1. Introduction

In this report, we discuss the design, implementation and evaluation of our application *Image Labeller*, a tool for researchers and developers to quickly produce data sets of fully annotated images.

We used two main design methodologies in creating *Image Labeller.* Our first prototype was modelled using *user-centred* design, after which users' needs and feedback were used to improve our application in an *iterative* process.

Our final product is shown below:

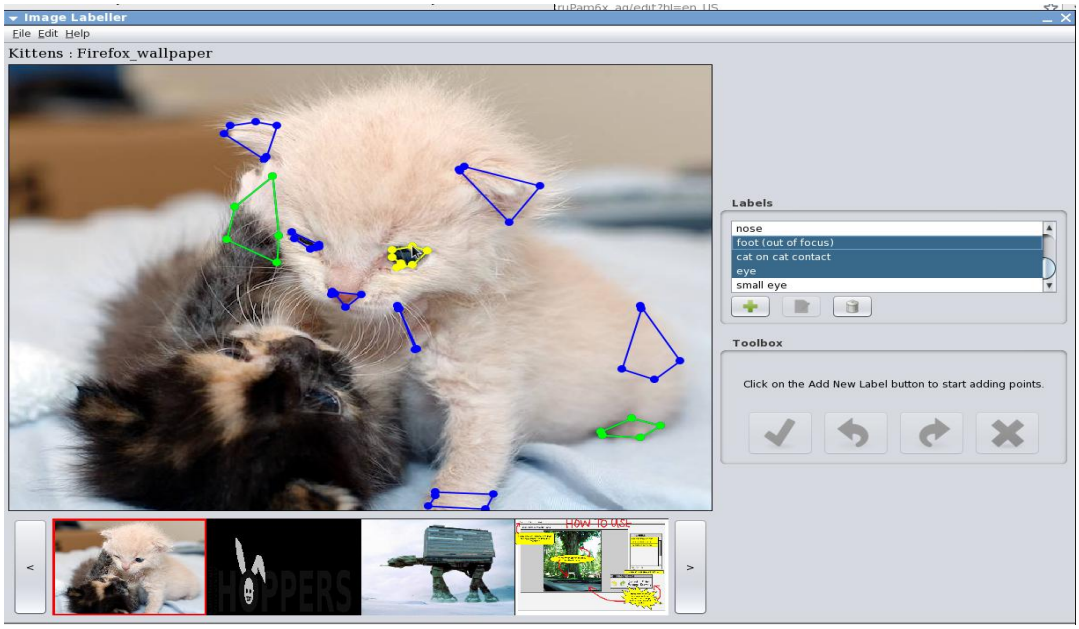

Figure 1: ImageLabeller 1.3

2. The Initial Prototype: User –Centred Design

2.1 Why User-Centred Design?

User-centred design focuses strongly on the context of usage; namely, who would use the application, what they would use it for, and when they would use it. In this approach requirements, design and evaluation are all motivated by usage conditions.

User-centred design has been shown to reduce software costs and shorten development time, and in 2002 the Gartner Group (as cited in Bias & Mayhew, 1994) found that this approach can raise customer satisfaction by 40%. Furthermore, usability is a large factor in deciding if customers will return to use an online service (Chen & Macredie, 2005).

A solid set of requirements and use cases were essential to us for the following reasons:

Limited Development Time:
We had to quickly identify the crucial features of our application to create our initial prototype.

Ease of Use:
If users could do everything they wanted to in an intuitive manner, they would be more likely to return. More useful data would therefore be gathered.

Hence, it was clear that user-centred design would be the most effective method for creating an interface that:

i. provided all necessary functionality;
ii. was user-friendly and encouraged continued use; and
iii. could be developed within a short time frame.

2.2 User-Centred Design Process

The structure of our process was adapted from *ISO 13407: Human centred design processes for interactive systems* (1999).

Before user evaluation, we passed through two phases - *Analysis*, which comprised user analysis, task analysis and comparative analysis, and *Design*, in which we created a mock-up and developed the corresponding application.

In each of these phases, the aspects we considered were an abridged version of those found in UsabilityNet's *First Steps to User Centred Design* (Tscheligi, Fröhlich, & Giller, 2002). As many of these guidelines were not relevant to our situation (e.g. "What is the physical environment in which the system is used?"), we filtered them down to the questions that we could answer in detail.

We used the withdrawn ISO 13407 standard rather than the newer ISO 9241-210:2010. While we felt that the new standard is correct in defining that usability should extend beyond ease of use and into the overall user experience, we decided it was not feasible to consider the "perceptual" and "emotional" aspects of our application within the limited time frame.

2.2.1 The Analysis Phase

User Profile Analysis

Image Labeller is an application to aid processes such as machine learning for image recognition. However, many researchers and developers who require this data do not have time to manually annotate large numbers of images and objects. Instead "crowd-sourcing" is now a common solution used to attain such information.

Hence, we considered our likely users to be a diverse pool of interested volunteers, and identified the following characteristics and requirements of this group:

- **Need to be able to easily and accurately label objects**
  We concluded that the common uses of our application (annotating images for machine learning) would require accurate identification of the objects in the image. Therefore, it would be important to be able to outline objects in detail, correct mistakes made while outlining, and edit labels after they had been made.

- **Need to be able to identify and label many different types of objects**
  Images used in machine learning often contain many different objects within them. For example, in an image of a park there could be one or more trees, as well as humans, dogs, clouds, flowers, and other objects. This means that our application should support viewing and editing a large number of labels on a single image.

Additionally, some users might be colour-blind. If the colours used to outline normal and selected labels were similar, it would be confusing for them. As such, our application should have a friendly palette with easily distinguishable colours.

- **Familiar with other editing systems and shortcuts**
  Many volunteers interested in technical research and developments would be students or workers in similar fields, or at least familiar with programs such as text editors and web browsers.

  Hence, our application should support advanced features such as keyboard shortcuts, as well as standard icons for common functionality (e.g. a trashcan symbol for a delete button).

Task Analysis

We identified two main goals for users interacting with the program: creating and reviewing labels for images, and sending these labels to the researchers or developers.

These high-level goals gave rise to a number of tasks. For example, while reviewing labels, the user might want to go back and edit an already-created label; by adding new points to a label, moving existing label points, or renaming the selected label.

We defined a list of ten main tasks:

- Load image
- Load labels
- Select label
- Create label
- View label
- Edit label
- Delete label
- Save image
- Exit program
- Use Shortcuts

With the exception of “Use Shortcuts”, each main task was comprised of a number of subtasks covering basic to advanced features. The subtasks, their priorities and dependencies between them can be found in **Annex A: Task Analysis**.

Comparative Analysis

We analysed one competing product in our initial design process: *LabelMe*, a web-based image labelling application developed at MIT. Our aim was to determine the features available for labelling, the conventions used in design, and the potential areas for improvement within the product.

We found navigating *LabelMe* to be relatively straightforward. It was easy to create and select polygons from either the image or the sidebar. Windows conventions are used, such as the relative position of the “OK/Cancel” buttons on pop-ups and “Zoom in/Zoom out” in the main navigation. Keeping in line with human memory capabilities, all possible options are displayed on the main screen (without extra menus), and users only have to scan a small area to recognise what to do.

Additionally, we identified a number of useful features. These included the use of a large marker for the first point added when creating a label, highlighting the area of a selected label when the mouse hovered over its name, and highlighting the label name when the lines of the corresponding polygon were hovered over. The first of these meant that the user did not need to spend much effort scanning for their starting point when looking to finish creating a label, and the latter two allowed the user to clarify which shape corresponded to each name. This was especially important since the application allowed for repeated label names.

However, there were some drawbacks to LabelMe.

We found that the placement and wording of the “Erase” button was ambiguous. It looked like it could be used to delete polygons, however, on hovering, the tooltip stated that it would “undo the last control point” - both an ambiguous statement and also possibly at odds with the button’s title. Also, it was not disabled when not in use. The inconsistency of the button’s title and tooltip, along with its failure to constrain users from trying to achieve impossible actions, was in contrast with the design principles of Norman (1998), Schneiderman (1987) and Nielsen (1994).

The dialog for editing polygons was intuitive but cumbersome for the advanced user. When editing a label’s points, accidentally clicking outside one of the points would cause the application to return to the default state. The user would then have to re-enter the edit dialog and click “adjust polygon” again to return to editing. The complication in this design was against Norman’s principle of simplification.

Other features that we considered to be missing from LabelMe were adding points to an existing polygon and selection of multiple labels for easy editing or removal.

2.1.2 The Design Phase

Feature Selection

While we already had a full list of functionalities that we wanted to include, we decided to quickly push a working prototype with only essential features so that we would have time to test and evaluate it. These “essential” features were selected by both their priority and ease of implementation.

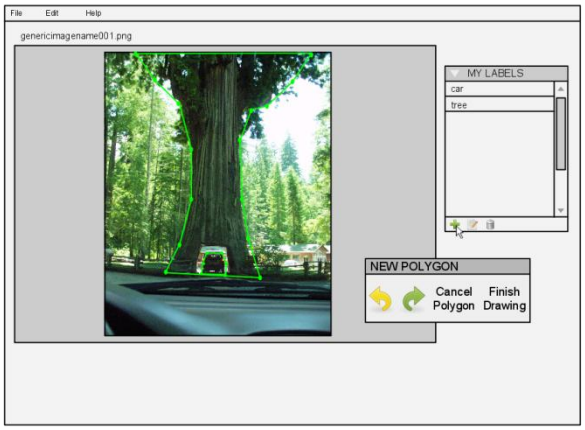

Figure 2: Initial mock-up

The first mock-up we created (Figure 2) contained only the essential features. At this stage, we decided on an “icon-centric” approach with little text. This and other decisions will be evaluated in greater detail in **Section 4**.

A screenshot of our first prototype is shown in Figure 3.

3. Improving our Product – Iterative Design

Within the remaining time for the project, we needed to improve our application by implementing more advanced features and improving on the ones that users had found confusing in our user trials.

Iterative design has been shown to create a large improvement in user satisfaction when using a system. A small study (Nielson, 1993) showed that an average of 38% improvement could be achieved between iterations.

We used two rounds of user feedback to ensure that we had identified and implemented features in line with what real users would want.

3.1 Test Design

To test our application, we designed a set of tasks for users to complete, a researcher evaluation form – on which we noted the timing of their performance on each task, features used, and difficulties encountered – and a user evaluation form, where users were asked to rate the program on its ease of use and if they would use it again. Sample instruction and evaluation forms used for each user are given in **Annex B: Test Instructions**.

Each task corresponded to one or more of the basic tasks that we had identified as essential to the application. By doing this, we could study the application features that users applied for each task, and determine any difficulties that users had with our interface.

3.2 User Sampling

In each iteration, we tested 5 students of varying gender and degrees of familiarity with application development. All were given the same task to complete.

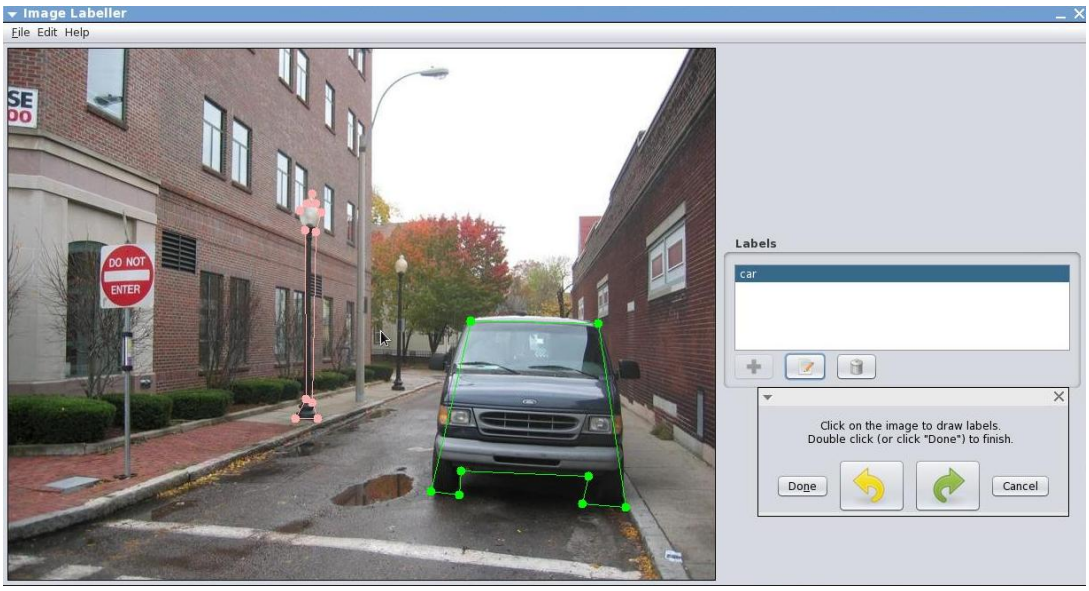

Figure 3: Screenshot of our initial prototype

3.3 Evaluating Results

We initially planned on collecting quantitative results through comparing timings of how long users took to complete each task on the list of instructions.

This proved to be quite difficult as users took different approaches to labelling images. Some took a long time outlining objects as accurately as possible, while others simply drew boxes around the objects. For example, Figure 4 shows the difference between two users labelling the same “House”. This was a weakness in our testing setup, as we should have found users who had an interest in accurately labelling objects (as we had identified our user base to be).

Furthermore, timing inaccuracies were caused by users reading the instructions while carrying out the tasks.

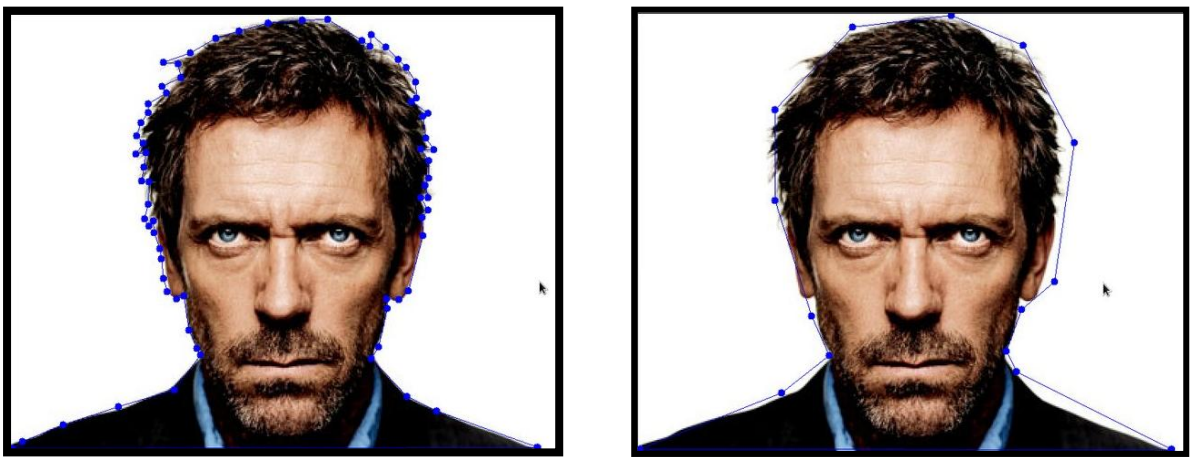

Figure 4: Two different users labelling the same “House”

To improve the features of our application, we used mostly qualitative suggestions and feedback rather than quantitative timings. To assess our overall improvement, users were asked to rate our application on a score of 1 (lowest) to 10 (highest) on how easy it was to use, whether they would use it again, and how aesthetically pleasing it was.

From the first to the second evaluation, our user feedback improved as follows:

- **Ease of use**: 6.5% improvement, from 7.7 to 8.2
- **Likelihood of using the program again**: 11.5% improvement, from 6.9 to 7.8
- **Aesthetics**: No significant change (-1%), from 7.9 to 7.8

Our sample size was small and hence statistics should be taken loosely. However, the non-trivial increase in the ratings of the first two aspects is a reflection of an overall improvement in our program.

4. Evolved Features and Evaluation

4.1 Key Design Features

4.1.1 Icon-Centric with Tool-tips

Naming buttons and menu items is a large Human-Computer Interaction problem, as studies have shown that user-preferred names overlap by only 15-35% (Furnas, Landauer, Gomez, & Dumais, 1987).

For our first user evaluation, many of our buttons were simply common icons depicting functions – for example, a trash can was used to delete a label. However, users struggled to associate some icons to their functions, and did not understand what to do if there was no button to click.

For example, users failed to click our "Add Label" button (depicted by a '+' sign) to create a label. After clicking it, they did not know to click on the image to start adding points.

We addressed this problem with a combination of icons, tooltips and instructions. For example, our editing menu now has a line of explanation, followed by clickable icons with tooltips (as shown in figure 5).

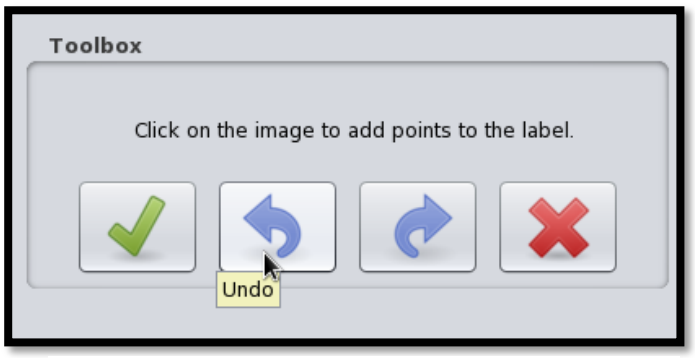

Figure 5: Icon-centric toolbar with helpful text

In our second user evaluation, problems associated with drawing labels were eliminated.

4.1.2 Quick Tips

One of our critiques of *LabelMe* was that the polygon editing dialog was too cumbersome. We wanted familiar users to be able to reach their intended function in minimal time, while allowing new users to learn how to use the interface.

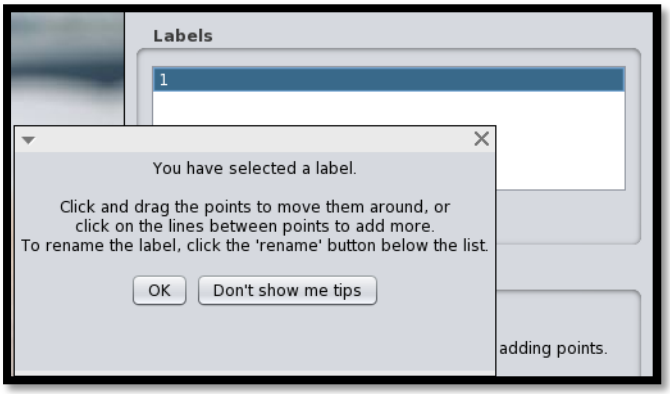

Figure 6: Quick Tip for label editing

Quick Tips brings up a popup dialog to aid the user with tasks that tests had proven to be slightly less intuitive. This is in line with Schneiderman's third rule of offering feedback.

They are enabled by default, but users can disable (and re enable) them from the Help menu.

In our first evaluation, users did not know how to select labels or move their points. This caused an average of a 160% increase in time taken between creating a particular label and editing its points. By our second evaluation, none of the users had this problem.

4.1.3 “Collections” View

From our first evaluation, we realised that we could speed up the time taken to navigate between images by allowing the users to have multiple images open at once. This gave rise to the “Collections” view, which is in line with the criteria that users should be able to edit many images at once.

In our evaluation tasks, users were asked to work on one image, open a different one to work on, then return to the first one. One of the users in our initial evaluation was unable to complete this task, as they did not consider their image they had just edited and saved to be a new image.

The ‘filmstrip’ view of images in a Collection makes it easy to import more than one image at a time, as well as switch between images, simplifying the task at hand, in line with Norman’s second principle. None of the users in our second evaluation had trouble with this.

Figure 7: Filmstrip of images in a Collection

4.1.4 Recovering from Mistakes

Users should be able to recover from their mistakes when creating a label, and hence we implemented their “undo” and “redo” functionality. This was used by 3 out of 5 users in our first evaluation, and hence we found that it was a useful feature worth keeping.

In the future, we would like to implement global undo/redo features, so that users have more complete control over all their actions.

4.1.5 “Add Label” Button

In both evaluations, users attempted to click on the image to start adding points, without first clicking the “add new label” button. It seemed rational to add this functionality to our program so that it would be more user-friendly and intuitive; however, this led to a clash in functionality.

When the points of a label were being edited, users could click outside the lines to deselect it, a common functionality used for selecting and editing in many programs such as Microsoft Powerpoint. This could not co-exist with “Click to begin”, and we felt that it was important to preserve the ‘deselect’ mechanism. Our solution for the “click to add” problem is therefore currently to just guide the user with tooltips; however, this is not ideal and we are still looking for ways to improve it.

4.1.6 “How to Use” Dialog

In our first prototype when our application had less functionality, we found it easier to use a graphical diagram of how to use our application rather than a text file. In our evaluation, one user used this feature and found it helpful.

However, in our second prototype our application was more complex and it was not feasible to have visual instructions on how to use all the functionality. Instead, a new “How to Use” dialog was created with information about specific tasks. Combined with our tool tips, quick tips and instructions, Nielsen’s tenth heuristic on help and documentation is well covered in our application.

4.2 Further Improvements

4.2.1 Non-Unique Label Names

Labels in our application must currently have unique names. When constructing requirements, one of our beliefs was that users would be more likely to recognise their own labels if they had different names. From a user-centred perspective, we implemented the constraint of unique names to increase user efficiency in viewing and editing their labels (as in Nielsen’s eighth heuristic).

However, we realised that since all labels would be sent for processing, it would be more sensible to be able to have non-unique names. If different labels had the same name, classification algorithms could be run without an extra step of parsing or grouping different names.

4.2.2 Viewing and Editing of Labels

In section 2, some interesting features of *LabelMe* that we identified included being able to move entire labels around while keeping their shape and shading the area of a highlighted label. However, we were not able to implement these in our current version of *LabelMe* due to time constraints. Highlighted labels are presently distinguished by a different colour outline. While this has been satisfactory to users thus far, we feel that such shading would be more eye-catching, in line with Norman’s third design principle of visibility.

4.2.3 Exporting Collections

As the main purpose of our application is to provide a dataset for researchers and developers, we would like to add a quick button for compressing, emailing or even uploading a collection. This would be best implemented after we had details of the database that would store the images and corresponding labels.

5. Conclusion

Our initial user and task analysis combined with iterative development has helped us make a reasonably intuitive interface with a good amount of functionality for labelling images. However, there remain a few decisions and functions that we hope to improve if given more time for future iterations.

Bibliography

1. Bias, R. G., & Mayhew, D. J. (1994). *Cost-Justifying Usability.* San Francisco: Morgan Kaufmann Publishers.
2. Chen, S. Y., & Macredie, R. D. (2005). The assessment of usability of electronic shopping: A heuristic evaluation. *International Journal of Information Management 25* , 516–532.
3. Furnas, G. W., Landauer, T. K., Gomez, L. M., & Dumais, S. T. (1987). The Vocabulary Problem in Human-System Communication. *Communications of the ACM , 30* (11), 964-971.
4. Nielsen. (1994). Enhancing the explanatory power of usability heuristics. *Proceedings of the ACM CHI'94 Conference.*
5. Nielson, J. (1993). Iterative User Interface Design. *IEEE Computer , 26* (11), 32-41.
6. Norman, D. A. (1988). *The Design of Everyday Things.* New York: Doubleday.
7. Schneiderman. (1987). *Designing the user interface: strategies for effective human computer interaction.* Addison-Wesley: Reading MA.
8. Tscheligi, M., Fröhlich, P., & Giller, V. (2002, July 15). *UsabilityNet.* Retrieved 10 20, 2011, from First Steps to User Centred Design: http://www.usabilitynet.org/papers/First_Steps_to_User_Centred.pdf

Annex A: Task Analysis

A.1 Listing of Sub-tasks and Priorities.

#	Main Task	Sub-tasks	Priority	Dependencies
1	**Load image (unlabelled)**	Load one image	**HIGH**	
2		Load multiple images	**MED**	
3	**Load image (labelled)**	Load labels onto image	**HIGH**	Prerequisite: #1
4		Load multiple sets of labels onto image	**LOW**	Prerequisite: #1, #3
5	**Select Polygon**	Select one polygon	**HIGH**	
6		Select multiple polygons	**MED**	Prerequisite: #5
7	**Draw Polygon**	Add points	**HIGH**	
8		Undo / Redo points	**MED**	Prerequisite: #7
9		Move existing points	**LOW**	Prerequisite: #7 UI: Move points vs. add points ?
10		Change polygon colour	**LOW**	Prerequisite: #7
11		Finish Drawing	**HIGH**	Prerequisite: #7
12	**View Polygon**	View polygons on image	**HIGH**	
13		View selected polygons	**LOW**	Prerequisite: #12
14	**Edit Polygon**	Add a point	**MED**	Prerequisite: #5
15		Move selected point	**MED**	Prerequisite: #5
16		Edit curve between 2 points	**LOW**	Prerequisite: #5
17		Edit polygon name	**HIGH**	Prerequisite: #5
18		Change polygon colour	**LOW**	Prerequisite: #5
19		Move polygon	**LOW**	Prerequisite: #5
20		Scale polygon	**LOW**	Prerequisite: #5
21		Undo / Redo editing	**MED**	Prerequisite: #5, at least 1 in #[14-20]
22	**Delete**	Delete selected polygon	**HIGH**	Prerequisite: #5

	Polygon			
23		Delete all polygons	**MED**	Prerequisite: #22
24		Delete multiple polygons	**LOW**	Prerequisite: #6, #22
25	**Save Image**	Save image in focus	**HIGH**	
26		Save all opened images	**LOW**	Prerequisite: #2
27	**Exit Image**	Exit program	**HIGH**	
28		Close image in focus without exiting	**MED**	
29		Close all opened images without exiting	**LOW**	Prerequisite: #2
30	**Use Shortcuts**	Use familiar keyboard shortcuts e.g. save, undo	**MED**	Prerequisite: Functions the shortcuts are for

Annex B: Test Instructions

B.1 User Instruction Sheet

User Evaluation

In this evaluation, you will be testing some software for labelling images (i.e. for indicating what region of an image contains what objects). You will be asked to label a few images, and afterwards will be asked a few general questions about your opinion on the software.

The two images you must label are both location in the following folder:

`Documents/HCI/images`

And are named **`image1.png`** and `image2.png`.

Task #1

Working with the first image (`image1.png`), please complete the following steps:

- Label the group of flowers, and label the tree.
- Delete the 'group of flowers' label.
- Label the main part of the house - not the extension on the side!
- Make sure that your house label is called "A House", and that your tree label is called "A Tree".
- Save the labels that you have created.

Task #2

Working with the second image (image2.png), please complete the following steps:

- Label the house, as accurately as you can.

Task #3

Working with the first image again (image1.png), please complete the following steps:

- Load the labels that you saved in the first task.
- Make your "A House" label cover the entire house, including the extension.
- Rename the "A House" label to "A House with Extension".
- Delete all of your labels, and close the program.

Feedback

(1 = not at all, 10 = very much so.)

How easy did you find the software to use?

1	2	3	4	5	6	7	8	9	10

If you had to label images in the future, how likely would you be to use this software again?

1	2	3	4	5	6	7	8	9	10

How aesthetically pleasing did you find the software?

1	2	3	4	5	6	7	8	9	10

B.1 Researcher Evaluation Form

Researcher Sheet

Features used:

- File:
  - Open New Image
  - Import Saved Labels
  - Save Current Labels
  - Close Current Image
  - Exit
- Edit:
  - Delete Selected Label(s)
  - Delete All Labels
- Help:
  - How to Use
  - About

- Label Panel:
  - “Load button” button
  - “Add label’’ button
  - Rename button

    - Delete button

- Toolbox:
    - “Done” button.
    - “Undo” arrow.
    - “Redo” arrow.
    - “Cancel” button.

- Misc:
    - Double-click to finish.
    - Dragging points.
    - Any keyboard shortcuts.

Questions to ask:

- Were you stuck at any point during the tasks? If so, what made you stuck?
- Was there anything that you wanted to do that you were unable to?
- Any general comments?

Notes:

Timing:

Task Number	Sub-task	Time (s)
1	Label the group of flowers and label the tree.	
	Delete the group of flowers label.	
	Label the main part of the house	
	(Possibly rename labels)	
	Save the labels	
2	Label the House	
3	Load the labels	
	Extend the “A House” label	
	Rename the “A House” label	
	Delete all of the labels	
	Close the program	