FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Illustrative Thumbnails

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Rebeka Koszticsak
Matrikelnummer 1325492

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Dr. techn. Manuela Waldner, Msc.

Wien, 1. Jänner 2001

Rebeka Koszticsak	Manuela Waldner

Illustrative Thumbnails

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Rebeka Koszticsak
Registration Number 1325492

to the Faculty of Informatics

at the TU Wien

Advisor: Dr. techn. Manuela Waldner, Msc.

Vienna, 1st January, 2001

Rebeka Koszticsak	Manuela Waldner

Erklärung zur Verfassung der Arbeit

Rebeka Koszticsak
Address

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Jänner 2001

Rebeka Koszticsak

Danksagung

Ihr Text hier.

Acknowledgements

Enter your text here.

Kurzfassung

Ihr Text hier.

Abstract

Thumbnails are used to display the screens when switching between them on the computer and on mobile devices. These images make it easier to recognize the opened applications, and help to find the needed window quicker. Thumbnails display however only a screenshot of the windows, so they get potentially confusing if there are more opened windows or if the same application is opened multiple times. Depending on the resolution of the display, the screenshot size decreases as the number of opened windows increases. Furthermore, within the same application (like MS Office World) the screenshots are similar in appearance (eg.: white paper and tool bar), but the important text is not readable. There are several approaches that filter the important areas of the images to make editting less obvious or enhance the main region. In this bachelor thesis an application is implemented that uses these methods on the screencaptured images. The less important areas of the screenshots are cut off, and the thumbnails show only important information, which makes them more illustrative and easier to fulfill their purpose.

Contents

CHAPTER 1
Introduction

Enter your text here.

CHAPTER 2

Related Work

There are several image processing algorithms, which can be helpful by creating illustrative thumbnails. The main difference between them is, if they consider the fact that the input images always are screenshots. Therefore, this section is divided into two parts. The first one discusses algorithms with UI processing segments. Algorithms, invented for retrieve visual data from common images, are examined in the second section. According to the information visualization method, like combining the most important parts in form of collages or simple resizing, two classes are divided. In the following some of these methods are compared.

2.1 Processing UI elements

Considering that the input is a screenshot, there is a high chance that common UI elements are shown on the screen. Exceptions are only the cases where graphics, images, videos or application containing these, e.g. video games, gallery program, video player are captured. The applications like that tend to hide all UI elements, "fullscreen" mode, or redefine them, e.g. game menu.

Labeling the image parts as content and non-content, the metadata about the UI elements can be helpful. Forras uses already existing accessibility APIs to segment UI and non-UI data. Matching the metadata with the screen content provides a fast and robust result about the location of any kind of UI content. There are howerer several disadvantages, that need to be take into account when using such APIs. The range and the granularity of the support is often not wide enough. The use of an accessibilty API does not make sure, that every UI element will be recognized, because some metadata is not reachable or it will be ignored by the application.

Because of that Forras Prefab works with its own prototypes. In the database models and prototypes of common UI elements are saved. The (components of) UI elements has to be matched with the prototypes in the data base, and after that the predefined metadata can be accessed. Since the Prefab system is able to split complex widgets into their base elements, the database does not need to be unnecessary big, but it is still able to cover the most common UI elements. In the case of special or rare UI widgets, like in a video game application or elements of a not not widely used software, the system fails. If a widget is not saved in the database, there is no way, that it will be recognized.

Forras Sikuli offers a solution not for the incompleteness of the database, but for the granularity issues too. It uses its own templates just like the Prefab system, but only in case of small icons and widgets. Since in case of larger objects template matching would be too expensive, after the processing of a training pattern, the Sikuli system is able to create new object models too. Although this feature is used in the application for other purpose, i.e. reduce the matching cost, it can solve the problem of database and of the granularity. Sikuli makes it possible to expand the database and to make the database entry afterwards more detailed.

But in case of accessibility APIs not only the availability of metadata causes possible problems. It has also no information about the actual visibilty of the UI elements. One window or rather one widget can hide an other one, some content can be out of the range of the borders of the screen etc. Forras is based on the Prefab system, but it builds a hierarchical tree from the widgets. The content is found at the leafs, and the parents are the widget, where the child is built in. With the help of this tree the order of the UI elements gets clear, and so the misleading information can be eliminated.

After the labeling of the UI elements correctly, they can be processed as needed. They can cut off as whole or processed according to the information content. Forras invented an algorithm to eliminate objects of a picture and fill their place with the texture of the object behind them using the z-buffer information. The tree mentioned above works as a z-buffer in this case. With this cut off algorithm unnecessary widgets can be easily eliminated and more important elements of the UI can so get better visible.

According to the definition of "illustrative" of this thesis the UI elements of a screen has to get cut so the segmentation of the UI elements is not needed. Although the Prefab and Sikuli systems can be helpful for labeling the images as UI and content. But theses approaches has the drawback of the usage of the database and the slow template matching. Since there is no need to know, which widgets are on the screen, and processing the actual content can already cause performance issues, the use these methods would overcomplicate the application without providing noteworthy advantages.

CHAPTER 3

Methodology

List of Figures

List of Tables

List of Algorithms

Index

Glossary

editor A text editor is a type of program used for editing plain text files.. 5

Acronyms

CTAN	Comprehensive TeX Archive Network. 11
FAQ	Frequently Asked Questions. 11
PDF	Portable Document Format. 6, 10, 11, 15
SVN	Subversion. 10
WYSIWYG	What You See Is What You Get. 9

Bibliography

[Tur36] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58:345–363, 1936.