608)

1. Introduction

The Image Labeller application provides the user with the functionality to identify and label objects in images. The primary role of the application is to provide an intuitive interface which will enhance the user‘s interaction with the program making it easier for them to complete their goal.

This report will cover tools used and describe the principles behind the design that gave birth to the program.

1.2 Requirements Analysis

The program is required to:

- allow creation of labels
- allow quick editing of labels
- allow quick reviewing of labels
- allow deletion of labels
- be able to open images from arbitrary directories
- be easy to use for non-specialists
- be able to save labels
- be able to load labels attached to an image

1.2 Requirements Completion

We ensured that the project sufficiently fulfilled all of the requirements to the best of our ability. After the program was complete we looked through the list and ticked them off one by one, and we believe every requirement was met. Most of the tasks are objectively complete, that is there is a function to do what is specified. For the subjective ease of use field, we tested our program on a person with no prior Computer Science experience. Feedback was positive and their intuition was enough that they had no hurdles when approaching the program.

1.3 Design Principles

We aimed to follow key design principles while implementing the project. In particular, we made use of Shneiderman’s 8 golden rules, Norman’s 7 principles and Nielson’s 10 Usability Heuristics.

Some of the ways we did so were as follows:

1. We made our application consistent, by not having buttons be available for certain tasks and not for others

2. All our functions have logical shortcuts attached to them. We made a conscious effort to avoid the staple of Computer Scientists to assume the standard user can remember a command like ctr-shift-alt-G. Instead, our shortcuts are based on industry standards. So control-O is the load dialogue, control-z and control-y undo and redo respectively, and so on. There are even multiple options, so for example some programs use ctr-shift-Z to redo actions, therefore we support that.

3. We clearly permit easy reversal of actions, allowing every action to be undone or redone at leisure. This includes the actual naming of tags as well as the saving of individual points.

4. We have taken to heart the idea of “Simplify the structure of tasks”. Our aim was to have the labeller be entirely intuitive, to the extent that the provided tutorial should not even be necessary. However we do still provide a step-by-step tutorial, as it is important to be accommodating to all users.

5. Our compliance with industry standards should increase the efficiency of users. Having known operations be located in a known area significantly reduces the 'think time', such as having “Load Image” be located in a File menu in the top left of the application.

6. Our minimalist design was inspired by Nielson’s 10th usability heuristic.

2. Interface and Design

If given a piece of paper with a specific picture we add polygons around the objects on the paper by drawing a continuous shape around a particular object and end at the starting point. Therefore the main priority was to provide the user with an interface that will allow him to perform the annotating task efficiently and with ease. It was imperative to amplify the predictability and familiarity of the application by introducing an intuitive design.

2.1 Functionality:

We first looked at the LabelMe project, as well as the sample solution. However we felt neither of these matched our opinions on how the task would best be completed. LabelMe has a lot of unused white space that we felt looks bad, and causes the useful information to be dispersed more than it should otherwise be. However we judged the icons in the toolbar to be effective and eye-catching, so we decided to use that feature. Instead of text they had simple images that conveyed the exact purpose of the tool.
When we looked at the sample interface we liked the way the canvas dominated the screen. This is the area where the user carries out their actual goal, and so should be where the majority of attention is focused. Any time the user has to move to a toolbar or click a button elsewhere that is them stopping their task in order to take care of some function that distracts them from total immersion. Therefore in our project we both made the canvas the focal point and removed the “New Object” button. We came to these conclusions following the discussions within lectures.
Instead we allow the user to complete a polygon by clicking on its starting point, in order to increase familiarity with the program, and instead of the button we added a menu on the right of the picture for the labels.
Since the goal of the interface is to create polygons around objects in pictures, in order for them to be tagged, and then to notify the user when that goal is reached, we didn’t implement any additional button to finish the polygon.
Instead when a polygon is completed a box immediately appears prompting the user with two choices: to add a label to the polygon or to cancel the action. We anticipated that in order gain a faster response time from the user we should add two custom buttons instead of the “Add” and “Cancel” buttons. Performing the same actions across similar tasks and observe the same outcome is crucial for the user. For this reason when entering a label name the user can press the ‘Enter’ key to submit the label.

2.2 Label Menu:

The Label menu shows the tags for each picture along with a button that resembles a bin at the end of each tag This allows the user to click the bin, which will delete the entry. Moving the cursor over the label’s name highlights the corresponding polygon. Editing is also straightforward, clicking on the chosen tag allows the user to redefine the name and change it by pressing the enter key.

2.3 Menu Bar:

The program also incorporates a menu bar that consists of ‘File’, ‘Options’, and ‘Help’ menu items in order to make the application similar and typical to other applications, thus promoting familiarity. From the File menu the user can open an image and from Quit he can quit the application, while been prompted by message to confirm he’s decision.
Within image editing applications, Undo and Redo functionality is expected given the frequent mistakes a user makes when drawing. Therefore the Options menu presents the Undo and Redo functionality, used for undoing or redoing any action the user has made on a given picture, which can be also accessed by the user using the shortcuts CTRL+Z for undo, CTRL+Y for redo.
Last but not least in case the user requires assistance, there is a Help menu that presents the user with a brief guide on how to use the program.

2.4 Split Pane:

The program provides easy access to previously used files. Once the user opens a file of an appropriate format (jpeg, jpg, gif, png or tiff) the image will be loaded on to the drawing canvas. At the same time, the name of the file will be added to a list on the left of the screen. Clicking a name will give you a thumbnail preview of the image, and double-clicking it will load the image. This load will maintain any metadata that exists, so tags will not be lost on a state change.
There are two arrows on the divider between the picture and list areas. One will maximise the picture panel, and the other will maximise the list panel. So if you decide you don’t want to see the preview picture you can minimise it, giving more room for the list. Additionally, each panel has the capability to scroll. So if the panel gets over-filled, you can easily scroll down and you won’t lose information.

3. Key Attributes:

3.1 Shortcuts:

In order to allow the user to complete one task in multiple ways we have ensured that with each and every menu item, there is an associated keyboard shortcut. We have anticipated that this will help experienced users to reduce the time taken to complete the task if they use the shortcuts instead of using the menus.

3.2 Drawing:

Drawing can be done in two ways:

1. Establishing points around an object to construct a polygon.

2. Cursively and freely using the cursor to draw a polygon around an object.

This flexibility allows the user to choose which method of input suits them best and to create polygons limited only to their imagination.

3.3 Auto-Saving:

In order to allow more comfort for the user to focus on their task of creating labels and so avoided unwanted situations each time a user adds, edits or deletes a polygon or label, the application automatically saves the changes, handing over the repetitive saving task to the system. Restricting the user of these options is not a bad thing since labels will still be responded to changes and it is better to remove the potential for a very frustrating situation where the user loses hours of work.

3.4 Multitasking:

Also we allowed the user to engage actively in multiple tasks at time. Perform multiple tasks at one time and maintain state in the case that a user switches from drawing a polygon to editing a label name. While finishing the polygon, the label being edited stays active and the user can resume editing it once he or she has finished drawing the polygon.

3.5 Recoverability and Observability:

When the user clicks on the picture to begin constructing a new polygon, the first point is coloured in cyan to indicate the initial starting point and stays cyan until the polygon is completed making the user aware that he is drawing a polygon, and clearly indicates the point to click to complete it.

Also, the first and final points of a polygon will turn white when the user hovers over them. The change of color clearly identifies that clicking on either of these points will produce a an action different from that of drawing.

In order to make the editing of existing labels easier, hovering over picture labels will highlight the corresponding polygon by filling in the shape in a transparent red allowing the user to quickly identify which polygon they are currently changing the label for.

The Undo and Redo functionality in the application allows a user to undo drawing, and creating or deleting of individual polygons and labels or to recover a past action. They are accessible both through the Options menu and a shortcut key.

The Undo option ensures that a user can recover from and mistakes and Redo protects against slips when using undo. This amplifies improves recoverability and ensures that the user will not be forced to repeat drawing a polygon whenever they make a mistake.
Both actions , undoing and redoing , take roughly the same amount of time.

3.6 High Number of Tags:

We allowed to user to add up to 20 tags per picture which we consider to be a high number. The limit the number of labels was created so that there will be a consistent response time for the application .

4.Tools Used:

We used a number of different tools to develop this program starting from Java Swing and the Web tutorials. We also tried to made good use of the Git repository for hosting the code. Eclipse was primarily used for coding, along with some minor use of basic text editors and Netbeans.

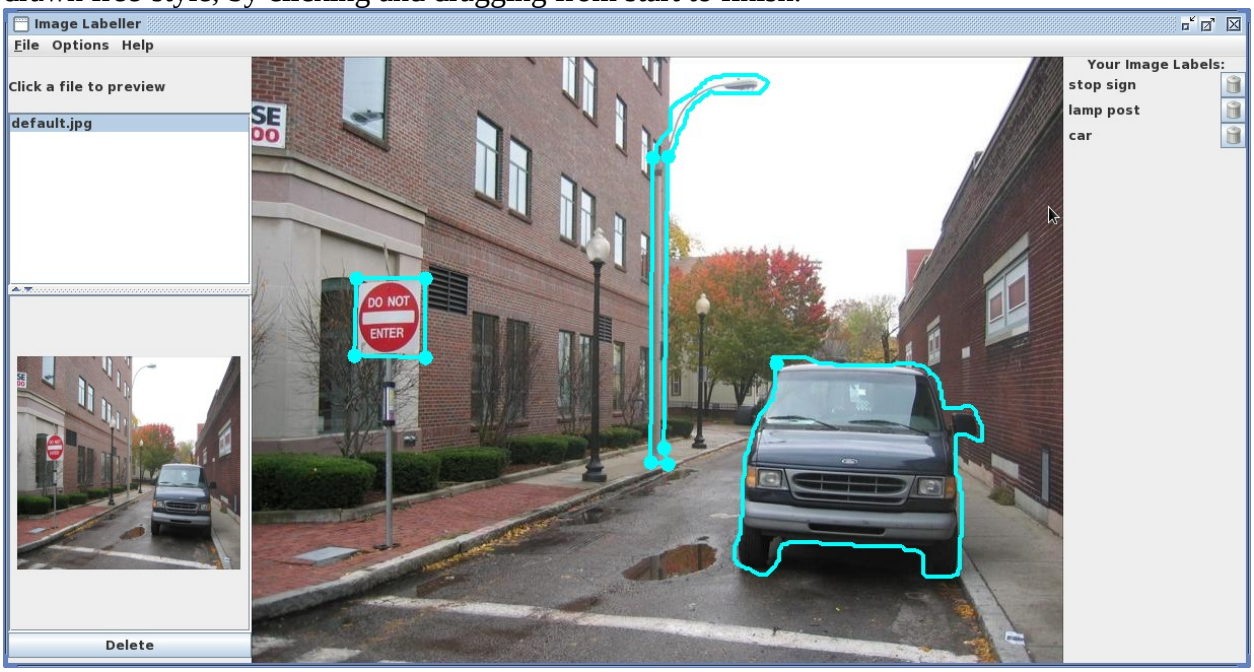
5. Analysis

The strength of our program lies in the natural feel of it. Users can quickly and intuitively learn how to use the tools provided.
We feel that the biggest strength is the guaranteed safety of the program. Using ours, the situation where you crash and lose hours of work is not one that can occur. Every time a tag is changed, this is saved. Again this more closely models the human thought process, as the habit of saving every 5 minutes while writing a report, for example, is not one inherently tied to the task at hand. Instead its a coping mechanism we have developed to deal with imperfect software, and we feel we have tackled this particular issue nicely.
One of the weaknesses of our program is that we make one copy of every image used in tagging. This was not optimal and is only there due to time constraints. Given more time, this would not be necessary, however it is a very small burden on the file system.

6. Screenshots

[Picture 1] This is the first screen a user who launches will see. It has the main canvas, a list of files (with only the default currently there), the preview of the selected file, and a list of tags.

[Picture 2] This is the application after applying a few tags. As you can see, the sign was labeled by 4 independent clicks, which created straight lines between the points. Conversely, the car was drawn free-style, by clicking and dragging from start to finish.



[Picture 3] This screenshot shows what the program looks like after it has been used more thoroughly. We have multiple entries in our files list now, and the canvas has changed so we can draw on the new image.