UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

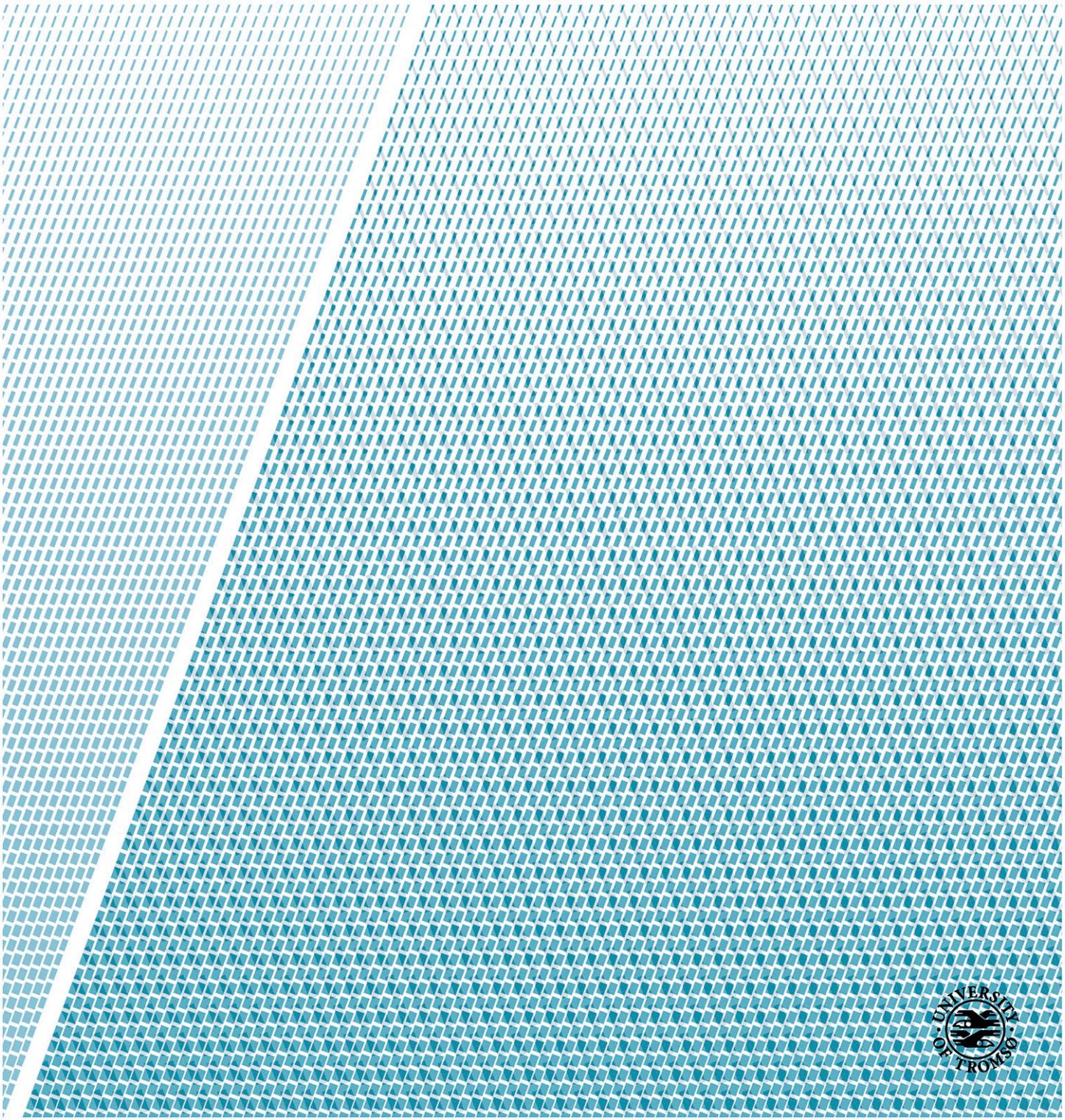
Faculty of Science and Technology
Department of Computer Science

Amdex: Automated meta-data extraction from KEGG pathways

—

Kenneth Knudsen

INF-2990 Bachelor's Thesis in Computer Science - June 2014



Abstract

A pathway map is a graphical representation of a biological process that allows researchers to gain new knowledge abaout an organism. Complex disease such as cancer requires the study of multiple biological processes, made possible by visualization pathway maps.

Visualization tools that exists today are either drawing on top of static image of pathway maps or dynamically drawing pathway maps images from database for curated pathways. The first approach limits the flexibility and visualization of the visualization tool. The second approach is creating pathway maps which are messy and is missing information.

This thesis presents Amdex, a tool for extracting meta-data from manually drawn pathway maps. Amdex is loading the pathways which are presented as png images. This allows the possibility to create dynamically pathway maps which looks like they are static. We utilize the OpenCV library for image processing, in order to detect elements such as genes within the pathway maps.

Amdex extracts individual genes from the pathway maps and we evaluated the extraction by testing 20 randomly selected pathway maps. Our results show an average accuracy on 80%, and 100% for several pathways. We have identified several challenges for extraction of meta-data from KEGG in addition to genes. An important factor for gene recognition mis-prediction are pixel-level errors made by the humans that draw the pathways.

Our initial results demonstrate that the process of automatically extracting meta-data from pathway maps was harder than expected. Additional meta-data has varous shapes, sizes and colors, and is therefore hard to extract automatically.

The Amdex system provides promising initial results, but in order to extract all connections and additional meta-data future research is required to tune the OpenCV methods, and find approaches for tolerating pixel-level errors.

Acknowledgements

I would like to thank my adviser Associate Professor Lars Ailo Bongo and my co-adviser Associate Professor John Markus Bjørndalen for their continuous feedback, support and motivation during the course of this project.

I would like to thank Robert Jenssen with adressing the problems of the assignment, which pointed us in the right direction. And big thanks to Ove Henrik Kåven for the help with the OpenCV library.

Finally, I would like to thank Bjørn Fjukstad for his continuously support and contribution through the project period.

Contents

List of Figures

List of Tables

Abbreviations

API application programming interface.

KEGG Kyoto Encyclopedia of Genes and Genomes.

KGML KEGG Markup Language.

NOWAC Norwegian Women and Cancer.

PNG Portable Network Graphics.

REST Representational state transfer.

Chapter 1

Introduction

Tools for visualizations of biological pathways are essential for understanding the biological processes in for example the development of cancer (carcigenesis). These pathways are networks of biomolecules and biochemical reactions that describe a series of actions leading to specific biological effects. An example pathway is shown in Figure 1.1.

The networks are both large and complex. Humans have approximately 20500 genes (nodes in the network), each with multiple reactions (edges), and many of which are involved in multiple pathways. It is therefore not practical to display the entire network at once. Instead, the network must be split into sub-network for important biological processes, cleaned manually by removing unimportant edges, and annotated by adding meta-data in the form of symbols. These expert curated graphs form the basis of many biological data visualization tools.

The most popular database for curated pathways is Kyoto Encyclopedia of Genes and Genomes (KEGG)[1]. Alternative databases include Ingenuity[2] and BioCarta[3]. In addition, there are many annotated pathways in the scientific literature. These provide the pathways as images and associated meta-data. Unfortunately, the meta-data provided by the databases may not contain information about content removed from the sub-network by the curator, nor the additional meta-data. It is therefore not possible for visualization tools to dynamically draw pathways that are identical (or similar) to the curated pathways (as in MetaCyc[4]). Instead, current visualization tools either draw a pathway using the meta-data, or draw on top of a static image of the pathway (as in Entourage[8] or Kvik[11]). The problem with the first

approach is that the resulting pathway is messy and is missing information, and it is therefore hard for domain expert to interpret. The problem with the second approach is that the static representation limits the flexibility and visualization approaches that can be used by a visualization tool.

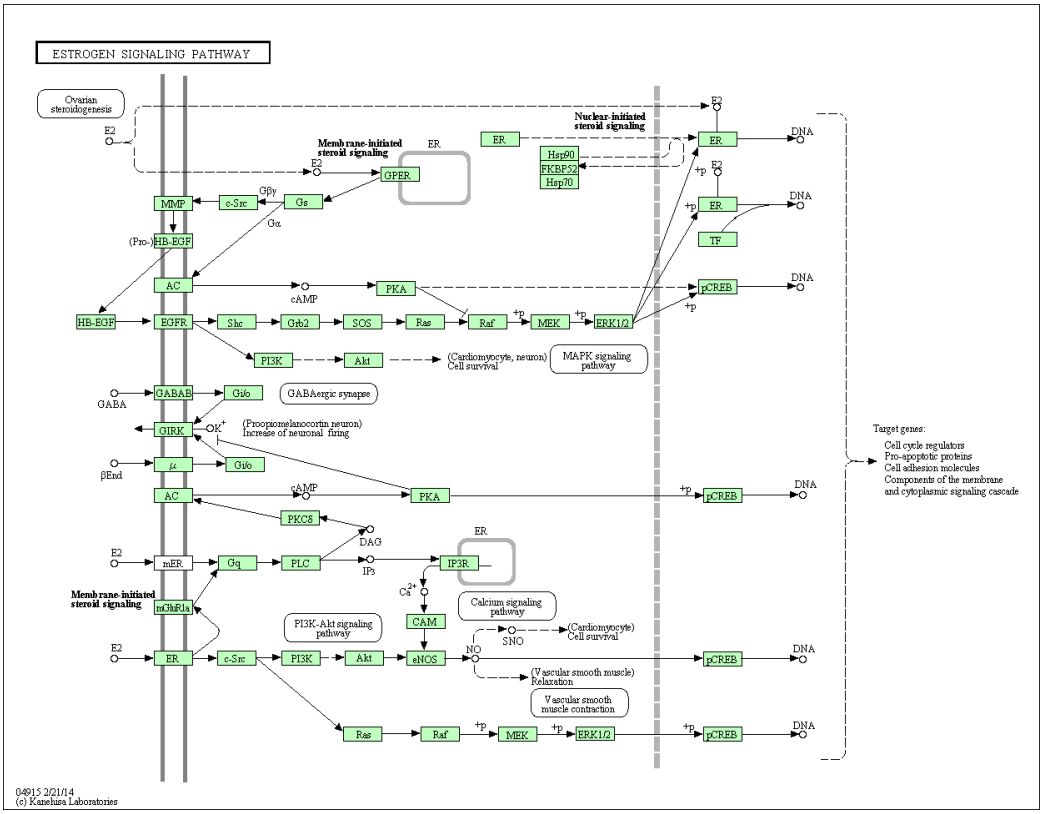


Figure 1.1: KEGG pathway for estrogen signaling.[1]

This thesis describes an approach for extracting meta-data, including the edges displayed from manually drawn pathway maps. The extracted meta-data will allow us to draw dynamic pathways which look similar to curated pathway maps enabling more flexible visualization.

The approach is realized as Amdex, which is a tool that extracts meta-data from KEGG pathway maps. Since licensing restrictions prevent publication of the extracted meta-data, Amdex will maintain a local database, populated by the user. Since Amdex is run on an ordinary laptop or workstation, it is therefore important that the image processing is fast and efficient.

Amdex retrieves pathway images from the freely available KEGG REST

[1] `http://rest.kegg.jp/get/hsa04915/image`

API[2]. The meta-data can be downloaded through this API, but the meta-data is not consistent with the image. Amdex utilizes OpenCV[5], to detect and locate genes. OpenCV is an open source computer vision and artificial intelligence library written in C++. There are other computer vision libraries such as VXL[3] or AForge.NET[4], but since the OpenCV is the most widely known with well-documented we chose to use it for this exact reason.

We have implemented gene detection which is locating the genes found in a static KEGG pathway map. Through an evaluation of Amdex we have achieved an average accuracy on gene detection on 80.3%. The edge detection is depending on the result of gene detection. Non detected genes are treated as edges, this is because the edge detection detects every pixel on the pathway map.

The experimental evaluation of image processing with the OpenCV library, has shown that extracting meta-data from curated pathways is no easy task. It shows there are still unsolved challenges in extracting meta-data that can enable automatic generation of complete pathway maps.

The contributions of this work are:

- Identifying the challenges of automatically extracting meta-data from manually curated pathway maps.
- The design and implementation of Amdex that successfully extracts genes from such pathway maps.

The thesis is structured as follows. A short presentation of related visualization tools is given in Chapter 2. Chapter 3 describes our approach for how Amdex solves the different problems in the pathway. The implementation of Amdex is presented in Chapter 4. Chapter 5 covers the evaluation of Amdex. Concluding remarks and future works are given in Chapter 6.

[2] `http://rest.kegg.jp`
[3] `http://vxl.sourceforge.net`
[4] `http://www.aforgenet.com`

Chapter 2

Related Work

There have been an increasingly number of new visualization tools for biological data. Many are based on pathway visualizations. Biologists often use these tools for exploring and managing large dataset of curated pathways. Many uses the approach of drawning on top of static images. This provides the illusion of dynamically drawn pathway, but the user can not drag elements or hide certain elements which may be done in a real dynamicly drawn pathway.

2.1 Bubble Sets

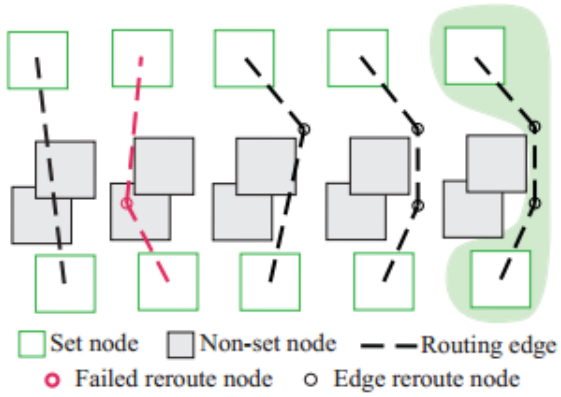


Figure 2.1: Bubble Sets technique to visualize set relation [10].

Bubble Sets is an highlighting approach to visualize set relations over existing visualizations[10]. It respects the spatial rights of the initial visualization and do not disturb it when displaying the secondary set relations.

2.2 KEGGViewer

KEGGViewer is a BioJS[12] component to visualize KEGG pathways. It uses the KGML representations of pathways from the KEGG REST API to build pathways, and visualizes them in a web browser using Javascript library Cytoscape.js. Since KEGGViewer only uses the KGML representation to generate the visualizations, they are lackin both in contextual information as well as nodes and edges.

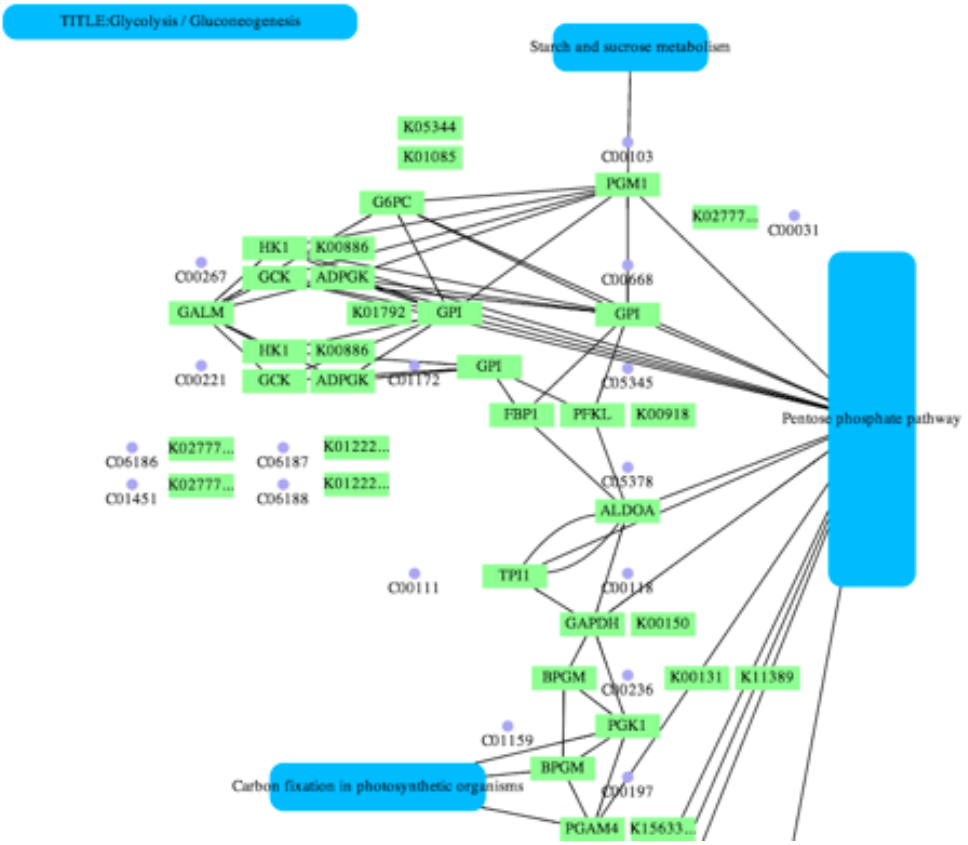


Figure 2.2: Overview of the KEGGViewer component[7]

2.3 Kvik

Kvik is an interactive system for exploring the dynamics of carcinogenesis through integrated studies of biological pathways and genomic data[11]. Kvik provides lightweight visualizations of biological pathways from the KEGG database integrated with genomic data from the NOWAC biobank. Similar to Entourage, Kvik visualizes biological pathways by overlaying nodes ontop of the static pathway image from the KEGG database. Figure 2.3 shows the user interface of Kvik.

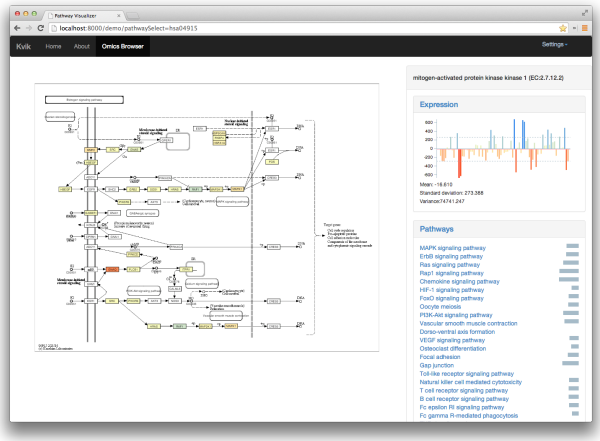


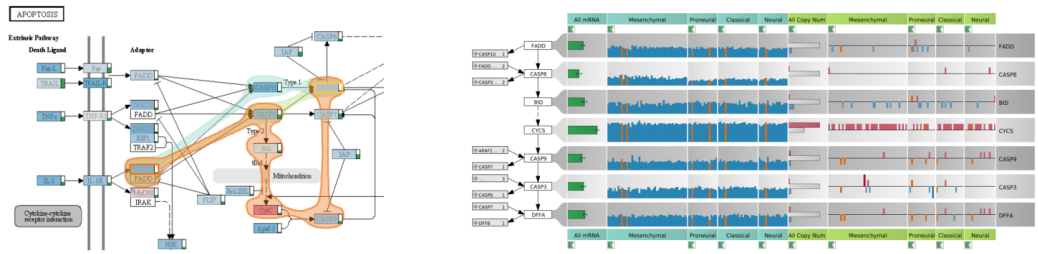
Figure 2.3: Overview of the user interface of Kvik. Figure from [11]

In the future, the developers of Kvik want to integrate Amdex into their system. This will enable more flexible visualizations allowing users to modify pathway maps and move nodes interactively.

2.4 enRoute

enRoute is an exploration tool that allow researchers to explore experimental data from paths that are dynamically extracted from biological pathways[9]. It visualizes pathways by extracting graph nodes from the KGML description of a pathway, and thereafter overlaying the nodes on top of the static pathway image from the KEGG database.

To highlight the selected path in the pathway, it uses a slightly modified ver-



(a) (b)

Figure 2.4: A path highlighted in orange in the pathway map in (a). This is extracted and shown next to associated experimental data in (b) [9].

sion of Bubble Sets. Beside the highlighted nodes, it visualizes the associated experimental data side-by-side for comparison.

2.5 Entourage

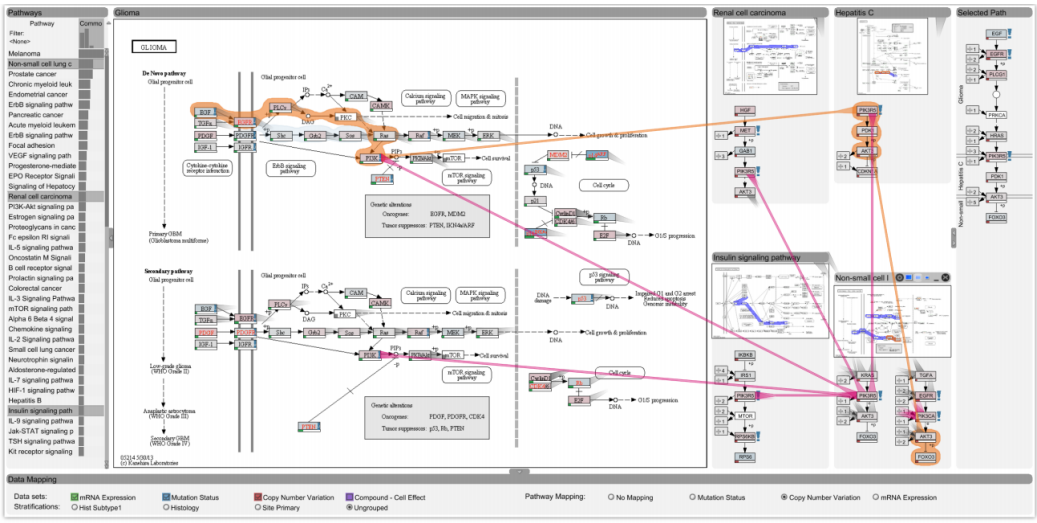


Figure 2.5: Entourage showing the Glioma pathway in detail and contextual information of multiple related pathways [8].

Entourage is a visualization technique that provides contextual information when visualizing multiple related pathways[8]. It uses a single focus pathway for main interaction and exploration, and visualizes only what is important to researchers from other related pathways. Entourage visualizes subsets of related pathways to give context information about the location of user selected genes within related pathways.

Entourage uses the enRoute technique to visualize experimental data and Bubble Sets to highlight the selected nodes within a pathway.

Chapter 3

Method

To extract meta-data from KEGG Amdex does the following:

1. Download a PNG image from KEGG.
   This has to be done manually from the KEGG REST API.[5]

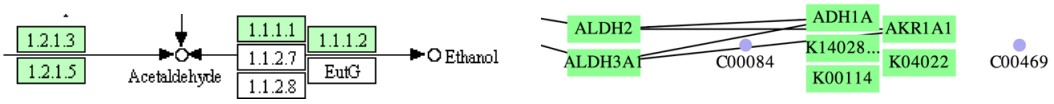
2. When the PNG image is locally stored it is loaded into the Amdex.
   Amdex uses the OpenCV library to extract meta-data which is stored for later use.

3. Amdex represents the meta-data as vectors containing information about nodes and interactions.

The KEGG KGML data is not consistent with the meta-data in the image. The meta-data could be potentially useful, but is not used at the moment. In Figure 3.1 we can see that the only consistent property is the position of the nodes. But in some cases there may even be nodes that is not present in the meta-data, and therefore we may not use this for the cause of the bigger picture. Although the genes can be mapped their interaction (edges in the graph) is very different in the KEGG KGML and the human drawn image. Figure 3.1 shows the difference between a subset of pathway from KEGG image, and how it would be if drawn based on the KEGG KGML data. The molecule *C00469* has no incoming edges but the pathway visualize that the cluster of nodes has an edge to the molecule Ethanol. This implies that if we would use the existing meta-data from KEGG, we would lose information.

[5] `http://rest.kegg.jp`

There are also far more edges in the KEGG KGML data than in the KEGG PNG.



(a) KEGG PNG (b) KEGG KGML

Figure 3.1: A small subset of a pathway visualized by the KEGG pathway image[7]and KEGG KGML drawn by KEGGViewer.

3.1 Approach

To extract the meta-data from the KEGG pathway maps, we need to identify all the elements and the connections between these. The elements may have interaction or relation between eachother. This information is important for the biologist who has interest in or use of these pathways. The genes have to be seperated from the other elements in the pathway, and the gene may have interaction to another gene, and this need to be captured and stored in our data structure. These genes are notated as boxes in the pathways and the expression, relation, interaction, etc. to another gene or molecule is seen as a line with or without arrows.

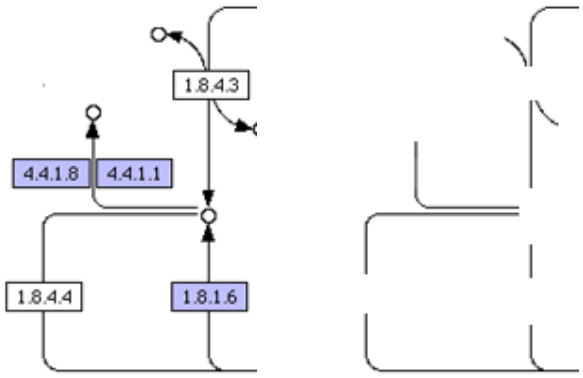
Pathway maps vary in size with regards to such as the number of genes, interactions. To extract the meta-data, we have to examine the pathway map, at the pixel level. For example, we need to distinguish genes from interactions. Both are represented as a two pixel wide line, so we need to implement a program that can differentiate between what a gene is and what an interaction is. OpenCV help us implement the required functionality.

Figure 1.1 shows a pathway where the genes are represented by boxes. Amdex detect the boxes and stores the box as a vector for later use. The gene names inside the gene boxes is also vital information that should be stored. To extract this information Amdax require another algorithm. The genes may have different color codings, so Amdax requires an algorithm to identify for the color of the gene.

Since we have a specified representation of the box, we know that the meta-

[7] `http://rest.kegg.jp/get/hsa00010/image`

data that is extracted is a gene box. Since we do not have any predetermined representation of interactions, every other element in the image may be classified as an interaction. So when we have extracted the gene box from the picture, we need to use background noise and paste it over the gene position. Such that the later detections such as interactions between the gene boxes, will not include the gene boxes as interactions. The procedure must be repeated for all elements until we only have the interactions and expression relations left in the png image. Figure 3.2 illustrates this process.



(a) (b)

Figure 3.2: In (a) all elements are present. In (b) all meta-data except interactions are extracted.

3.2 The KEGG KGML database

The KEGG KGML representation of pathway is not consistent with the hand drawn KEGG PNG image. There may also be nodes that are not present in the database, so the use of the nodes position in the database will exclude nodes in the manually drawn pathway image. Also, the additional meta-data is not present at all in the database. The additional meta-data is important to for the biologist who studies the image. If a gene product is within or a interaction/relation goes through an annotation, it gives the biologist vital information.

We compared the meta-data in the database[8], with several pathways. We found some similarities in the structure of the elements in the pathway map

[8]KEGG meta-data getter from Kvik: `https://github.com/fjukstad/kvik`

which we took the advantage of in our implementation. Such as the genes which are represented as boxes all have the same height and width. By exploiting the existing structure we can extract all the genes by filtering out all the other elements which does not fit in this gene representation.

3.3 OpenCV

The implementation of this project is built on the OpenCV library. It uses several computer vision algorithms in order to recognice the different elements within the . However, it is necessary to tune the OpenCV library. This requires knowledge about image processing.

To extract the meta-data from the PNG image, we have to copy the image and convert the copy to a binary image. This implies that the copy of the image only has 0's and 1's to represent itself. All the 0's is white and is considered as the background. Whenever the image processing algorithms find a pixel which is a 1, it is black and it represent an element. This could be a gene, interaction, context within a gene.

Chapter 4

Implementation

In this chapter we present the OpenCV functions we have used for image processing. Each section below is a separate function or method. We bescribe the motivation behind the implementation and present the results we achieved.

We will mention function calls to the OpenCV library, these are presented in *italics*. The python OpenCV library has legacy code, *cv*, and the up to date code, *cv2*.

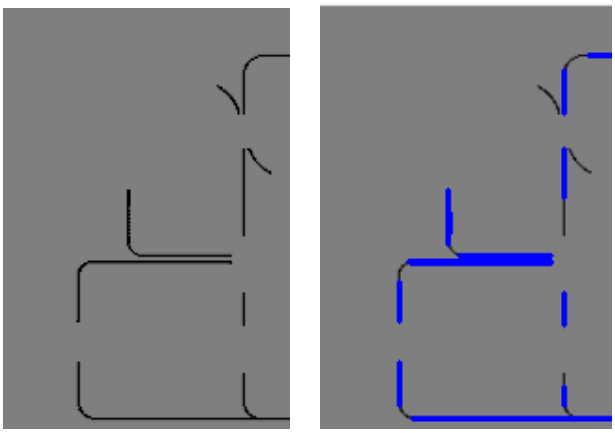
4.1 Houghlines method

We used the houglines method to find the lines on the pathway. This method should find all the lines: horizontal, vertical or sloped. We extract the lines to determine the interaction between different genes. This method does only returns straight lines, which implies that if an interaction has a curve, we have to find nearby interactions and merge these together.

We give the function, *cv2.HoughLines()*, its required inputs such as the binary image. The function has properties to look for straight lines which has space between them. This enables the extraction of stippled lines, because it allow the straight line to have spaces in it.

As seen on Figure 4.1, the houghline only finds the straight lines as expected. Some portion of the straight lines have not been included as a line representation. We tried several modification of inputs to the *cv2.HoughLines()*

function, such as modifying the threshold and angle resolutions, but this was the best result we had.



(a) Small pathway portion.

(b) Small pathway portion with houghline method.

Figure 4.1: Visualization of houghline method.

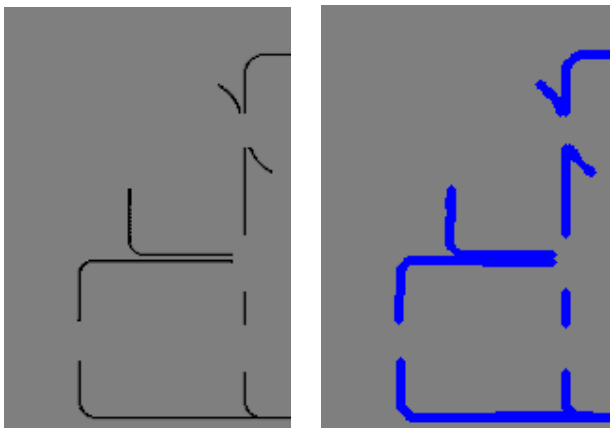
Based on our experiences using *cv2.Houghlines()* we decided not to use it. It did not satisfy our requirements for our representation of an interaction. In Figure 4.1 we can see that some lines only fractions are detected and no sloped lines are detected.

4.2 Edge detection

The goal with edge detection is to find the lines on the pathway. We want to use edge detection to find all the lines, even if they are not straight. The extracted lines are used to determine the interaction between different genes. We will therefore find contours within the pathway, but in order to do this we have to exclude all the other elements before applying this detection. Since we have no requirements for what an edge is, even textual context is classified as a contour.

Before we can find the contours within the image, we have to find the edges. We used first the *cv2.Canny()* function to retrieve all the edges from the image, but since all the edges were thick so it was unnecessary for our contour representation. The *cv2.Canny()* uses other functions with already specified

parameters. Therefore we used *cv2.Laplacian()* and *cv2.adaptiveThreshold()* with our own parameters, in order for the thickness to only be 1 pixel wide. And these two functions returns approximately the same as *cv2.Canny()*, but when we control the parameters we also control the output. This gives a list of 1 pixel wide lines. In order to find the contours we have used the function *cv2.findContours()*.



(a) Small pathway portion.

(b) Small pathway portion with edge detection.

Figure 4.2: Visualization of combined edge detection expressed as contours.

As seen on Figure 4.2 we have found every segment of all lines. On this small portion of a pathway, we have found 8 contours. In this portion there are 9 interactions, the contour at the bottom is two interactions that is merged together. To find every separate interaction between the genes, we need to check this. An interaction has only one end and one start, so this contour has to be split up. This can be checked by making sure the ends of a contour have genes in the end. If so, the contours is the start of an interaction

With this method we find all the interactions, but it does also classifies pixels that are not an interaction as an interaction. This detection must therefore be run after the other elements in the pathway have been extracted as presented in Figure 3.2

4.3 Gene detection

The goal with the gene detection method is to find all the genes. The genes are represented as square boxes which all has the approximate height and width. The information about the representation of the gene boxes, was found by using the KEGG KGML, Chapter 3.2. When we have square boxes with approximate same size, we can exploit this by using contours.

We used the same approach on the gene detection as the edge detection. But we can use the *cv2.Canny()* since we are not very interested in the box itself. It does not matter if the box is represented as 2 pixel wide. The interesting part of the gene is the context within the gene, which we will not consider at this stage. When the canny function has given us the edges, we send these edges to the *cv2.findContours()* function. These contours are useful to analyse the image for genes. We specify that the contours should have 4 corners and the width and height match the representation in the KEGG KGML. This way we exclude all the contours which are not squares and then those that do not fit the representation.

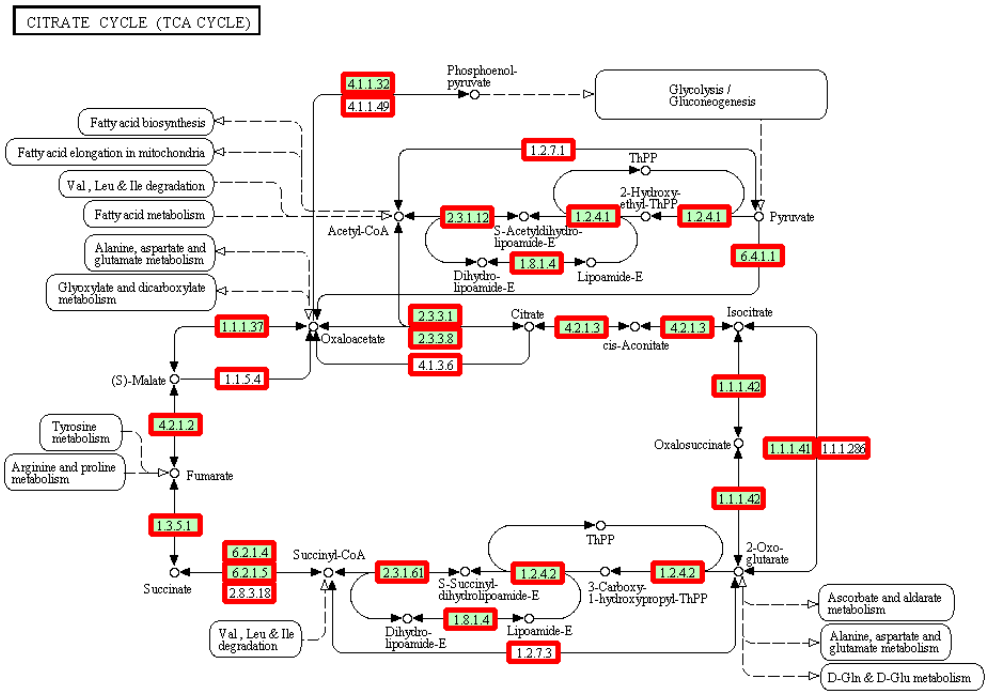


Figure 4.3: Pathway for citrate cycle with gene detection on.[9]

[9] `http://rest.kegg.jp/get/hsa00020/image`

The implementation does not currently find all genes. In Figure 4.3 all the genes are found and are highlighted with a red box. We evaluated the accuracy on this detection which is found in Chapter 5.

Chapter 5

Evaluation

We want to evaluate Amdex in terms of accuracy and how long time it uses for image processing. To check the accuracy of the pathway, Amdex printed how many match it found on each pathway. We highlighted the pathways as done in Figure 4.3 and counted how many genes that was not highlighted.

We evaluated gene detection accuracy by analysing a sample of 20 randomly selected pathways using Amdex. The result are in Table 5.1. For five pathways Amdex recognised all genes. For most pathways the accuracy was over 80%. For four pathways it was less than 80%. For the worst pathway Amdex only recognised 5% of the genes.

Amdex is very fast. The time used on one pathway is averagely 0.04 seconds, which is would be acceptable for running on locally computers. The measurements were for pathway images already downloaded to the machine.

5.1 Discussion

5.1.1 Gene box recognition

We evaluated the gene detection function. This is the most important function because in order to extract the interactions, all genes must be erased from the image.

Since the pathway maps are manually drawn on a machine, there may be

Pathway	Genes found	Total number of genes	Percentage of genes found	Execution time (s)
hsa00010	57	57	100.0	0.02
hsa00020	29	29	100.0	0.03
hsa00051	77	77	100.0	0.04
hsa01040	28	28	100.0	0.04
hsa05219	27	27	100.0	0.03
hsa00061	170	171	99.4	0.04
hsa05322	47	49	95.9	0.04
hsa00450	21	23	91.3	0.02
hsa05144	50	56	89.2	0.07
hsa04010	103	125	82.4	0.05
hsa05033	23	26	88.4	0.06
hsa04915	47	54	87.0	0.04
hsa05100	52	61	85.2	0.03
hsa05034	51	62	82.2	0.06
hsa05416	30	37	81.0	0.05
hsa05215	34	42	80.9	0.04
hsa03410	35	60	58.3	0.04
hsa04064	67	125	53.6	0.05
hsa03440	15	55	27.2	0.04
hsa00190	9	182	4.9	0.04
Total	972	1346		0.83
Average	48	67	80.3	0.04

Table 5.1: Experimental data

human errors. For example as when drawing an edge from a node, the edge may start within the node. These small pixel errors are not visible for humans, but may interfere with the algorithm that extract the nodes from the pathway picture. In Figure 5.1 the human expert have drawn the interactions too far and it ends within the gene box, which then causes the box not to be detected. The interaction is only a single pixel inside the box, but that is all it takes to exclude the box.

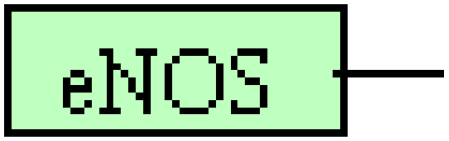


Figure 5.1: Human error on a gene box.[10]

Our approach for recognising gene boxes may add additional gene boxes as seen on Figure 5.2. The space between the boxes are exact the same as the representation of the gene box. It is the *cv2.findContours()* function that returned these as a contour. We have not solved these issues yet. One possible way to solve this is to check the edges of the contour for color verification. If all edges actually have a black pixel, if not then it is not a gene box.

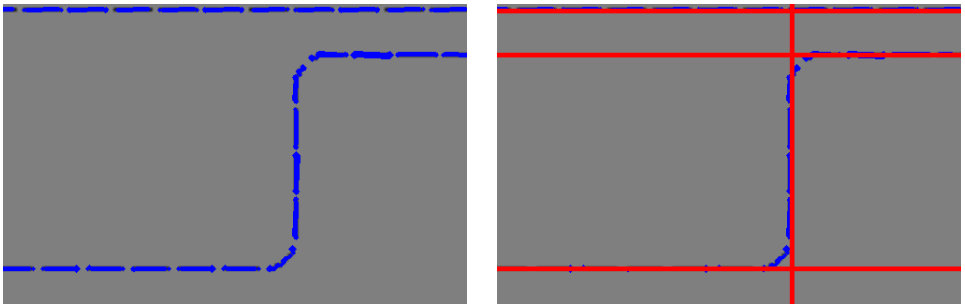
(a) Small sample of genes with interactions.

(b) Gene detection fault.

Figure 5.2: A subset of a pathway illustration gene detection faults.[11]

5.1.2 Edge detection

For edge detection, there is one drawback by using contours method, rather than the houghline method. The stippled lines are being discovered as a several interactions, instead of a separate interaction. But when discovering several small interactions, this could indicate that there is a stippled line. We considered using the houghline method on these, which would result in something like Figure 5.3.



(a) Stippled lines marked with *cv2.findContours()*.

(b) Stippled lines with a probalistic houghline method.

Figure 5.3: Visualization of the stippled line problem. [12]

[10] `http://rest.kegg.jp/get/hsa04915/image`
[11] `http://rest.kegg.jp/get/hsa04915/image`
[12] `http://rest.kegg.jp/get/hsa05416/image`
[14] `http://rest.kegg.jp/get/hsa00190/image`

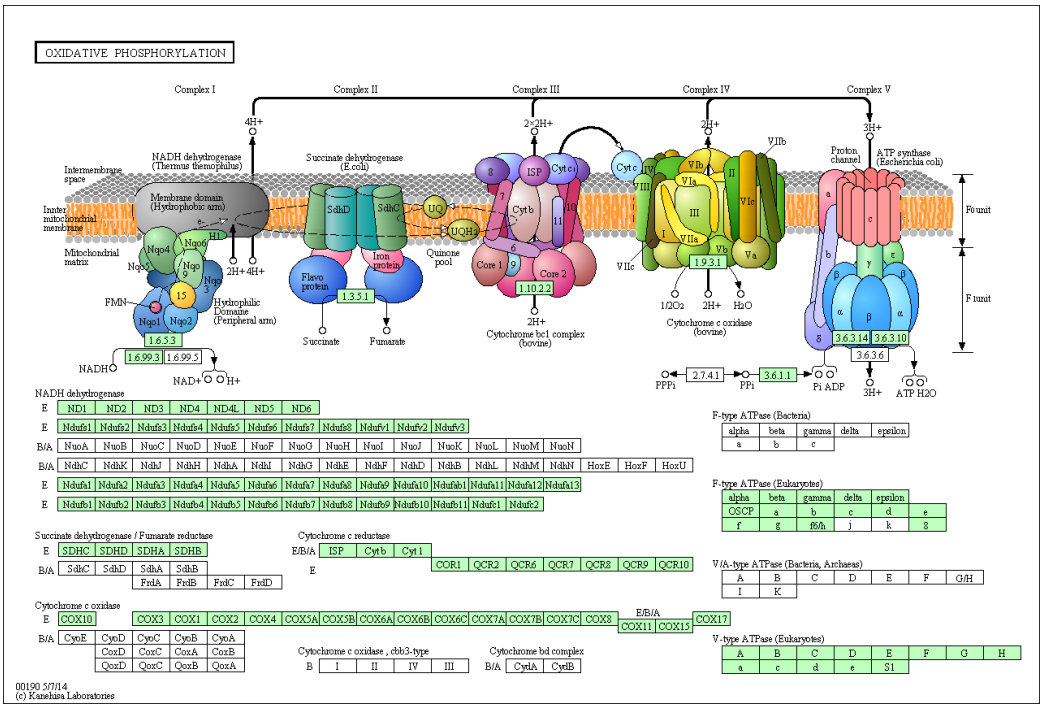


Figure 5.4: Pathway for oxidative phosphorylation which has additional meta-data.[14]

5.1.3 Additional meta-data

Additional meta-data is not supported in Amdex at all. This meta-data is challenging to detect since these tend to be unique and vary in size, shapes and color. Figure 5.4 and Figure 5.5 are examples of pathways with extra meta-data. This additional meta-data is important for the user, so future versions of Amdex should support these.

5.1.4 Lessons learned

It may seem like an easy problem to extract meta-data from PNG images. However, we quicly realized that this was challenging. The prototype recognised the most important elements and is therefore a useful first step in solving the problem. But a lot more additional work is required to reach an

[16] `http://rest.kegg.jp/get/hsa03440/image`

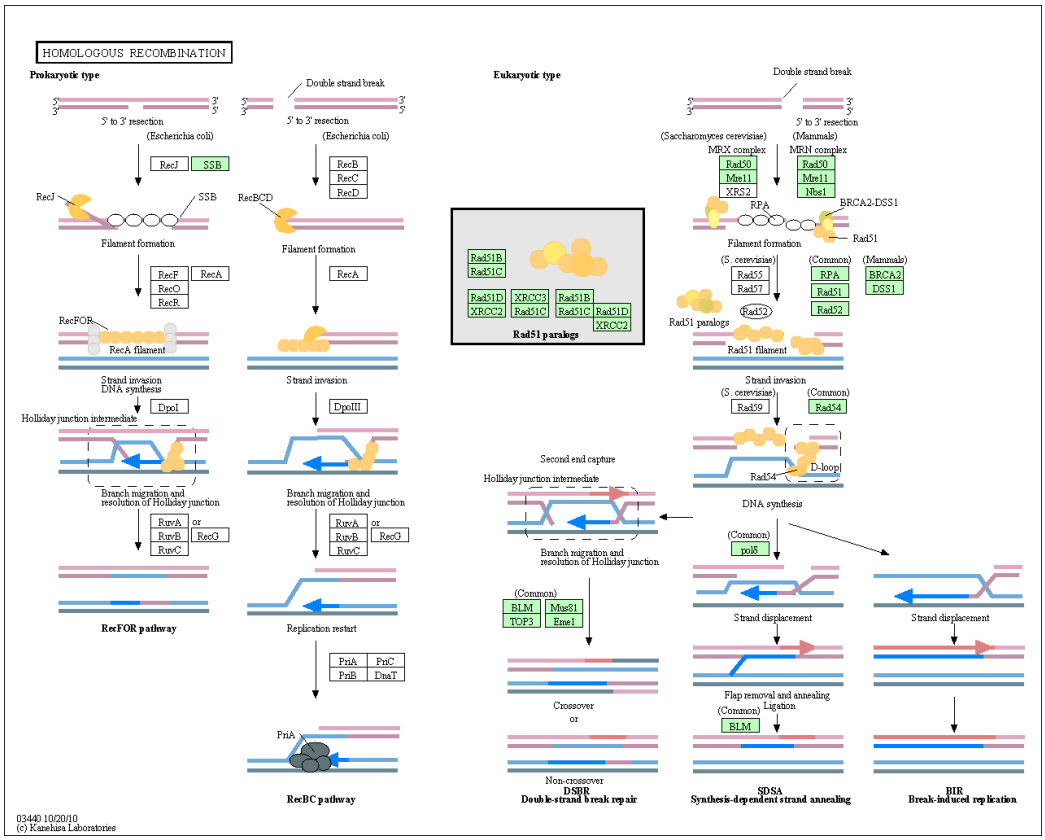


Figure 5.5: Pathway for homologous recombination which has additional meta-data.[16]

accuracy and precision to fully solve the problem.

OpenCV is a powerfull library for image processing, which is why we chose this in the first place. But the library requires in-depth knowledge of the implemented algorithms/methods. The documentation is well-defined, but is challenging to understand without a prior backgrount in pattern recognition or image processing. We received help some help from collaborators which has experience with the OpenCV library and physics background. For example when we used the houghline method on our interactions, we did not initially find every straigth line. We found all vertical lines in some areas and the horizontal in another area. We were advised to use contours which would find every seperate contour. These does not only find the straight lines such as vertical and horizontal, but also the sloped interactions.

Chapter 6

Conclusion

This thesis has described the challenges of extracting meta-data from a manually drawn pathway image.

There are no out-of-box tools that can recognise all meta-data elements in KEGG images. There are methods in OpenCV that can be used to recognise some/most meta-data elements, but the parameters of these methods must be tuned. Since the pathway maps images are manually drawn, there are small errors within the images. These cause problems for finely tuned OpenCV methods.

Our initial results show that the accuracy is as good as KEGG KGML, since we cannot find all genes in an image. Since all gene boxes have not been extracted, the tool can not currently recognise gene interactions.

We believe that the tool is important for further exploration of biological pathway. The ability to make a biological pathway dynamic, enables more interactive implementations. If all meta-data could be extracted, even more cross-viewing through the pathways could be done.

6.1 Future Work

Several interactions may merge together into one interaction towards a gene. The tool, Amdex does not split these up into their representative interactions.

Interactions which are represented as stippled lines are another problem

which is not implemented in this tool. A possible way to solve this have been presented in Subsection 5.1.2.

Additional meta-data support is a requirement for fully functionally automated meta-data extraction from KEGG pathways. These additional meta-data may be unique in a single pathway, and the KEGG pathways may change over time.

Integrate Amdex into Kvik for a more flexible visualization allowing users to modify pathway maps and move nodes interactively.

References

[1] KEGG Pathway Database: http://www.genome.jp/kegg/pathway.html

[2] Ingenuity Pathway Analysis: http://www.ingenuity.com

[3] BioCarta: http://www.biocarta.com

[4] MetaCyc Pathway Analysis: http://metacyc.org

[5] OpenCV: http://opencv.org

[6] KEGG Pathway Maps Information: http://www.genome.jp/kegg/document/help_pathway.html

[7] KEGGViewer: http://www.ebi.ac.uk/Tools/biojs/registry/Biojs.KEGGViewer.html

[8] Lex, Alexander, et al. Entourage: Visualizing relationships between biological pathways using contextual subsets. Visualization and Computer Graphics, IEEE Transactions on, 2013, 19.12: 2536-2545.

[9] Partl, Christian, et al. enRoute: Dynamic path extraction from biological pathway maps for in-depth experimental data analysis. In: Biological Data Visualization (BioVis), 2012 IEEE Symposium on. IEEE, 2012. p. 107-114.

[10] Collins, Christopher, et al. Bubble sets: Revealing set relations with isocontours over existing visualizations. Visualization and Computer Graphics, IEEE Transactions on, 2009, 15.6: 1009-1016.

[11] Bjørn Fjukstad. Kvik: Interactive exploration of genomic data from the NOWAC postgenome biobank. UiT The Arctic University of Norway, 2014.

[12] J. Gmez, et al. BioJS: an open source JavaScript framework for biological data visualization., Bioinformatics (Oxford, England), vol. 29, pp. 11034, 2013.

Appendix A

Source Code

The source code follows on a CD-ROM.