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

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

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

Kurzfassung **xi**

Abstract **xiii**

Contents **xv**

1 Introduction **1**

2 Related Work **3**
2.1 Processing as UI 3
2.2 Processing as common picture 4

3 Methodology **9**

List of Figures **11**

List of Tables **13**

List of Algorithms **15**

Index **17**

Glossary **19**

Acronyms **21**

Bibliography **23**

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 as UI

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.

2.2 Processing as common picture

There are several information retrieving methods for processing images with any content. Furthermore in case of making any kind of image more illustrative, there are important tasks like interesting point recognition, Region of Interest (ROI) selection, image or feature composition etc. which they can handle easily. According to the manner of the input data there are two different groups of these algorithms, which need to be discussed in the following sections. The first category works with more than one picture in the same time. Its strength is to choose singe features which represents the most the whole input data. In exchange it is likely that no input image will be recognizable on the result. On the other hand there are the methods in the second group, which take only one picture for input and process it as one unit. So the result is similar to the input data, but therefore the chance is high that not only the unimportant but the areas with high information ratio are damaged At the very and of this section some approaches are presented, which can help the work of both of the above mentioned groups.

2.2.1 Collage creating methods

A collage is an assembled image, containing parts of a bunch of input images and being representative for the whole input data. There are two cases by creating thumbnails more illustrative where such methods can be helpful. On the one hand the actual information of a screenshot image concentrates only in few regions of the picture. Many parts, for example UI elements, space between the content etc, can be ignored. The other way around the content can be retrieved in form of ROIs, and afterwards they can be combined arbitrary. On the other hand using collage creator algorithms thumbnails for desktop switching can be easily generated, where the input are screenshots of the open application of the desktop instead of some ROIs of one screen.

For a representative collage the most important task is to choose the best images, which information content covers the whole input data. Forras takes the parameters representativeness, importance costs, transition cost and object sensitivity into account. Representativeness means being interesting in this case. A picture tends to have high representativeness value, if there are many special textures on it, and if it is not similar to the rest of the data (so that no image is chosen twice). Importance cost evaluates and collect the ROIs of the input. While transition cost stands for the smooth transition between every two images. At last, the parameter object sensitivity holds the results of object recognition, and it helps in the arrangement, that the object has a reasonable placement.

Forras concentrates however only the first two parameters above. It clusters the images, according their source and the time, and measures the quality. Thank for the clustering by the choice of the final images it is clear, which images are the same or have similar content. This feature, accordingly modified, can be useful by sorting ROIs of a screenshot, i.e.: text content, image content etc. or of the running applications of the desktop, i.e.: textprocessing, gallery application etc. The parameter quality summarizes the value of the results of the following calculations: blurriness, compression, contrast and color balance. Since in this case only screenshots, thus computer generated pictures, can be the input, these measurements invented for camera data would provide less meaningful results than the algorithm above.

Having the best ROIs for the collage the last task is to merge them into one output picture. For this purpose Forras uses a method called ROI packing. First the central point of every ROI needs to get chosen and every pixel on the canvas needs to get assigned to one of the using the k-Means algorithm. After that the ROIs can be placed on the area calculated for them. To fill the place between the ROIs they have to grow, keeping the aspect ratio, until they overlap. Then every ROI is shifted to the middle of its area. This method can be repeated until no ROI grows anymore. To fill the white areas, neighboring ROIs are allowed to cover them, and so fill the whole image.

Collage methods are very useful to represent a large image dataset only in a small place. They work with numerous input data, and taking the most important parts of them create a new image, that is not similar to any before. That is why they are more useful for making a thumbnail for a desktop but not for one application. Even though taking only the most important ROIs of one screen it would be possible to create an image more illustrative that any other, since the important content is the largest and the best readable. Cutting one screen apart and arranging some parts of them willingly would confuse the user, and even more time would be needed to recognize the screen.

2.2.2 Resampling methods

Treating the input image as one unit, has a big advantage against the approaches mentioned before. Even after modification the result will be highly similar to the input. Resampling means, that some parts of the image will be eliminated, but all remaining areas will have the same proportion to each other, not like by the collage algorithms. The image remains recognizable because of the same look, even though the most important areas are less readable.

To select areas, which needs to kept same, Forras uses several attention models, which defines Attention Objects (AO). AOs usually are objects from the real world, which catch the human eye, because of their familiarity, shape, color etc. With three values AOs can be easily parametrized, these are ROI, Attention Value (AV) and Minimal Perceptible Size (MPS). The attention models fit the AOs into the context. Forras works with three different attention model at the same time, with saliency, face and text attention. Having the importance value of every pixel the most important area can be detected.

The algorithm above after carefully calculation chooses only one area, which contains as many AOs as possible. So some possibly important AOs have to be ignored and cut off. With Feature awared Texturing described in Forras this do not have to be the case. The algorithm expects an input image and a feature mask. A grid is generated, which lies on the input image. This grid can be modified in an optional shape, but the gridpoints on the feature mask are not allowed to change their proportion to each other. In that way the picture elements between the AOs fill the new shape, but the AOs get barely distorted

A detailed importance is essential by creating thumbnails, since a screen usually contain more information and ROIs than a common picture. But the algorithms described above has aspects, like face recognition and grid calculation, which overcomplicate the calculations, causing performance issues without reaching better quality. A face on a computer screen is not so common as on usual photos, and it is not so important too than text for example. With a grid the input image can get reshaped in any other form, and in addition no important part needs to get damaged. But it is meant to form the input in completely other shape, not for resize it according to aspect ratio. Some aspects however, like definition of AOs, could be changed with the actual used approaches, this possibility will be discussed in the future work section.

2.2.3 Feature combining methods

All algorithms mentioned before works with some kind of feature combining algorithms. After choosing the interesting areas, which need to stay recognizable on the resulting image too, they will be merged according to their methods into one output picture. In this process it is usual that some artifacts show up near the joining are.

A possible solution for this problem is the Piosson equation for image processing, introduced by forras. A modified Poission equation using a guidance field needs to be solved for every color in the color space, where the guidance field is calculated right from the gradients of the input image. This algorithm is actually invented for seamless cloning, but it is helpful, when some not neighboring parts of the same image needs to be placed near each other. To make the Poisson calculation more efficient an algorithm suggested by forras can help. It proposes to solve the equation on an image pyramid built from the source and its destination image, so from the two regions, which needs to be combined in this case. To reach the smoothest result forras works with gradient-domain fusion developed from the Poisson equation. It collects the color gradients of the source images into a composite vector field. Afterwards a color composite is calculated, where the gradient fits in the vector field as much as possible.

Combining and accumulating the color data promises very smooth results, joining areas, which appears naturally. These algorithms are invented however for combining pictures from the real world, and not for computer generated images. In case of thumbnails, it is less important to make the transitions less edgy, since the source does not seem to be natural neater. So the use of such advanced methods like Poisson equation is pointless, since the offer, providing natural edges and smooth transition, matches not even the input screenshot.

CHAPTER 3

Methodology

The algorithm, which creates thumbnails more illustrative, has three main steps. At the beginning the UI elements are cut off. In the case of thumbnails there are other ways to get information about what application are actually opened, for example the labeling of the thumbnails. In addition it is usual that the same software is running many times for other purposes, for example the usage of text editor for writing and for reading some document. The content not the surface of these application matters, although reading the title of the thumbnails is smaller than the software recognition through visual data, this aspect needs to take count for the better visualization of the actual content area. The second step is the calculation of the importance map weighted on the place of text data. Unlike to common pictures from the real life the occurrence of text on computer screens is higher and more important. For this reason the calculation of the final importance map have two steps. First the importance value of every picture is generated according to the color changes of the source image. Second the color value is increased at the places where text is found. In that way not only the silhouette but the whole body of the letters seams to be salient. Lastly seam carving and simple resampling is performed until the output size is reached. In this section these three main sections is discussed, and an overview about the algorithm is shown.

3.1 Eliminating UI elements

For the elimination of UI elements, they need to get identified first. In the search for UI widget, it is supposed, that they are placed near the border of the screen but not in the central area. The middle of a screen works as reference data for the investigation, and it is not checked if there are some unimportant UI elements placed. The UI elimination algorithm works right on the source window, no further premodification is needed for this process.

Cropping the border first the horizontal then the vertical margins are checked, where the order is relevant. The algorithm is based on the observation, that there are the upper and the lower bars, which cover the whole width of the display. The sidebars, if they exist, run however only between the the upper and the lower bars, and cover the remaining area. Furthermore it is common that the sidebar have slightly different style that the other UI widgets mentioned above. The UI matching method investigates slices of the screen running along the margins and checking if they belong to the UI or to the middle area of the window. To get better results, it is important to investigate coherent data, and to not mix parts of different UI bars in one slice. That is why the order of margin cropping is chosen in the mentioned way.

To find the best cropping line, where the UI meets the content area a double checking method is performed. The first step is searching for a line having the same color and no interruption along the horizontal sides of the screen. This method is based on the consideration that toolbars often have a one color background. Therefore the task is to find a line, where no more widget is placed, but it still belongs to the UI area, so it is filled with a background color.

If there are a defined UI are the method above works well, but there are several cases, where it is not able to provide any result. Game applications usually uses their own graphical UI, but even by more traditional applications it can happen that the UI area is so overloaded, that no background line can be found. For that reason is it important to perform an other checking loop too. This step is based on the correlation value of the histograms of the border and the center. The margin is split into thin slices. Initially the first slice, near the border is checked. The histogram of this slice is compared to the histogram of the center. If the correlation is high, it means that the two areas have no big difference, so nothing should get cut off, the algorithm returns. But if low correlation is shown, they are not part of the same unit, the slice comes supposedly from UI area. In this case the two histograms, the one of the first slice and the other of the center region, is kept as reference value. In the next step the other slices are compared with the two reference histograms starting at the border and heading to the middle area. At the beginning the correlation with the border histogram is high and with the center is low, which shows that the slice is still in the UI area, it matches the theme of the border widgets. But by the slices being near to the center this tendency turns. At the slice whose correlation with the center histogram is higher than the one with the border the algorithm stops, cuts every slices which were checked before, and it terminates.

This method is executed four times. In the first two cases the upper and the lower borders are investigated. The third and the fourth loop is adjusted to the sidebars. In the first step not a horizontal but a vertical line is searched, looking for a straight route, where the background color of the sidebar taking the whole space between the, already cropped, upper and lower borders of the image. Finally in the second checking loop the slices are defined also vertically.

3.2 Saliency calculation

3.3 Seam Carving

List of Figures

List of Tables

List of Algorithms

Index

distribution, 5

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.