MASTER'S THESIS

Interactive Visualization of Annotated DNA Sequences

by

T.H.J.M. Peeters

Supervisors: H.M.M. van de Wetering  
             J.J. van Wijk
Advisor:     M.W.E.J. Fiers

Eindhoven, January 12, 2004
Abstract

In molecular biology DNA sequences are studied intensively. Laboratory experiments and automated analyses are used to identify subsequences (i.e. to produce annotations on the DNA sequence) that are potentially interesting. Visual inspection is needed to decide if these are meaningful, and to find other interesting patterns and structures. Current tools for the visualization of annotated DNA sequences often fall short. They do not provide smooth interaction, the user suffers from lack of context when viewing small parts of the huge datasets, and it is not easy to compare datasets.

The goal of this project was to develop an interactive visualization tool that can be used for browsing and comparing very large DNA sequences and their annotations in an intuitive way. After an introduction of the problem, requirements, and current solutions, this new tool, called DNAVis, will be described and demonstrated.
## Contents

Preface 4

1 Introduction 5

2 Biological background 7
   2.1 Cells 7
   2.2 Genetic material (DNA) 7
   2.3 RNA transcription 9
   2.4 Protein synthesis 10
   2.5 Micro RNA 12

3 Problem description 16
   3.1 Input specification 16
      3.1.1 DNA sequences 16
      3.1.2 Annotations 17
      3.1.3 Example annotation types 18
      3.1.4 Example datasets 19
      3.1.5 Data size 20
   3.2 Visualization requirements 21
      3.2.1 Annotations 21
      3.2.2 Levels of detail 22
      3.2.3 Navigation and flexibility 23
      3.2.4 Comparison 23

4 Existing methods 25
   4.1 Web-based genome browsers 25
      4.1.1 Datasets 26
      4.1.2 Levels of detail 26
      4.1.3 Navigation and flexibility 29
      4.1.4 Comparison 29
   4.2 Apollo Genome Browser 31
      4.2.1 Datasets 31
      4.2.2 Levels of detail 31
      4.2.3 Navigation and flexibility 31
      4.2.4 Comparisons 33
   4.3 Visualizing similarities 34
      4.3.1 Dot plots 34
4.3.2 Alignment viewers ........................................... 35
4.3.3 3D Pattern visualization ................................. 36
4.4 Summary ......................................................... 37

5 A new tool: DNAVis .............................................. 38
  5.1 The bar view ................................................ 38
    5.1.1 Linear visualization .................................. 40
    5.1.2 Bars ...................................................... 40
    5.1.3 Indication of the visible subdomain .................. 40
    5.1.4 Sequence ................................................ 41
    5.1.5 Annotations ............................................. 42
    5.1.6 Multiple, configurable annotation bars .............. 42
    5.1.7 Perspective walls ..................................... 43
    5.1.8 Interaction and navigation ............................. 46
  5.2 The matrix view .............................................. 48
    5.2.1 Comparison matrix .................................... 48
    5.2.2 Navigation .............................................. 50
    5.2.3 Conclusion .............................................. 50
  5.3 Multiple views .............................................. 50
    5.3.1 Single view limitations ............................... 50
    5.3.2 View behaviour ......................................... 51
    5.3.3 Conclusion .............................................. 52
  5.4 Evaluation ................................................... 52
    5.4.1 Adherence to requirements ............................. 54
    5.4.2 Usefulness .............................................. 55

6 Future work ..................................................... 57
  6.1 Database connectivity ..................................... 57
  6.2 Searching .................................................... 58
  6.3 Selection and bookmarks .................................. 59
  6.4 Visualization ................................................ 59
    6.4.1 Bar view ................................................ 59
    6.4.2 Matrix view ............................................ 60
  6.5 Conclusion ................................................... 61

A Requirements .................................................. 62

B Implementation ............................................... 64
  B.1 Platform ..................................................... 64
  B.2 System architecture ....................................... 65
    B.2.1 Observers ................................................ 65
    B.2.2 Data ...................................................... 66
    B.2.3 Views and canvasses .................................. 68
C File formats

C.1 FASTA ................................................................. 71
C.2 General Feature Format ........................................ 72
C.3 Dataset specification ............................................. 73

Bibliography ............................................................ 75
Preface

This report describes the result of research on the visualization of large biological datasets, particularly annotated DNA sequences. The research was carried out as my master’s project at the Eindhoven University of Technology (TU/e) and was supervised by Huub van de Wetering and Jack van Wijk. Huub and Jack are both visualization experts and work at the computer science department of the TU/e.

The project resulted in a tool called DNAVis, which will be put into use at Plant Research International in Wageningen, where research is done on (among other areas) plant genomics. Mark Fiers, who works at Plant Research International helped in defining the problem and provided advise on (micro-) biological subjects.

My thanks go to Huub and Jack for supporting me with their knowledge and experience in the field of information visualization and computer science, and for reviewing the drafts of the chapters of this report. I also thank Mark for providing feedback on DNAVis, reviewing this report, and describing the visualization problem from a biologists point of view while keeping it understandable for a layman in microbiology such as myself.

Tim Peeters
Chapter 1

Introduction

The factors that determine the inherited characteristics of organisms have been the subject of research and debate for centuries. The substance that specifies these characteristics is called the genetic material of an organism and we now know that most organisms utilize deoxyribonucleic acid (DNA) as their genetic material.

DNA molecules are very large molecules that consist of sequences of so-called nucleotides. Even though for some species (including the human species) full DNA sequences are available, the encoding of many characteristics in DNA and how they are expressed is still unclear or even unknown. So DNA and its expression is still an important area of research.

One of the institutes where research on DNA is being performed is Plant Research International in Wageningen. The research that is done there includes the analysis of the genetic material of plants such as Arabidopsis thaliana (see figure 1.1), which is widely used as a model organism in plant biology.

Because of their sizes (in the order of millions of nucleotides) it is practically impossible to store and analyse complete DNA sequences without using computers. Annotations indicate the locations on a DNA sequence of, for example, genes, which are the basic units of heredity. They can be stored alongside the sequence for inspection and analysis.

Automated analysis can be used to detect potential interesting subsequences and to annotate them. Manual inspection and analysis is needed to verify the results of the automated analysis and to detect new

Figure 1.1: View of Arabidopsis thaliana. Source: http://www.arabidopsis.org/
patterns, relations and structures. However, manual inspection is hard because of the sheer sizes of the datasets. In order to solve this problem, information visualization [1] can be used to make the data more accessible. This is what the project focused on: the visualization of DNA sequences and annotations on those sequences.

This report describes various aspects of the visualization problem and the tool which was created to solve the problem. First, in chapter 2, a biological background is given which is needed to fully understand the problem. Chapter 3 gives a detailed description of the problem and a set of requirements to which the tool must adhere. In chapter 4 an overview is given of existing tools that partially solve some of the problems described in chapter 3, and chapter 5 describes the newly created tool, DNAVis, that satisfies all requirements of chapter 3. Finally, chapter 6 describes possible improvements and expansions of DNAVis.
Chapter 2

Biological background

This chapter gives a biological background that is needed to understand both the problems and solutions that are described in the following chapters. One of the most important subjects that is dealt with is DNA, of which the chemical structure is described in section 2.2. But first, section 2.1 explains some important features of cells. This includes the location of DNA in cells and some important terms that are used in later chapters. Section 2.3 explains what RNA is and how it is transcribed from DNA and section 2.4 explains how RNA is used to synthesize proteins. Finally, section 2.5 covers a recently discovered type of RNA: Micro RNA.

2.1 Cells

All organisms, except viruses, consist of one or more cells. Each cell contains cytoplasm and is surrounded by a membrane. The membranes around cells function as a separation between a cell’s interior and its environment. Cell membranes can selectively let substances enter and leave the cell, but they also have other important functions, such as energy storage [2].

There are two kinds of cells: prokaryotic and eukaryotic cells. Eukaryotic cells have organelles, which are membrane-enclosed internal parts of a cell that have a specific function. The most prominent organelle is the nucleus [3], which contains most of the cellular DNA (see section 2.2). Prokaryotic cells have a simpler internal structure without a defined, membrane-limited, nucleus or other organelles. Organisms have either prokaryotic or eukaryotic cells. If an organism has prokaryotic cells it is called a prokaryote. Organisms with eukaryotic cells are called eukaryotes. All prokaryotes are single-celled organisms [2], but single-celled organisms are not always prokaryotes. All plants and animals are eukaryotes.

2.2 Genetic material (DNA)

“The genetic material of an organism is the substance that contains the information specifying the inherited characteristics of that organism.” [4]
With the exception of some viruses in which the genetic material is RNA, all organisms utilize DNA as their genetic material. Both DNA (deoxyribonucleic acid) and RNA (ribonucleic acid) are nucleic acids. A nucleic acid consists of a long ‘chain’ of nucleotides. Each nucleotide has the following components:

1. A sugar. In RNA the sugar is ribose and in DNA it is deoxyribose.
2. A phosphate attached to a carbon of the sugar.
3. A base attached to another carbon of the sugar.

The sugar and phosphate form the so-called backbone of the nucleic acid that is constant across the whole molecule. The bases, which can vary, are attached to the backbone. DNA and RNA contain the bases cytosine (C), guanine (G) and adenine (A). DNA also contains the base thymine (T). Uracil (U) is found only in RNA. A part of a nucleic acid can be represented as shown in figure 2.1. Because the backbone is constant, a DNA or RNA chain can be represented by a sequence of letters representing the chain’s bases, e.g. ATCCGCAATAC. Two DNA strands can be coiled around each other to form a double-stranded helix (see figure 2.2). Such a double-stranded helix can be formed because the bases of DNA will be hydrogen-bonded with the bases that are their complement to form base-pairs. In DNA, A and T are each other’s complement and so are C and G. In RNA, U takes over the role of T as the complement of A. Figure 2.3 shows a systematic representation of a part of a DNA molecule.

If a cell divides, first the cell’s DNA must be duplicated. For this to happen, the two strands of DNA are separated and each of the four different nucleotides are paired with ‘free’ nucleotides that are their complement such that two new DNA molecules are formed. Figure 2.4 shows a simple representation of this process. The actual process is very complicated for multiple reasons, including the required uncoiling of the double helix that is necessary for the two strands to be separated. In eukaryotic cells the replication may be initiated (under the influence of the enzyme DNA polymerase) on multiple locations along the DNA molecule at once. Parts of a DNA molecule are also separated if RNA
transcription takes place (see section 2.3). DNA polymerase can be used in an ingenious way to determine the nucleotide sequence of a DNA strand [4]. This can usually not be done for the whole strand at once, but for very small subsequences of up to 800 bases. These subsequences can be assembled into large subsequences called contigs which subsequently are joined together into whole sequences.

2.3 RNA transcription

“A gene is the basic unit of heredity consisting of the nucleotide sequences of DNA involved in synthesis of a protein.” [4]

For a gene to be expressed (in a protein), its genetic information must first be transferred from DNA to RNA. This process of transferring information is called transcription. Transcription is catalyzed by enzymes called RNA polymerases and starts at particular sites on a DNA molecule called promoters. Promoters have a loosely defined nucleotide sequence to which a RNA polymerase can bind. A complex of RNA polymerase and several other proteins opens up a part of the double-stranded DNA molecule and initiates the transcription of that part. Transcription stops when a ‘stop sequence’ or terminator is encountered. Only a small part of a DNA strand is transcribed (in Arabidopsis thaliana around 30%). Depending on its nucleotide sequence, either strand of the DNA molecule can be used for RNA transcription. But transcription occurs only in one direction along
Figure 2.5: Schematic representation of RNA transcription. The top and middle (coding and template) strands are DNA strands. The bottom strand is the RNA strand that is being transcribed from the template strand. Vertical lines indicate hydrogen bonds in the DNA molecule. These bonds are temporarily broken in the segment where RNA transcription takes place.

a strand. As with DNA replication the result is acquired by complementary base pairing. The resulting molecule, or primary transcript, is single-stranded and consists of nucleotides that are the complements of the nucleotides of the strand they where transcribed from: the template strand. The primary transcript does not contain DNA nucleotide T, but has RNA nucleotide U as the complement of A (see section 2.2). Except for the replacement of T by U, the primary transcript has the same sequence as the DNA strand complementary to the template strand: the coding strand. Figure 2.5 shows a representation of a part of a DNA molecule and a RNA sequence that is being transcribed from it.

Before the result of the transcription is used in the synthesis of proteins it will be modified by specific enzymes. These modifications include RNA splicing, which is the removal of sequences in a gene that do not encode amino acids. These non-coding sequences are called introns. The remaining (coding) segments are called exons and are joined together in the RNA molecule.

2.4 Protein synthesis

There are multiple kinds of RNA, including messenger RNA (mRNA), transfer RNA (tRNA) and ribosomal RNA (rRNA). Proteins, which consist of a sequence of amino acids, are encoded by mRNA. Each amino acid in the sequence corresponds to a codon of the mRNA molecule. A codon is a triplet of nucleic acids that corresponds to an amino acid. This correspondence between nucleotide triplets and amino acids in proteins is called the genetic code and is shown in table 2.1. The translation of mRNA to an amino acid sequence takes place on in the so-called ribosomes. These ribosomes contain rRNA. Because amino acids do not line up against the mRNA template independently, they are transported and aligned by tRNA.

*The tRNA molecules are small, single-stranded nucleic acids ranging in size from 73 to 93 nucleotides. Due to pairing of complementary base-sequences short double-stranded regions form, causing the
<table>
<thead>
<tr>
<th>First position</th>
<th>Second position</th>
<th>Third position</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>Phe</td>
<td>Tyr</td>
</tr>
<tr>
<td></td>
<td>Phe</td>
<td>Ser</td>
</tr>
<tr>
<td></td>
<td>Leu</td>
<td>Ser</td>
</tr>
<tr>
<td></td>
<td>Leu</td>
<td>Ser</td>
</tr>
<tr>
<td>C</td>
<td>Leu</td>
<td>Pro</td>
</tr>
<tr>
<td></td>
<td>Leu</td>
<td>Pro</td>
</tr>
<tr>
<td></td>
<td>Leu</td>
<td>Pro</td>
</tr>
<tr>
<td></td>
<td>Leu</td>
<td>Pro</td>
</tr>
<tr>
<td>A</td>
<td>Ile</td>
<td>Thr</td>
</tr>
<tr>
<td></td>
<td>Ile</td>
<td>Thr</td>
</tr>
<tr>
<td></td>
<td>Ile</td>
<td>Thr</td>
</tr>
<tr>
<td></td>
<td>Met</td>
<td>Thr</td>
</tr>
<tr>
<td>G</td>
<td>Val</td>
<td>Ala</td>
</tr>
<tr>
<td></td>
<td>Val</td>
<td>Ala</td>
</tr>
<tr>
<td></td>
<td>Val</td>
<td>Ala</td>
</tr>
<tr>
<td></td>
<td>Val</td>
<td>Ala</td>
</tr>
</tbody>
</table>

Table 2.1: The genetic code. *Italic* codons are used for initiation. U, C, A and G are abbreviations for nucleic acids. Phe, Ser, Tyr, Cys, Trp, Pro, His, Arg, Gln, Ile, Thr, Asn, Ser, Lys, Met, Val, Ala, Asp, Glu and Gly are abbreviations for amino acids. Stop codons are used for indicating the end of an amino-acid sequence.
molecule to fold into a structure in which open loops are connected
to one another by double-stranded stems (figure 2.6).” [4]

The three bases shown at the bottom of figure 2.6 are called the anticodon.
This anticodon can base-pair with a mRNA codon if the bases of the codon
and the anticodon are each other’s complement. At the top there is an amino
acid attachment site to which the amino acid corresponding (see table 2.1) to
the mRNA codon that base-pairs with the tRNA anticodon is linked. Now in
protein synthesis, tRNA molecules attached to amino acids align with a mRNA
molecule and the amino acids are linked by peptide bonds. After linking the
amino acids, the tRNA molecules are cut off so that a sequence of amino acids
remains. Synthesis stops when a “Stop” codon occurs in the mRNA molecule.
The resulting amino acid sequence can be an enzyme, a hormone, or another
protein.

2.5 Micro RNA

Non-coding RNAs or ncRNAs are all RNAs that do not encode proteins. This
set of RNAs includes all RNAs except mRNA [5]. Besides the well-known rRNA
and tRNA, there are other, less-known types of ncRNA. This includes Micro
RNA or miRNA. miRNAs were only recently discovered and they are proven
to regulate the production of proteins from protein-encoding genes [6, 7]. A
miRNA is a part of a miRNA precursor; a small RNA molecule that is folded
as a hairpin. Figure 2.7 shows such a miRNA precursor stem from C. elegans
(a worm). The enzyme Dicer cuts out a part of this molecule to form miRNA.
This newly formed miRNA molecule can bind to certain target mRNAs (see
figure 2.8) and slows down protein production by that mRNA by doing so.
Some miRNA-encoding parts have been found in DNA sequences, but much is
still unknown about miRNA and other non-coding RNAs so more research is
needed.
Figure 2.6: A tRNA cloverleaf structure. B’s represent the bases. The numbers indicate a numbering of the connected bases from 1 to 76. ‘*’ is the amino acid attachment site. The three bases at the bottom marked with ‘#’ form the anticodon. Horizontal (‘—’) and vertical (‘|’) lines represent hydrogen bonds between two bases. Bases 44 to 48 form an extra arm of which the length varies in different tRNAs.
Figure 2.7: C. elegans miR-84 and its precursor stem [5, 6]. As can be seen here, it is possible for G and U to form a base-pair.
Figure 2.8: An example of how a miRNA molecule could (imperfectly) bind to a part of a mRNA molecule, blocking protein production.
Chapter 3

Problem description

In the previous chapter some biological background was given which is needed to understand the problem. In this chapter, the problem is specified as an information visualization [1] problem. Section 3.1 gives a formalization of relevant biological features and visualization requirements are given in section 3.2.

3.1 Input specification

This section describes the datasets that will be used, where a dataset \( D = (N, S, A) \) consists of a name \( N \), a DNA sequence \( S \) and a collection \( A \) of annotations on sequence \( S \). The unique string \( N \) can be used for identifying the dataset. The definition of a DNA sequence is given in section 3.1.1 and in section 3.1.2 annotations are defined.

3.1.1 DNA sequences

As seen in section 2.2, DNA can be represented by a sequence consisting of four different bases which are identified by the letters \( A, T, C \) and \( G \). Here we define a DNA sequence as a string that adheres to the following regular expression [8]:

\[
e_{\text{dna}} = (A + T + C + G + N)^+
\]

So a DNA sequence is a non-empty string consisting of letters from the alphabet \( \Sigma_{\text{dna}} = \{A,T,C,G,N\} \). In this alphabet \( A, T, C \) and \( G \) represent the bases adenine, thymine, cytosine and thymine. A letter \( N \) in the sequence means that there is a base at that position but it is not known which one. The actual files containing DNA sequences are in FASTA format (see appendix C.1) which has an alphabet \( \Sigma_{\text{fasta}} \) that includes representations for bases that are neither exactly known (\( A, T, C \) and \( G \)) nor completely unknown (\( N \)). For example, \( W \in \Sigma_{\text{fasta}} \) stands for “\( A \) or \( T \)”. Likewise, \( N \) in both \( \Sigma_{\text{dna}} \) and \( \Sigma_{\text{fasta}} \) means “\( A \) or \( T \) or \( C \) or \( G \)”. Of all DNA sequences we had for testing only 70 out of 118962460 bases were in \( \Sigma_{\text{fasta}} \setminus \Sigma_{\text{dna}} \). Because of this and to avoid unnecessary complexity in this report, here we use the simplified definition so that the language of all DNA sequences \( L_{\text{dna}} \) is the language denoted by \( e_{\text{dna}} \):

\[
L_{\text{dna}} = L(e_{\text{dna}}) = \{A, T, C, G, N, AA, AT, AC, AG, AN, TA, TT, \ldots\}
\]
For a natural $n > 0$ and a DNA sequence $S = b_1 \ldots b_n$ with $(\forall i : 1 \leq i \leq n : b_i \in \Sigma_{dna})$, we define:

$$\text{length}(S) = n$$

$$S[i] = b_i \text{ for } 1 \leq i \leq \text{length}(S)$$

Note that indexing starts at 1.

DNA, in most forms found in the cell, consists of two strands: a forward or coding strand and a reverse or template strand. Naming one strand forward and the other one reverse is completely arbitrary. A DNA sequence directly defines the bases of one strand, which we name the forward strand. The reverse strand can be computed by taking the complement of a DNA sequence that defines the forward strand, where the complement is obtained by applying the function $\text{compl}$ of type $L_{dna} \rightarrow L_{dna}$:

$$\text{compl}(xS) = \text{compl}(S) \text{ compl}(x)$$

where $x \in \Sigma_{dna}$ and $S \in L_{dna}$

$$\text{compl}(A) = T$$

$$\text{compl}(T) = A$$

$$\text{compl}(C) = G$$

$$\text{compl}(G) = C$$

$$\text{compl}(N) = N$$

Obviously when $\text{compl}$ is applied to a DNA sequence representing the reverse strand the forward strand is obtained. Note that the reverse strand is read in the opposite direction as the forward strand, which is important in transcription and duplication.

### 3.1.2 Annotations

Besides a DNA sequence $S$, a dataset also consists of a collection $\mathcal{A}$ of annotations on $S$. An annotation $\alpha \in \mathcal{A}$ can be represented by the following tuple:

$$\alpha = (\text{type}, \text{source}, \text{strand}, \text{regions}, \text{properties})$$

where the attributes are specified as follows:

- $\text{type}: \text{string}$ identifies the type (for example “Gene”) of the annotation.

- $\text{source}: \text{string}$ identifies the source of the annotation. This can be for example the program used to predict the occurrence of this annotation or the public database the annotation was acquired from.

- $\text{strand} \in \{+,-,'.'\}$ indicates whether the annotation applies to the forward strand (‘+’), the reverse strand (‘-’), or the strand is of no importance (‘.’).

- $\text{regions} \subseteq \text{Regions}(S) \setminus \emptyset$ is a non-empty set of regions of $S$. The definition of $\text{Regions}$ is:

$$\text{Regions}(S) = \{(i,j)|1 \leq i \leq j \leq \text{length}(S)\}$$
where $i$ and $j$ denote positions in $S$ and specify the subsequence $S[i] \ldots S[j]$ of $S$ that the annotation applies to. Although many annotations have only one region, multiple regions per annotation are allowed and these regions may overlap. Sometimes the type of an annotation dictates the number of regions it must have. The region $\text{region}(\alpha)$ of an annotation $\alpha$ is defined as $(\text{min}, \text{max})$ where $\text{min}$ is the minimum of all start positions and $\text{max}$ the maximum of all end positions of the regions of $\alpha$.

$\text{properties} \in \mathbb{P}(\text{string} \times \Omega)$, where $\Omega$ is an arbitrary type, contains (possibly optional) extra properties that are not contained in the standard attributes of an annotation. The $\text{string}$ part of a property is its name and $v \in \Omega$ is the property’s value. Depending on the type of the property, there may be restrictions on (the type of) $v$. Note that $v$ can also be a set of values.

To further clarify the meaning of the attributes of an annotation, section 3.1.3 describes some important annotation types and 3.1.4 gives two sample datasets.

Annotations as described in this section are loaded from files in GFF format. See appendix C.2 for a description of this file format. Section 3.2 describes which features of the datasets are to be visualized.

3.1.3 Example annotation types

There can be many types of annotations in a dataset. This section describes three types that are relevant in our context and that show the variation in annotation types.

**Genes**

Annotations with type “Gene” represent genes. Genes have introns and exons (see section 2.3) and the exons are explicitly defined by the regions of the annotation. Those parts inside the region of the annotation that are not exons are implicitly defined as introns. Annotations representing genes have a property $\in \text{string} \times \Omega$ of which the name is “Sequence” and the value a string representing the name of the gene, for example “At1g01330”. Other properties are also possible.

**miRNA**

Possible occurrences of miRNA (see section 2.5) are represented by annotations of type “miRNA_candidate” which are predictions of sections of DNA that may be transcribed to miRNA molecules. Each miRNA-candidate annotation has two regions that represent the two parts of a precursor miRNA molecule that can stick together (see section 2.5). The subsequences of the DNA sequence in the two regions of a miRNA candidate should approximately be each other’s reverse complement.
Similarities

Annotations of type “Similarity” are special, because they do not always apply to only one DNA sequence. These annotations are used to describe similarities between two different regions on a single or two different DNA sequences. We name the two DNA sequences (which may be the same) on which a similarity is defined $S_1$ and $S_2$. Each similarity has one region $r_1 \in Regions(S_1)$ to specify a part of $S_1$. This part is in some way similar to a part of $S_2$. To identify the part of $S_2$ that it is similar to, the annotation has a property with name “Target” and a value $v$ of type $string \times Regions(S_2)$. The $string$ part of $v$ contains the name of $S_2$ and is used to identify that sequence. The other part of $v$ is a region $r_2 \in Regions(S_2)$. The similarity may be that the subsequences of $S_1$ and $S_2$ specified by $r_1$ and $r_2$ are the same. Other similarities, such as a resemblance of the structures of genes in two datasets, are also possible. An example of a similarity and annotations of other types are given in section 3.1.4.

3.1.4 Example datasets

This section gives two imaginary (and biologically improbable) sample datasets $D_1$ and $D_2$ and then explains the meanings of their annotations:

$$D_1 = ("I", S_1, \emptyset)$$

$S_1 = \text{GT CGCAACTGTAGAAATCCATAA}$

$$D_2 = ("II", S_2, A_2)$$

$S_2 = \text{ACAT GCGT AT AAAT CCGAT CNGGCT AT AAGCAG}$

$A_2 = \{\alpha, \beta, \gamma\}$

where

$\alpha = ("\text{Gene}\), "\text{SourceX}\), '+', \{(3, 13), (20, 29)\}, prop_\alpha)$

$prop_\alpha = \{("\text{Sequence}\), "\text{Gene001}\), ("\text{Note}\), "\text{Example}\)\}$

$\beta = ("\text{ncRNA}\), "\text{SourceX}\), '–', \{(9, 15)\}, \emptyset\)$

$\gamma = ("\text{similarity}\), "\text{ProgY}\), '–', \{(7, 16)\}, prop_\gamma)$

$prop_\gamma = \{("\text{Target}\), ("II\), (11, 20)\})$

$D_1$ consists of a DNA sequence $S_1$ of length 24 and no annotations. $D_2$ consists of a DNA sequence $S_2$ of length 33 and a set $A_2$ containing three annotations. Of course real datasets have much longer DNA sequences and much bigger sets of annotations.

Annotation $\alpha$ represents a gene on $S_2$ and was acquired from some database called “SourceX”. It is located on the forward strand and has exons from bases 3 to 13 and 20 to 29. It has one, implicitly defined, intron between the two exons from bases 14 to 19. The property with name “Sequence” defines the gene’s name as “Gene001” and there is another property with name “Note” that has value “Example”. The region of the gene and its intron and exons are shown in figure 3.1.

The other annotation shown in figure 3.1 is $\beta$. This is a simple annotation of type “ncRNA” that is located on the reverse strand and has the same source as $\alpha$. It has only one region (9, 15) and no properties.
|<------- Gene001 ------->|
| exon | intron | exon |
forward strand --> ACATGGTATAAAATCGGATCGGCTATAACAG
reverse strand --> TGTACGCAATTTAGCTAGNGCTTGATATCGTC
|ncRNA|
0 10 20 30
123456789012345678901234567890123

Figure 3.1: The forward and reverse strand of sequence $S_2$ with two annotations: “Gene001” ($\alpha$) on the forward strand; and an unnamed ncRNA annotation ($\beta$) on the reverse strand.

0 10 20
123456789012345678901234567890123
I --> GTCGCCAACTGTAGAAATCCATAA
**********
II --> ACATGGTATAAAATCGGATCGGCTATAACAG
123456789012345678901234567890123
0 10 20 30

Figure 3.2: DNA sequences named “I” (top) and “II” (bottom). The similarity defined by annotation $\gamma$ is identified with stars. Note that the mismatch between base 14 of I and base 10 of II is intentional.

The final annotation in $A_2$ is $\gamma$. It is an annotation of type “Similarity” and it defines a similarity between sequences $S_2$ and $S_1$. The region $\langle 7, 16 \rangle$ specifies a part of $S_2$. This part is in some way similar to the “Target” region $\langle 11, 20 \rangle$ on the sequence with name “I”. The similarity, which in this case indicates almost identical (there is one mismatch) subsequences is visualized in figure 3.2 by aligning the two similar regions and marking them with stars (‘*’).

There can also be similar regions that are not explicitly formulated in annotations. For some of these similar regions it may be possible to detect them by comparing the visualizations of two (parts of) datasets. There can be, for example, parts of datasets where groups of annotations have the same, or similar, structures (or relative positions). Such structures may be precomputed and represented by “Similarity” annotations, but if it is unknown what kind of structures can be expected to be interesting, that is not possible.

3.1.5 Data size

Sections 3.1.1 and 3.1.2 defined the type of the input data, and this section deals with the size datasets may have. Because of the nature of DNA sequences, before giving requirements on its visualization in section 3.2, requirements are given that indicate the sheer sizes datasets can have:

Requirement 1 (Sequence size) Datasets with sequences of arbitrary length must be admissible.
The longest known contiguous DNA sequence is that of human chromosome 14, which is 87410661 bases long [W1]. However, there are organisms with much more DNA than human, thus it is very hard to specify an upper limit for the length of DNA sequences that suffices. Note that in the first instance DNAVis will be applied to annotated DNA sequences of Arabidopsis thaliana, of which the longest sequence (that of chromosome 1) is ‘only’ 30035521 bases long.

**Requirement 2 (Number of annotations)**  *Datasets with an arbitrary number of annotations must be admissible.*

Because not all secrets of DNA have been unraveled it is hard to specify an upper limit for the number of (meaningful) annotations on a DNA sequence. Of the datasets we had available for testing, Arabidopsis thaliana chromosome 1 had the most annotations (around 0.5 million).

**Requirement 3 (Own datasets)**  *It must be possible for the user to use his/her own datasets.*

Perhaps this seems self-evident, but there are web-based systems where only a certain set of datasets can be viewed.

### 3.2 Visualization requirements

The goal of this project is to create one or more visualizations of annotated DNA sequences that can help users to get more insight in the data. To accomplish this, certain subgoals have to be satisfied. These are described in sections 3.2.1 to 3.2.3 and with each of these goals one or more requirements are formulated.

#### 3.2.1 Annotations

The first main goal is to show the locations and internal structures of annotations, which are both defined by the annotation’s regions. Not only the absolute start and end positions of the regions are important, but also (and mainly) the positions and structures of multiple annotations with respect to each other.

Currently, research on miRNA (see section 2.5) is being done in Wageningen and it turns out that the miRNA’s position with respect to genes is interesting. So it is important to see if, for example, a miRNA annotation is positioned before, after, or (partly) inside a gene. More general, it is important to see how annotations of certain types are positioned with regard to other annotations of the same and different types and if, for example, certain annotations overlap. Not only the relative locations of the annotations that are being shown are important, but also their internal structures and how they relate to each other. Because a lot is still unknown about the contents of the genome and many different programs predict many different features, it cannot always be predicted what kind of structures are interesting. Thus, visualizing the annotations is essential in order to gain more insight in (the meanings of) their locations and structures.

The need to view annotations gives rise to the following two requirements:
**Requirement 4 (Annotation attributes)** For a dataset it must be possible to show the attributes of each annotation. The regions and type of the annotations are the most important attributes.

**Requirement 5 (Multiple annotations)** If there is a set of annotations that are near each other it must be possible to see how their locations and internal structures relate to each other. Annotations are near each other if their regions overlap or there are at most \( k \) bases between them, where \( k \in \mathbb{N} \) may vary depending on the needs of the user.

If the regions of different annotations are further apart, it is less probable that their relative locations and structures are important, thus requirement 5 does not state that these must be shown for annotations that are very far apart. Comparison of annotations that are not close to each other is discussed in section 3.2.4.

### 3.2.2 Levels of detail

On a single DNA sequence there can be tens of thousands of annotations and it can contain millions of bases (see requirements 1 and 2). Because the datasets are so large it is not possible to show (all bases of) the whole sequence and (the attributes of) all annotations at once. So an overview that does not show all details is needed. However, biologists need to be able to inspect individual annotations, relations between annotations, and even the individual bases of interesting parts of the sequence [9]. This means that multiple levels of detail are needed, which is the second main goal.

Except with the lowest level of detail, which shows an overview of the whole dataset, only a part of a large dataset can be shown at once. Such a part is denoted by a subdomain. A subdomain is similar to a region in the sense that it specifies two locations on a DNA sequence, but it differs because a region specifies a subsequence of a DNA sequence to which an annotation applies, and a subdomain denotes a part of a dataset by specifying a subsequence and the annotations on that subsequence. A subdomain \((sb, se) \in \text{Regions}(S_1)\) of dataset \(D_1 = (N_1, S_1, A_1)\) specifies subsequence \(S_2\) and annotations \(A_2\) as follows:

\[
S_2 = S_1[sb] \ldots S_1[se] \quad A_2 = \bigcup_{\alpha \in A_1} : (\alpha_{\min}, \alpha_{\max}) = \text{region}(\alpha) : \alpha_{\min} \leq se \wedge \alpha_{\max} \geq sb
\]

Levels of detail that show one or more whole annotations are already induced by requirements 4 and 5. For the levels of detail that shows individual bases of a part of a sequence we have the following requirement:

**Requirement 6 (Bases)** It must be possible to view the individual bases of a subsequence of the whole DNA sequence.

If only a part of the dataset is shown, this can lead to disorientation or “lostness” of the user. A guideline that should be followed to avoid this is having a continuously viewable map of the “world” (a dataset) [10], which implies the following requirement:
Requirement 7 (Overview) When showing a subdomain $s$ of dataset $D$ an overview of $D$ must be visible that highlights $s$.

The overview not only functions as a map to avoid disorientation but also includes the requirement of having a level of detail that shows the whole dataset at once.

3.2.3 Navigation and flexibility

When exploring a region of a dataset, biologists may formulate new questions about what they see and subsequently need to modify the visualization to answer these questions. These visualization modifications can include panning to the left or right or changing the scale of the view, all preferably in a continuous (as opposed to using fixed presets) and real-time interactive way [9]. Being able to change settings interactively in real-time, and for suitable options in a continuous way, will also make the program more intuitive (and thus easier to use and more effective) and gives the user a better feeling of the current context of the visualization. The latter is particularly important because navigating through very large datasets may cause disorientation [10].


It is also important that the user is enabled to simplify or re-organize a scene to avoid it becoming too complex [1], which can occur because the number and type of annotations can vary enormously in different subdomains of a dataset [9].

Requirement 9 (Flexibility) The user must be enabled to interactively specify which (types of) annotations must be shown and re-organize the scene so that it is the most effective for the current task (which varies). Results of changed options must be immediately visible to the user.

Requirements 8 and 9 also require good performance of the program, because it must be possible to, for example, zoom in and out and re-organize the scene, and directly see the result without having to wait.

3.2.4 Comparison

The third main goal is that it must be possible to compare (parts of) multiple, possibly different, datasets. The purpose of this is to make it possible to visually detect recurring features such as certain base sequences or annotation structures. When viewing only one dataset and no comparisons between datasets, multiple levels of detail are required (requirements 4 to 7). When visualizing a comparison between multiple datasets these levels of detail must also be available. When comparing with a high level of detail, it must be possible to directly compare the bases of a DNA sequence. When comparing at a level of detail showing annotations and their relative positions and structures, it must
be possible to compare these. And finally it must be possible to have some kind of overview of either the comparison or the datasets that are being compared in such a way that regions that have a lot of similarities can be identified.

**Requirement 10 (Base comparison)** *It must be possible to visually compare the bases of at least two datasets with each other.*

**Requirement 11 (Annotation comparison)** *It must be possible to visually compare the annotations and their internal and relative structures of at least two datasets with each other.*

**Requirement 12 (Dataset comparison)** *It must be possible to visually compare at least two whole datasets that makes it possible for the user to identify interesting (similar) regions at a global level.*

The requirements given in this chapter are also listed in appendix A which can be used as a reference when reading the following chapters.
Chapter 4

Existing methods

As became clear in the previous chapters visualization is essential when researching aspects of (biological features represented by) annotated DNA sequences. Because of this, tools were developed to visualize biological data. Some of these tools are evaluated in this chapter.

First, section 4.1 describes a group of web-based genome browsers. Then, in section 4.2, a stand-alone program is discussed that can be used for browsing genomes. Section 4.3 describes various methods that can be used for visualizing similarities in datasets. Finally, section 4.4 gives a summary of all methods discussed in this chapter.

Note that the tools discussed may have been developed for other/more purposes than those indicated in chapter 3. Here they are mainly assessed for how well they fulfil the requirements given there. So if a tool does not fulfil many of the requirements it still can be very useful for the purpose it was designed for.

4.1 Web-based genome browsers

Most tools that are used to visualize annotated DNA sequences use a web-based approach. This approach uses a server-client model where all data is located on a server that generates images and webpages. These images and webpages can then be viewed by a client: a web-browser.

There are many of such web-based tools, and here we discuss the following:

- Ensembl Genome Browser, the result of a joint project between the European Bioinformatics institute (EMBL - EBI) and the Sanger Institute [W2];
- NCBI Map Viewer, one of many tools developed by the National Center for Biotechnology Information (NCBI) [W3];
- UCSC Genome Browser, a genome browser created by a group of individuals and the Genome Bioinformatics Group at UC Santa Cruz. [W4];
- WormBase uses the Generic Genome Browser (GGB) of the largely open source Generic Model Organism Database (GMOD) Construction Set for browsing the genome of *C. elegans* (a worm). [W5, W6].
This section mainly describes the Ensembl Genome Browser. The other web-based visualization tools mentioned here have similar features, but if there are striking differences with the Ensembl Genome Browser, we will mention them.

4.1.1 Datasets

All tools in this section were created as graphical front ends for online databases containing biological data, so they can show (parts of) large datasets and thus satisfy requirements 1 and 2. Some of them do not support visualizing a user’s own datasets (requirement 3) because they focus on showing datasets from their own databases. However if the source or another installable version of the tool is available (as is the case with the Ensembl Genome Browser and the Generic Genome Browser which are released under open-source licenses) it is possible to install the tool on a local server and use private datasets, provided they have the correct format.

4.1.2 Levels of detail

After selecting a dataset by choosing the species and chromosome, Ensembl shows a webpage with an overview of the dataset that contains some statistic information such as histograms and graphs of important information such as the number of genes in the parts of the chromosome. In this overview the user can click on the graphs to go to a webpage that shows the selected subdomain in more detail. The user can also fill out the start and end position of the exact subdomain to be shown in more detail. NCBI does not have a separate overview but starts at a detail level that shows the whole dataset. WormBase simply starts showing a subdomain of the dataset, but a very simple overview is always available for quick navigation. UCSC has no overview at all and to start browsing a dataset the exact subdomain must be specified.

Ensembl’s detailed view or contigview shows multiple detail levels at once [11]. Each detail level is shown in a part of the webpage called a panel. The top ‘Chromosome’ panel shows an overview of the whole chromosome (or dataset) and a red box highlights the subdomain that is shown in the ‘Overview’ panel (figure 4.1). The ‘Overview’ panel shows some high level features such as contigs (see section 2.2) and has a similar red box as the ‘Chromosome’ panel for marking the subdomain that is shown in the next panel: the ‘Detailed View’ panel (figure 4.2). The ‘Detailed View’ panel shows annotations and their (both internal and relative) structures and more information is shown if an annotation is clicked on. The ‘Detailed View’ panel satisfies requirements 4 and 5 (showing annotations) and the two panels above it satisfy requirement 7 (overview). The inspection of individual bases of the sequence, which is required by requirement 6 is made possible by the final panel: the ‘Basepair View’ panel shows the individual bases of the DNA sequence, their complements, and the amino acids they can form (figure 4.3).

NCBI, UCSC and WormBase also offer the detail levels that Ensembl offers, but (except for a possible overview of the whole dataset above the more detailed view) they all show only one detail level at once. Showing multiple detail levels
Figure 4.1: Part of Ensembl’s contigview webpage showing the ‘Chromosome’ and ‘Overview’ panels. The ‘Chromosome’ panel shows human chromosome 6 and the ‘Overview’ panel shows a small part of that.

Figure 4.2: Part of Ensembl’s contigview webpage showing the ‘Detailed View’ panel. It shows the range of bases 99682517 to 99782516 of human chromosome 6.
Figure 4.3: Part of Ensembl’s contigview webpage showing the ‘Basepair View’ panel. It shows a small range of 101 bases of human chromosome 6.
at once definitely helps because it gives the user a better idea of the context of the part he/she is viewing, especially when viewing only a small part of a dataset.

4.1.3 Navigation and flexibility

All panels in Ensembl’s *contigview* have a navigation bar. In the ‘Chromosome’ panel the representation of the chromosome itself acts as a navigation bar and in the other panels it is the black and white bar with the locations or base numbers shown below it. The user can click on these navigation bars and a new contigview webpage is loaded where the location that was clicked on is centered. The ‘Detailed View’ and ‘Basepair View’ panels also offer other ways of navigation. They have buttons the user can click on for moving to the left or right or to zoom in or out, upon which a new webpage is loaded. This is not ideal because the user can not see a transition from one visualization to another. For example, if the user is looking at the detailed view panel with the chromosome and overview panels above it, depending on the user’s screen resolution, it is not unlikely that the user has to scroll down to see the detailed view panel. On itself this may not be a big problem, but when navigating through a dataset, a new webpage is loaded every time a different subdomain has to be shown in the detailed view panel. This can easily cause disorientation because the user has to scroll down every time a new webpage was loaded and try to relate the new image to the one previously shown. This is already difficult because no helpful transition between the two images was shown, and having to perform extra tasks before seeing the new image makes it even harder.

It is also possible to explicitly specify the subdomain to be viewed. This may be useful for some purposes, but it is not helpful when browsing through a dataset. With respect to requirement 8, Ensembl and the other web-based tools can be used to browse a dataset interactively, but neither in real-time nor step-less (continuous). Something similar goes for the fulfilment of requirement 9: The web-based tools are all (more or less) flexible because of the options the user can set (for example for selecting which types of annotations must be shown), but the results of setting these options are not immediately visible which makes it harder to relate a new image to the previous one or to ‘play’ with the settings in order to optimize the view. Also, except for the Generic Genome Browser it is not possible to specify the order in which the different types of annotations are shown (from top to bottom or left to right).

4.1.4 Comparison

For a limited subset of the databases Ensembl offers the *syntenyview* that shows a global comparison between one chromosome of a species and the chromosomes of another species (see figure 4.4) [12]. Besides the limitation on the datasets that can be compared, this view is also limited to only one level of detail; it is not possible to zoom in on parts of the two datasets that are being compared and show more local similarities or, for example, to compare different subdomains of the same dataset to each other. This satisfies requirement 12 but
Figure 4.4: Ensembl syntenyview showing an overview of large similar regions between human chromosome 6 and all mouse chromosomes. The chromosome in the center is the human chromosome and the surrounding ones are the corresponding mouse chromosomes. The regions are clickable to open the contigview of that part of a mouse or human chromosome. On the right hand side there is a list of possible similar proteins between the two genomes in the region identified by the red rectangle in the left hand panel (currently at the top of the human chromosome).
not requirements 10 and 11. The other web-based tools have no functionality for comparing datasets.

4.2 Apollo Genome Browser

Apollo is a sequence annotation editor. It is a collaborative project between the Berkeley Drosophila Genome Project [W7] and Ensembl [W2] set up to create a tool for creating and modifying annotations on large eukaryotic (see section 2.1) genomes [W8].

4.2.1 Datasets

Datasets can be loaded from both local files or several online databases and their sizes are not restricted (requirements 1 to 3).

4.2.2 Levels of detail

Apollo’s standard view (figure 4.5) shows both annotations and computational evidence for these annotations, such as computer-generated gene predictions. It is possible to create so-called “tiers” or horizontal rows [13] and specify what evidences are shown in each tier. Depending on whether the user chooses for the collapsed or expanded view, each tier is displayed as one horizontal row with all features of that tier shown in that row, or as multiple rows so that none of the features in the tier overlap. When showing features it is possible to click on them to display more (textual) information. This view makes it possible to show the annotations in a part of the dataset and to scroll left and right and zoom in or out (see below). Apollo uses semantic zooming [1], so when zooming in more details (such as the individual bases of a DNA sequence) will be shown. Because of this, requirements 4 to 6 are fulfilled.

4.2.3 Navigation and flexibility

Under the visualization of the annotations and their evidences, a scrollbar and zoom buttons for navigation are shown. The scrollbar can be used for scrolling left and right. It has a slider that can be dragged left or right for scrolling through the dataset. This slider works fine when showing a large part (e.g. half) of the whole dataset, but it is unusable when showing only a (small) part of a (possibly big) dataset because dragging it for as little as one pixel can result in scrolling a distance that is many times larger than the range currently shown on screen. This makes it impossible to scroll precisely and to see in the visualization of the dataset what is happening. It is also possible to click on the arrows on the left and right of the scrollbar to scroll but this is also not ideal; mainly because it is not possible to set the scrolling speed. So in order to get a reasonable way of scrolling the user will have to use one of the two methods, depending on the range of the dataset that is being shown. It would be better to have one way of scrolling that always works.
Figure 4.5: Apollo’s standard view showing a range of 292459 bases of the sample dataset that comes with the program. The ruler in the center indicates the location. The parts directly above and below the ruler show annotations on the forward and reverse strand and the features drawn on a black background are supporting evidence for the annotations. At the bottom of the screen information is displayed about the feature that was selected by clicking on it.
Figure 4.6: Apollo’s synteny view showing the synteny between a part of a human and a part of a mouse chromosome. At the top and bottom two views are shown that are similar to Apollo’s normal view, and between those two views links are shown that indicate similar regions in both datasets. Information about selected features of the datasets shown at the top and bottom are displayed on the left.

Zooming cannot be done continuously, but only step-wise by clicking on the \( \times 10, \times 2, \times 5 \) and \( \times 1 \) zoom buttons below the scrollbar. When zooming in or out, no transition is shown between the old and the new image. Because of this lack of good interactive and intuitive navigation methods, Apollo does not fulfil requirement 8. As mentioned before, it is possible to configure the layout of the features that act as evidence for the annotations shown, but this is not possible for the annotations so requirement 9 is also not fulfilled.

### 4.2.4 Comparisons

Like Ensembl, Apollo has a so-called synteny viewer which makes it possible to view similarities between two datasets [13]. It has two views: one overview which is similar to Ensembl’s synteny viewer (see figure 4.4) that satisfies requirement 12, and a detailed synteny view which shows a part of two datasets - one at the top of the view and the other at the bottom - and in the middle links between similar regions (figure 4.6). This fulfils requirement 11. Unfortunately when there are a lot of similarities that must be shown, the middle part becomes too full (and messy) to be very useful. It is not possible to directly compare the
4.3 Visualizing similarities

4.3.1 Dot plots

A dot plot is a 2D visualization of a comparison of sequences. The horizontal and vertical axes represent the two sequences $S_1$ and $S_2$ that are being compared. In its simplest form it is represented by an $\text{length}(S_1) \times \text{length}(S_2)$ image in which a pixel with coordinates $(i, j)$ with $1 \leq i \leq \text{length}(S_1)$ and $1 \leq j \leq \text{length}(S_2)$, is colored black if $S_1[i] = S_2[j]$ and white otherwise. See figure 4.7 for an example dot plot. If two sequences have equal subsequences this will result in a diagonal line in the dot plot from the upper left to the lower right. Subsequences that are each other’s reverse will result in diagonal lines from the lower left to the upper right. Users may need some training before they can quickly detect interesting features in dot plots.

There are many programs for creating dot plots such as [W9, W10, W11]. In many of them it is possible to specify a window size that will be used for each pixel instead of a window of only one base. It then colors a pixel black if in the appropriate windows of the two datasets no more than a specified number of bases differ. Another option is to specify a transfer function that defines a color for the pixels depending on the number of mismatches in a window [W10].

Standard dot plots do not have functionality for displaying annotations and
comparisons are only possible on base level. Because dot plots only compare sequences on base level, they are not suitable for visualizing similarities in huge datasets.

Wattenberg [14] presents arc diagrams as a possible alternative for dot plots. An arc diagram linearly shows a string or sequence and uses arcs to visually connect regions that match exactly. However, for DNA sequences arc diagrams are not very suitable because (1) long exact repetitions are not common because of mutations, (2) DNA sequences are very long which will result in unclear images, and (3) with arc diagrams only repetitions in one sequence are shown while it may be necessary to compare two different DNA sequences.

4.3.2 Alignment viewers

There are some tools that were created for visualizing alignments of multiple datasets. In order to align DNA sequences, they are cut at certain locations so that similar parts can be placed below (or next to) each other. Figure 4.8 shows a part of two aligned datasets visualized using SynPlot [15]. At the top it shows annotations of two datasets and below that there is a graph indicating how similar the two DNA sequences are at each location. Cuts are shown as gaps in the lines on which the annotations are drawn. Parts between two cuts are aligned to maximize the correspondence between the two datasets. SynPlot is a non-interactive tool that reads the input files and exports visual output to a (postscript) file.

The Vista Genome Browser [16] takes a similar approach in visualizing aligned sequences, but it is more interactive. The Vista Genome Browser is executed as a Java applet that shows the visualization. In this applet it is possible to select a region of the displayed aligned sequences with the mouse and this region will then be zoomed in to. It is also possible to zoom and pan using clickable buttons (both in steps, so not in a continuous way that would be needed in order to satisfy requirement 8).

Alignment viewers are better suited for visualizing similarities of large sequences than standard dot-plots, which are good for visualizing short alignments, but less adept when the length of the sequences becomes larger than the
Figure 4.9: A screenshot of PATTVision, from the PATTVision website on http://www.ece.udel.edu/praveen/pattvision/

screen resolution [16]. The Vista Genome Browser however, can show comparisons of sequences up to five million bases at once which is enough to satisfy requirement 11 (but not requirements 1 and 12).

4.3.3 3D Pattern visualization

PATTVision [17] takes another approach for visualizing similarities in biosequences. It uses a 3D visualization model in order to have more possibilities in data representation when visualizing patterns in biological sequences than when visualizing in 2D. PATTVision shows multiple (parts of) biosequences above each other and highlights specified patterns using colored planes that are orthogonal to the representation of the sequences (figure 4.9). The heights of the planes can represent additional information such as the level of support of the pattern in the dataset [17]. PATTVision does not seem suitable for huge sequences and it does not support annotations as defined in section 3.1.2 so most of the requirements of chapter 3 are not fulfilled.
As a summary of this chapter, we give an overview of the discussed visualization methods in table 4.1, which shows which of the requirements of chapter 3 each of the methods satisfy. Note that some times programs only partially fulfill requirements and a +, a □ or a – was put in the table depending on the degree to which a requirement was satisfied. Explanations of those partially fulfilled requirements were given in the preceding sections of this chapter. Of course, the table does not give all information about the purpose and functionality of each program.

Concluding; there is no single tool that fulfils all requirements given in chapter 3. The web-based genome browsers that can be installed on a local server such that the user can use his/her own datasets fulfil requirements 1 to 7 and can be used for browsing datasets. Unfortunately they lack some real-time interaction and do not offer comprehensive methods for comparing datasets. Apollo was created as a tool for creating and modifying annotations, but it performs less well as a browser for datasets. There are also tools that can be used for visualizing similarities between datasets, but none of them can do this with multiple detail levels and they are not suitable for browsing of datasets. So, what is needed is a tool that incorporates both browsing and comparing functionality, that is flexible, and can be used in a real-time and interactive way.

### Table 4.1: An overview of the requirements fulfilled by various existing methods.

<table>
<thead>
<tr>
<th></th>
<th>datasets</th>
<th>levels of detail</th>
<th>interaction</th>
<th>comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>A web-based</td>
<td>A</td>
<td>+    +    +    +</td>
<td>+</td>
<td>□</td>
</tr>
<tr>
<td>B WormBase/Generic Genome Browser</td>
<td>B</td>
<td>+    +    +    +</td>
<td>+</td>
<td>□</td>
</tr>
<tr>
<td>C NCBI Map Viewer</td>
<td>C</td>
<td>+    –    +    +</td>
<td>–</td>
<td>□</td>
</tr>
<tr>
<td>D UCSC Genome Browser</td>
<td>D</td>
<td>+    +    –    +</td>
<td>+</td>
<td>□</td>
</tr>
<tr>
<td>E Apollo Genome Browser</td>
<td>E</td>
<td>+    +    +    +</td>
<td>□</td>
<td>–</td>
</tr>
<tr>
<td>F Dot plots</td>
<td>F</td>
<td>–    –    –    –</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>G SynPlot</td>
<td>G</td>
<td>?    ?    +    –</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>H Vista Genome Browser</td>
<td>H</td>
<td>–    –    +    –</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>I PATTVision</td>
<td>I</td>
<td>–    +    –    –</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Plus (+) means ‘yes’ and minus (−) means ‘no’. □ means a requirement is partially fulfilled. A = Ensembl Genome Browser; B = WormBase/Generic Genome Browser; C = NCBI Map Viewer; D = UCSC Genome Browser; E = Apollo Genome Browser; F = Dot plots; G = SynPlot; H = Vista Genome Browser; I = PATTVision.

### 4.4 Summary

As a summary of this chapter, we give an overview of the discussed visualization methods in table 4.1, which shows which of the requirements of chapter 3 each of the methods satisfy. Note that some times programs only partially fulfill requirements and a +, a □ or a – was put in the table depending on the degree to which a requirement was satisfied. Explanations of those partially fulfilled requirements were given in the preceding sections of this chapter. Of course, the table does not give all information about the purpose and functionality of each program.

Concluding; there is no single tool that fulfils all requirements given in chapter 3. The web-based genome browsers that can be installed on a local server such that the user can use his/her own datasets fulfil requirements 1 to 7 and can be used for browsing datasets. Unfortunately they lack some real-time interaction and do not offer comprehensive methods for comparing datasets. Apollo was created as a tool for creating and modifying annotations, but it performs less well as a browser for datasets. There are also tools that can be used for visualizing similarities between datasets, but none of them can do this with multiple detail levels and they are not suitable for browsing of datasets. So, what is needed is a tool that incorporates both browsing and comparing functionality, that is flexible, and can be used in a real-time and interactive way.
Chapter 5

A new tool: DNAVis

The conclusion of chapter 4 was that a flexible tool is needed for both browsing and comparing annotated DNA sequences interactively and in real-time. This chapter describes the functionality of such a tool: DNAVis. A user manual of DNAVis is supplied with the tool itself.

Figure 5.1 shows a screenshot of DNAVis with some notes besides it to identify the different parts of the application. The largest part of the screen is taken up by the canvas, which is discussed below. The text area is used for displaying attributes of selected annotations, particularly those attributes that are not shown in the visualization. Because in this chapter we only explain what can be done with DNAVis and (in most cases) not how, the other parts of figure 5.1 are not described here. For that we refer to the user manual.

Input for DNAVis are the datasets as specified abstractly in section 3.1. Multiple datasets can be loaded simultaneously and there are no restrictions on the sizes of the datasets (except for possible hardware limitations that are not nearly reached on modern computers for currently available datasets).

The canvas contains zero or more views. Each view is associated with one or more datasets that are visualized in that view. Figure 5.1 has four views: three bar views and one matrix view. How to browse a single dataset using the bar view is described in section 5.1. In section 5.2 it is explained how datasets can be compared using the matrix view. Section 5.3 does the same using multiple bar views, and finally, in section 5.4, the whole program is assessed according to the problem specification of chapter 3.

5.1 The bar view

Each bar view visualizes one dataset, and this section describes how this is done. First, in section 5.1.1, the general approach is explained of how a subdomain of a dataset can be represented on screen. Section 5.1.2 describes how a bar view is divided into several types of bars which each contain different types of information and in sections 5.1.3 to 5.1.6 each of the bar types are described. In section 5.1.7 it is explained how perspective walls are used to integrate detailed and contextual information and in section 5.1.8 ways of interacting with a bar view are explained.
Figure 5.1: Screenshot of DNAVis at a resolution of 1280 × 1024 with four views. To the left and right of the screenshot notes were added to identify the different areas of the DNAVis window.

Figure 5.2: A bar view showing a subdomain of Arabidopsis thaliana, chromosome 1, with two annotations: At1g60820 representing a ‘tRNA’ annotation and At1g60830 representing a ‘Gene’ annotation with multiple exons. The parts between the exons are implicitly defined as introns (see section 3.1.3)
5.1.1 Linear visualization

The input data consists of a very long linear sequence with annotations on that sequence, so the most natural way of visualizing it is by displaying the sequence and annotations along a single linear axis. We call this axis the sequence axis. The sequence axis should be either horizontal or vertical. Using a diagonal line or even a curve as an axis will make it more difficult for the user to interpret the resulting image, and to form a mental model of what is being shown and the current position with respect to the whole DNA sequence. It can be argued that an axis that is not horizontal or vertical can have a larger (on-screen) length so that a greater part of the dataset can be shown. But this argument is invalid since it limits the room on an axis orthogonal to the sequence axis that is needed for showing other information such as annotations.

We chose a horizontal axis and not a vertical axis because both computer screens and the natural view of people expand more in width than in height. Also, for wide annotations it is easier to display and read text written inside those annotations if they (and the text inside them) are drawn horizontally.

5.1.2 Bars

Since we have a horizontal axis as the sequence axis, the vertical axis can be used for showing other information or to spread information that belongs to the selected subdomain. In order to do this, each bar view is partitioned into bars. A bar is a part of a view that covers the view’s full width and a certain range of the vertical part of the view. Since the bars form a partitioning of the view, all bars together fill the whole view. Figure 5.2 shows a bar view which contains multiple types of bars:

1. one overview bar;
2. one location bar;
3. one optional ruler bar;
4. a configurable amount of annotation bars.

The first three types of bars are described in section 5.1.3 and in sections 5.1.4 to 5.1.6 it is explained how the DNA sequence and the annotations on that sequence are visualized on the annotation bars.

5.1.3 Indication of the visible subdomain

If a subdomain $d$ of dataset $D$ is shown, $d$ is indicated in different ways on three different bars:

1. The location bar identifies the dataset and the exact borders of the visible subdomain by showing the following parameters as text (see figure 5.2):
   - begin base: The number of the first base in $d$.
   - dataset name: The name of dataset $D$. 

40
**location:** The number of the base in the center of the d.
**range:** The width in bases of d.
**end base:** The number of the last base in d.

Naturally, the difference between the begin and end of the subdomain equals the range.

2. The ruler bar shows ruler-like vertical lines at the bottom of the view indicating the scale of the visualization and some locations along the sequence axis. This scale indication automatically adapts to the range and is, as opposed to the indications on the location bar, linked to fixed locations in the dataset instead of locations on screen.

3. The overview bar shows an overview representing the whole dataset and a histogram is drawn in it to indicate the frequency of annotations in the respective parts of the dataset. A red line marks the current location and d is highlighted with a transparent red rectangle. The overview does not indicate the location and range of the visible subdomain as accurately as the location bar, but it gives the user a direct visual indication of d with respect to the whole dataset.

Using these three bars the user can quickly determine which subdomain is currently being shown and how the subdomain changes while zooming or panning (see section 5.1.8).

### 5.1.4 Sequence

Because we have a horizontal sequence axis, it is straightforward how the DNA sequence can be shown: by drawing the representations of the bases along the sequence axis. To accomplish this, each base was assigned its own color (we used the same colors as Ensembl, see section 4.1):

- A (adenine): green
- T (thymine): red
- C (cytosine): blue
- G (guanine): yellow
- N (unknown base): gray

Each base is now represented by a colored rectangle along the sequence axis, and if there is enough room the letters that represent the bases (A, T, C, G and N) also drawn inside the rectangles. The vertical location of the representation of the bases is defined by the annotation bar to which the sequence is assigned by the user. In figure 5.2 the bases and their complements of a subdomain of chromosome 1 of Arabidopsis thaliana are shown on annotation bar 1. Figure 5.12 on page 47 shows a DNA sequence at a smaller range, so that the letters representing the bases can also be shown. Display of complementary bases can
be switched on and off by the user. If there is less than one pixel width available for each of the bases, they cannot be shown, hence the DNA sequence can only be shown if the range of the subdomain that is being visualized is smaller than or equal to the width of the bar view in pixels.

5.1.5 Annotations

Annotations are displayed on the annotation bars of the bar view. Annotations have several important attributes, which are visualized in the following ways (see figure 5.2):

- The color, which is configurable for each type of annotation, represents the annotation’s type. In this figure, yellow represents the type “Gene” and green the type “tRNA”.
- The region of an annotation can be derived from its location on screen, though with this level of detail it may not be possible to see the exact region.
- The subdivision of an annotation in multiple regions (e.g. introns and exons of genes) is shown by using thick rectangles for defined regions (exons) and thin ones for the parts between those regions (introns). This results in inlets in the representations of annotations where introns occur.
- The name of the annotation is shown as text inside its representation.
- A transparent gray triangle drawn over the annotation represents its strand. A triangle pointing to the right indicates the annotation was defined on the forward strand. Annotations defined on the reverse strand have triangles pointing to the left and if the strand is of no importance, no triangle is drawn.

Some of these representations of attributes are only shown if there is enough room for them on screen. For example, if the text representing the name does not fit in the annotation it is not shown and if a level of detail is used such that the representation of annotation is reduced to one pixel on screen its strand and subdivision in regions are not drawn. This improves both the rendering speed and the clarity of the resulting image.

If the user clicks on an annotation, all textual information that is available for that annotation is shown in the text area (see figure 5.1). This is useful for showing the exact regions of an annotation or information that is not visualized at all (such as extra notes attached to an annotation).

5.1.6 Multiple, configurable annotation bars

Figure 5.3 shows annotations in a range of 15720 bases on one annotation bar. Despite the range is only about 0.05% of the whole DNA sequence of 3035519 bases, the image is too chaotic to be usable. To remedy this, the user can specify the number of annotation bars in the bar view and specify for each type of annotation on which annotation bar(s) it should be shown. This is a
large improvement over just one annotation bar, as can be seen in figure 5.4. Now, on annotation bar 1, one large pink annotation can be identified with the name “F9M8” that spans (at least) the whole visible subdomain. This is a “contig” annotation, which is very large (see section 2.2). Annotation bar 2 contains two simple orange annotations named “At1g41630” and “At1g41660” that represent annotations of type “pseudogene”. On annotation bar 3 five more complex yellow “Gene” annotations are shown. Finally, annotation bar 4 shows multiple small green annotations that represent annotations of type “miRNA_candidate”.

The configuration of the annotation bars is not limited to assigning one type of annotation to each of the annotation bars. The user may (for each bar view) freely set the number of bars and assign any range of annotation bars to each of the annotation types. This means that multiple types of annotations can be shown on one annotation bar and that annotations of one type can be spread over multiple annotation bars. The former can be useful to save space, especially the concerned annotations do not overlap, as is the case with genes and pseudogenes in the range of figure 5.4. The latter is useful if annotations of one type overlap, which is the case with miRNA candidates in that figure. Figure 5.5 shows an image where genes and pseudogenes are placed on one annotation bar and miRNA candidates are spread over three annotation bars. The placement of an individual annotation on an annotation bar depends on the following:

- The start bar $s$ and the number of bars $n$ to use for the annotation’s type. These two values are configurable by the user.
- The number of the annotation $i$. Annotations are incrementally numbered when the dataset the annotation is part of is loaded.

The annotation is then placed on annotation bar $s + (i \mod n)$.

If a smaller subdomain is selected, more details can be shown of some of the annotations. In figure 5.6 it can be seen that each miRNA candidate has two regions (that are approximately each other’s reverse complement, see sections 2.5 and 3.1.3) and it is visible in more detail how the locations of these miRNA candidates relate to the locations of the exons of the genes.

5.1.7 Perspective walls
The perspective wall was first implemented in 1991. Mackinlay et al. [18]:

“The perspective wall integrates detailed and contextual views to support the visualization of linearly structured information spaces.”

This is done by mapping a 2D layout onto a 3D visualization: the wall. This wall is folded so that the center part remains orthogonal to the view direction and the left and right parts of the wall are folded away and displayed with perspective so that items that are further away appear smaller.

In DNAVis perspective walls are used in order to give the user a better idea of the context of the annotations shown in the bar view. Figures 5.7 and
Figure 5.3: The annotations in a subdomain of 15720 bases of chromosome 1 of Arabidopsis thaliana. All annotations are shown on a single annotation bar.

Figure 5.4: The same annotations as in figure 5.3, but now with an annotation bar for each type of annotation.

Figure 5.5: The same annotations as in figure 5.4, but with one annotation bar more and a different distribution of the annotations over the annotation bars.

Figure 5.6: A subdomain that is included in the subdomain as shown in figures 5.3 to 5.5, but with a higher level of detail.
Figure 5.7: A bar view showing a subdomain of dataset “I”.

Figure 5.8: A bar view with perspective walls. The detailed part in the middle shows the same subdomain as figure 5.7. On the left and right part of the view perspective walls show the context of the subdomain that is shown in detail.

5.8 show two bar views; figure 5.7 has perspective walls disabled and figure 5.8 has the same settings except that perspective walls are enabled. Without perspective walls only a very limited subdomain of the dataset can be shown if a high level of detail is required and the context (annotations just outside the visible subdomain) is not visible. It is, of course, always possible to zoom out in order to show a larger subdomain, but then the level of detail can become too low for a detailed inspection of the annotations the user wishes to focus on. The use of perspective walls solves this problem. With perspective walls, one subdomain is shown in detail in the center part of the bar view, and in the left and right parts of the view, perspective walls are displayed that show nearby annotations. In the perspective walls, near annotations are shown in more detail than annotations that are further away. This results in a very natural way of viewing a dataset.

As can be seen in figure 5.8, the overview highlights the subdomains that are shown on the perspective walls, but with a less opaque color than the subdomain that is shown in detail. The highlighted part also indicates that the subdomain the perspective walls show are large, especially with respect to the amount of screen space they use. The perspective walls have the following parameters, which are configurable by the user:

- The width of the ‘middle wall’ showing the detailed subdomain;
- The angle the perspective walls make with the middle wall;
- The length of the perspective walls.
5.1.8 Interaction and navigation

As said in chapter 4, many of the currently available tools for visualizing annotated DNA sequences lack good interaction. Good interaction is one of the most important features of DNAVis; the user can both pan and zoom in an intuitive and continuous way. Each bar view has two areas with a different kind of interaction: the overview bar and the area containing all other bars.

The overview can not be used to zoom in or out, but it has two ways of changing the location:

1. The user can click (and then release the mouse button) anywhere in the overview, and the clicked location is set as the new location of the bar view. Using this feature, the user can, for example, directly jump to a location, which can be identified by the overview’s histogram, where there are very many or very few annotations.

2. The user can click on the (red) indication of the location and visible subdomain and then drag it to the left or right. This way the user can quickly scroll through large areas of the whole dataset.

Outside the overview, the user can click and drag the mouse to the left or the right in order to pan in the respective direction. Zooming is done in a similar way: by dragging the mouse up to zoom in and dragging it down to zoom out. Both panning and zooming speeds are configurable. The default panning speed is set such that if dragging the mouse, objects move with the same speed and over the same distance as the mouse cursor, resulting in a very intuitive way of panning.

DNAVis uses semantic zooming [1, 9]: as soon as there is room for certain details, these details are shown. In figures 5.4 and 5.5, it can be seen that for some annotations the names are shown and for others not. If the user would zoom in there so that the annotations will be shown bigger, eventually (as soon as there is enough room for the text in the annotation), the names of even the smallest annotations will appear. Also, both the strand indication and the visualization of introns and exons are only drawn if the user zoomed in far
Figure 5.10: The area around gene At5g45580 in chromosome 5 of Arabidopsis thaliana showing a range of 19412 bases.

Figure 5.11: Gene At5g45580 and three miRNA candidates. The colored lines at the top represent the bases of the DNA sequence and their complements.

Figure 5.12: A part of gene At5g45580 and a miRNA candidate. The bases of this part of the DNA sequence of chromosome 5 of Arabidopsis thaliana are shown at the top.
enough for these features to be distinguishable. This results in both a clearer image and a better performance.

Of course the advantages of good interaction and semantic zooming are not easily shown in static images, but nonetheless we included figures 5.10 to 5.12 in order to show some different levels of detail. These three images have a resolution of $807 \times 318$ pixels. With a higher resolution there is more room for showing details, so on a high screen resolution details will start to appear when larger subdomains of the dataset are being shown than on a low screen resolution.

5.2 The matrix view

Besides the bar view, DNAVis has a matrix view, which was created for visualizing a comparison between (parts of) two (possibly different) datasets. In order to do this, the matrix view has two sequence axes: one horizontal and one vertical. Figure 5.13 shows a matrix view comparing parts of $D_h = \text{Arabidopsis thaliana chromosome } 2$, and $D_v = \text{Arabidopsis thaliana chromosome } 4$. Along the horizontal and vertical sequence axes, bar views were placed to visualize subdomains of the horizontal and vertical datasets $D_h$ and $D_v$. Except for the fact that the vertical bar view was rotated 90 degrees counter-clockwise, the two bar views contained in a matrix view act as normal bar views: the user has the same configuration options and navigational possibilities at his/her disposal as with stand-alone bar views.

How the comparison between the two datasets is visualized in the comparison matrix is explained in section 5.2.1. Navigation in the matrix view is discussed in section 5.2.2 and section 5.2.3 sums up the uses of the matrix view.

5.2.1 Comparison matrix

The comparison matrix shows a comparison of the subdomains on the horizontal and vertical sequence axes, which are the same as the subdomains shown in the horizontal and vertical bar view. Inside the comparison matrix, similarities are identified with dots and rectangles (depending on the scale of the comparison matrix and the sizes of the similarities) in several colors:

- Regions where a similarity was defined by an annotation (see section 3.1.3) are colored red;

And, if the level of detail is high enough:

- Identical bases in the compared subdomains of $D_h$ and $D_v$ are colored orange. This gives the same result as a dot plot (see section 4.3.1);

- Complementary bases in the compared subdomains of $D_h$ and $D_v$ are marked in the same way as identical bases, but with a different color: green.

Because the similarities on base-level do not overlap and they are drawn slightly transparent over possible higher-level similarities, these three types of similarities can be shown in one image (see figure 5.14).
Figure 5.13: A matrix view comparing parts of chromosomes 2 and 4 of Arabidopsis thaliana. Each matrix view consists of two bar views and one comparison matrix.

Figure 5.14: A matrix view comparing two different subdomains of Arabidopsis thaliana chromosome 2 with each other. The level of detail is high enough for the bases to be shown in the bar views and to mark the identical and complementary bases (with orange and green) in the comparison matrix.
5.2.2 Navigation

Navigating the two datasets and their comparison can be done in the following ways:

1. The bar views can be used for navigating the comparison matrix because the subdomains of the horizontal and vertical sequence axes are linked to the subdomains of the horizontal and vertical bar views. If the user pans or zooms in one of the bar views, the subdomain of the matching axis of the comparison matrix is also adapted.

   The opposite is also valid: when navigating the comparison matrix using the two ways that are described below, the subdomains of the bar views are automatically adapted.

2. Panning along the two sequence axes can be done by left-clicking anywhere inside the comparison matrix and dragging the mouse. The comparison shown in the matrix will follow the mouse cursor. This means that in order to pan the subdomain of $D_h$ the mouse should be dragged horizontally, and to pan the subdomain of $D_v$ the mouse should be dragged vertically. It is also possible to pan along the horizontal and vertical sequence axes at once by dragging the mouse diagonally.

3. Zooming is done by right-clicking inside the area of the comparison matrix and then dragging the mouse. Dragging left and right will zoom in and out on $D_v$ and dragging up and down will zoom in and out on $D_h$. As with panning, it is possible to zoom on the horizontal and vertical axes at the same time by dragging the mouse diagonally.

5.2.3 Conclusion

The matrix view can be used for visualizing a precalculated and, on a high level of detail, base-wise comparison of two (possibly different) datasets. The user can interactively browse through the comparison and change the level of detail in a continuous (step-less) way. However, the comparison view has some limitations, which are described in the section 5.3.

5.3 Multiple views

This section describes why it is useful to have multiple views and how these work in DNAVis. Section 5.3.1 gives some limitations DNAVis would have if its canvas would not support multiple views. Section 5.3.2 explains the behaviour of views, which may be linked to other views, and section 5.3.3 describes how the limitations of section 5.3.1 are solved by having multiple views.

5.3.1 Single view limitations

If the user could only choose between having either one bar view or one matrix view on the DNAVis canvas, DNAVis would have some severe limitations:
1. At most two (parts of) datasets can be compared with each other at once;
2. The comparison matrix only shows precalculated similarity annotations and a comparison on base-level. Thus it is very hard to use it for finding certain similarities that are not stored in annotations such as certain recurring structures of (non-similarity) annotations;
3. The dataset(s) is (are) only shown at one detail level at a time, while the user may want to see several levels of detail at once.

In order to solve these problems, the DNAVis canvas can contain multiple views (see figure 5.1) which the user can add and remove.

5.3.2 View behaviour

As long as different views are not linked (see below), they are totally independent:

1. Each view can visualize a different dataset or comparison of datasets;
2. Each view shows one or two (depending on the type of view) subdomains independent of the subdomains shown in the other views, so the location and level of detail can differ per view;
3. Each view has its own settings. Thus the number of annotation bars, the settings for perspective walls, the distribution of annotations and the DNA sequence over the annotation bars, etc. can differ per (bar) view.

If the canvas contains at least one view, there is always one view that is selected. By selecting a view the options for that view become available in the view-specific options area of the DNAVis window (see figure 5.1) and it can be moved up or down in order to change the order in which the views are arranged on the canvas. The user can also link and unlink views. Thus, the views of the canvas can be partitioned into three categories of views:

**Selected view:** The view of which the options are available and which can be directly manipulated by zooming and panning. The user can select a view by left-clicking inside its area on the screen; this deselects the previously selected view. The selected view is identified by its blue background. Selecting a matrix view is done by clicking its comparison matrix. Clicking on one of the bar views contained in a matrix view will select only that bar view.

**Linked views:** Each view is contained in a single group of views that are linked: a link group (which may contain only one view). Linked views are the views in the link group that contains the selected view, excluding the selected view itself. Link groups are modified by adding and removing views to and from the link group that contains the selected view. This is done by right-clicking inside the area of views that are not selected. If that view was not already linked to the selected view, it is added to
the selected view’s link group and removed from the link group it was in. If it was linked to the selected view, it is removed from the selected view’s link group and placed in its own (new) link group. In order to identify the selected view’s link group, linked views have a green background. Of course, if the user selects a different view, the current link group may change which is immediately reflected on the screen by updating the background colors of all views.

**Other views:** All views that are not selected or linked (to the selected view) fall into this category and have a yellow background. This category can include views that are in a link group with other views, as long as these other views are not selected.

If the user pans or zooms in the selected view, the linked views follow this behaviour. If the user pans $n$ bases to the left or right in the selected view, the linked views also pan $n$ bases in the same direction, independent of the current location and level of detail those other views have. Something similar goes for zooming in and out: If the user zooms in or out such that the range of the selected view becomes $x$ times that range, the ranges of the linked views are also multiplied by $x$.

### 5.3.3 Conclusion

By having multiple views that can be linked, the three limitations mentioned in section 5.3.1 have been solved:

1. It is possible to show more than two datasets at once and to compare them. The range of the selected view can be applied to all linked views by clicking one button. This is a handy feature because it is easier to compare parts of datasets if they are all shown with the same level of detail (see figure 5.15);

2. Because the annotation structures and DNA sequences of parts of datasets can be shown directly under each other, the user can look for similarities that were not predefined by similarity annotations;

3. By linking multiple views that show the same dataset but with different levels of detail and copying the location of the selected view to the linked views, the user can see several levels of detail at once and can immediately see how they are related (see figure 5.16).

### 5.4 Evaluation

This section evaluates DNAVis. First, in section 5.4.1, it is evaluated how the requirements for DNAVis are fulfilled, and then section 5.4.2 gives the actual responses of users and a description of current projects that use DNAVis.
Figure 5.15: Three linked views that display subdomains of the same size of different datasets.

Figure 5.16: Three views that show the same location in a dataset, but each with a different level of detail.
5.4.1 Adherence to requirements

Here we list all requirements given in chapter 3 and explain why DNAVis fulfils each of the requirements. For a full description of each of the requirements, see appendix A.

1. **(Sequence size)** DNAVis does not limit the number of bases in a DNA sequence.

2. **(Number of annotations)** DNAVis does not limit the number of annotations in a dataset.

3. **(Own datasets)** Users can specify their own datasets for use in DNAVis.

4. **(Annotation attributes)** In the bar view the user can directly see the most important attributes of annotations (see section 5.1.5). Other attributes are available as text after clicking the representation of the annotation in the bar view.

5. **(Multiple annotations)** It is possible to see the attributes of multiple annotations that are near each other in the bar view (see section 5.1.6).

6. **(Bases)** DNAVis uses semantic zooming (see section 5.1.8) and if the user zooms in far enough, the bases of the DNA sequence become visible (see section 5.1.4).

7. **(Overview)** Each bar view has an overview bar (see section 5.1.3) that highlights the currently visible subdomain. In addition to this, perspective walls (see section 5.1.7) and the possibility to show one dataset with multiple levels of detail at once (see section 5.3) give the user an even better idea of the context of the subdomain on which he/she is focusing.

8. **(Navigation)** DNAVis offers a real-time and interactive way of navigating and uses semantic zooming (see section 5.1.8).

9. **(Flexibility)** The user can interactively edit many parameters in order to reorganize the scene shown in a bar view. The result of changed values for parameters are immediately visible in the bar view.

10. **(Base comparison)** When zooming in far enough, the matrix view (see section 5.2) shows dot plots (one for identical bases and one for complementary bases) that identify similar subsequences in two datasets.

11. **(Annotation comparison)** The matrix view (see section 5.2) shows the similarities between two datasets. It is also possible to show multiple bar views below each other (and to link them) such that the annotations of these datasets can be compared (see section 5.3).

12. **(Dataset comparison)** When zooming out far enough in the matrix view (see section 5.2) an overview of the similarities between two datasets.
are shown. It is also possible to show multiple bar views below each other and zoom out far enough on each of those bar views in order to compare the overviews of the annotations in multiple datasets (see section 5.3).

5.4.2 Usefulness

Since the first prototype of DNAVis, which was limited to showing (at most) one bar view (and no matrix view), the tool has been received with enthusiasm. The main reason for this was the (real-time) interactivity and flexibility of the tool. It invited the user to look around and investigate the dataset. Later, when the matrix view and the possibility of having multiple views at once were added, the comparing abilities could be added to the list of the main advantages of DNAVis.

DNAVis is used in an ongoing project to identify all candidate miRNA precursor molecules (see section 2.5) in the Arabidopsis thaliana genome. In this project, miRNA candidates are identified as following:

1. Custom designed software divides the genome in small chunks which are subsequently analysed on the presence of a reverse complementary repeat.

2. If this repeat falls into a restrictive set of criteria based on biological knowledge of actual miRNA discovered in plants [19] it is classified as a miRNA precursor candidate.

3. Possible targets for the miRNA candidates are identified by searching for homology to a known candidate, resulting in over 12,000 genes showing similarity with a miRNA. Laboratory validation was performed on a small subset of the candidates showing miRNA presence for over 60% of the tested candidates [20].

Having generated over 2500 candidates which are able to target a significant part of the Arabidopsis genes raises the question how to get a deeper insight into this large set. One of the most urgent questions raised by the research is how and where the miRNA precursors are transcribed. Some important research questions with respect to miRNAs are:

- Are they transcribed in conjunction with other genes?
- What is the relationship between miRNAs and their target?
- How are they distributed over the genome?

The most intuitive way to get an insight into these type of questions is through visual inspection. Conventional genome browsers lack real-time interactivity and do not offer the same flexibility in the number and type of views which one can display at the same time (see chapter 4). Using DNAVis it was possible to visually inspect many miRNA candidates in relation to the genes in their direct environment and also in relation to the genes they target. Figure 5.17 depicts an example of this where Mir10733 is located directly downstream of the gene At1g41860. This suggests that the expression of At1g41860 affects the
expression of the target genes of Mir10733. Using DNAVis it was possible to generate several hypotheses in relation to the function of the newly discovered miRNAs which can then be verified.

DNAVis has proven to be an invaluable tool to generate insight into these large and complex data sets. In the near future it will be used to visualize the miRNA-target relationships with the matrix view. DNAVis will also be used in a genome sequencing project where the incompletely sequenced genomes of three bacteria will be compared to three already sequenced bacteria from the same family.
Chapter 6

Future work

The main goals of the project was to create a tool that interactively visualizes annotated DNA sequences and offers possibilities for comparing multiple datasets. To accomplish this goal, requirements were formulated in chapter 3. We succeeded in fulfilling these requirements, but for DNAVis to be an all-round usable tool for use in the visualization of genetic information some improvements and expansions are still needed. This chapter describes several possible improvements and expansions for DNAVis. Section 6.1 deals with database connectivity. Section 6.2 gives some searching abilities that can be added to DNAVis, and in section 6.3 selection and bookmarking is handled. In section 6.4 we describe some possible visualization enhancements to both the bar view and the matrix view. And finally, in section 6.5, we give a conclusion.

6.1 Database connectivity

Currently DNAVis reads datasets from files in GFF and FASTA file format (see appendix C). At Plant Research International however, they use a MySQL [W12] database for storing their sequences and annotations. In order to use their data with DNAVis, the datasets first have to be exported from the database to GFF and FASTA files. This step may be eliminated by connecting DNAVis directly to the MySQL database. Besides eliminating the need to export datasets to files first, this opens up possibilities such as having dynamic datasets in which changes such as new annotations become directly available in DNAVis. Also with a database connection it becomes possible to implement functionality for directly manipulating datasets in the core database by editing and adding annotations from DNAVis.

When implementing database connectivity for DNAVis, the following issues must be considered:

**Loading of data** Depending on the speed at which data can be acquired from the database (possibly over a network connection), the time needed to load a full dataset may be quite long, so loading the complete dataset at once may not be the best solution. However, if parts of datasets need to be loaded while visualizing data, this can result in poor real-time performance of the tool. Thus, a good buffering strategy is necessary.
**Dynamic datasets** Databases are especially useful if the data that is stored is *dynamic*. This means the data can change, even while it is being visualized with DNAVis. This is something that needs to be dealt with and it may be useful or even necessary to show an indication of which parts of a dataset are currently being updated or were recently updated.

**Editing data** If functionality is added to DNAVis that allows the user to modify data, it must be avoided that the database becomes inconsistent as a result of simultaneous editing of one database by multiple database clients (including, but probably not solely, DNAVis) at once.

### 6.2 Searching

The ability to search datasets would be a very useful addition to DNAVis. Currently, the only way for a user to find certain annotations or subsequences of a dataset is by browsing the dataset (using the bar view, see section 5.1) and to look if the feature he/she is looking for is shown. The user should be enabled to search on:

**Subsequences of a DNA sequence.** This may be done in several ways. The simplest would be to search for exact subsequences only, but it could be done as complex as matching subsequences to user-definable regular expressions. Note that sequences may not only contain the bases A, T, C and G, but also elements such as N which act as wild cards (see section 3.1.1), and the user may or may not want the search tool to consider this.

**Attributes of annotations** The attributes of annotations contain information such as the type and source of the annotation. The `properties` attribute can contain extra information such as notes describing the annotation’s function or relation to other annotations. Information such as processes a gene is involved in is also available and could be made searchable.

**Multiple arguments** Combinations of multiple arguments can be useful. For example, the user may want to search for annotations of type “Gene” that contain the subsequence `CCGCATCGAACAGTTAGCA` and have the substring “sucrose” in their notes.

Results of a search query may be returned in several ways:

1. As a (textual) list of results. Of course each result must be linked to its location in the dataset so that the user can quickly go to that location in a bar view.

2. As a visual indication of where the results can be found in a bar view. This may be done, for example, by marking the locations of results in the overview or by highlighting the found subdomains or representations of annotations of the dataset themselves.

Of course, a combination of these two methods is also possible.
6.3 Selection and bookmarks

Because datasets are so vast, user may have problems retrieving interesting features or locations they were examining before. This can be avoided by enabling the user to select annotations and bookmark locations and to store these selections and bookmarks. Storing may be done by exporting selections and bookmarks to files or directly to a database DNAVis may use (see section 6.1).

When selecting annotations it should be possible to select more than one annotation at once. For example, groups of nearby annotations or the results of a search (see section 6.2) should be selectable, and thus exportable. Having multiple selections at once and the ability to add comments to both selections and bookmarks can also be useful.

If selections and bookmarked locations are stored in files, users can share these files with other users so they can have a look at them by importing them into DNAVis. If bookmarks and selections are stored in a database, it may be possible to give each user his/her own set of bookmarks and selections. Sharing of bookmarks and selections can then be made an option.

6.4 Visualization

This section describes possible expansions and improvements to the visualizations of datasets in DNAVis. In section 6.4.1 possible extensions to the bar view are discussed and section 6.4.2 gives possible extensions to the matrix view. Of course, it is also possible to create new views that use completely different visualizations. But currently it is not clear how other visualizations of annotated DNA sequences could look like.

6.4.1 Bar view

This section describes some possible improvements and expansions of the bar view.

Distribution of annotations over bars Currently, the distribution of the annotations over the annotation bars depends on the numbering of the annotations when a dataset is loaded and the assignment of annotation types to bars by the user (see section 5.1.6). Other distributions may be thought of that are better. Note that the following must be kept in mind:

- It must be visible which annotations overlap and how;
- If multiple annotations are connected, it must be visible where each of the connected annotations begin and end.

Possibilities of visualizing overlapping or connected annotations without distributing them over multiple bars may also be researched.
More information in the overview  The histogram in the overview can be expanded to show information on the different types of annotations. Currently it only indicates the frequency of annotations of any kind, but it can be expanded such that the frequencies for each individual type of annotation can be distinguished. It may also be made an option which types of annotations must be taken into account when generating the histogram.

Amino acids  The genetic code (see table 2.1 on page 11) defines the correspondence of triplets of nucleic acids in RNA to the amino acids in a protein. Because RNA is transcribed from DNA (see section 2.3), each triplet of nucleic acids in DNA also corresponds to an amino acid. These amino acids may be shown in a similar way as the DNA sequence itself in the bar view. See figure 4.3 on page 28 for an example of how this is done in the Ensembl Genome Browser.

GC base frequency  For a biologist it is interesting to see the GC base frequency in parts of a dataset. Thus, it is useful to show a GC base frequency graph of the whole dataset in the overview or of the visible subdomain on the annotation bars. The latter could be used to show information of the DNA sequence if the level of detail becomes too low to show the individual bases of the sequence. Besides the GC base frequency there are other frequencies and densities that can be interesting.

Aligning sequences  In alignment viewers (see section 4.3.2) DNA sequences are visualized linearly and cut at certain locations so that similar parts of multiple datasets can be shown below each other. It may be interesting to research the use of alignments for DNAVis.

6.4.2 Matrix view

As with the bar view, the matrix view can also have some features added, of which four are described below. The first two features are already available in the bar view, but they can also be applied to the matrix view.

Comparison overview  Like the linear dataset overview in a bar view, the matrix view could benefit from an overview of the comparison in 2D that highlights the visible subdomains that are being compared. There is room for such an overview in the upper-left corner of the matrix view (see figure 5.13 on page 49).

Multiple types of similarities  Multiple types of similarity annotations could be shown, each with its own settings (color, visibility), as with ‘normal’ annotations in the bar view. The annotations in the matrix view could also be made clickable to show extra information if there is any (which is currently not the case but it may be added in the future).
Linking of only the ranges of the sequence axes  The ranges of the horizontal and vertical sequence axis can be linked. This gives a similar advantage as having two bar views with the same range linked so that they can easily be compared. However, the user may want to link only the ranges of the two bar views without linking their locations. This can be useful if the user wants the ranges on the two sequence axes to remain equal when changing one of them, while still being able to pan on one sequence axis without the other one following.

Advanced dot plot  The dot plot of the comparison matrix may be expanded to use a sliding window where the user can define a transfer function that defines the color in the comparison matrix depending on the number of matching bases in a window of the specified size. This will make it possible to use dot plot-like features even if the range is too large to show the direct comparison on base level. In order to see a possible result of this feature, see [W10].

6.5  Conclusion

This chapter listed some possible improvements to DNAVis that can be added in the future. Many of the features listed here are already available in existing tools, but DNAVis can still be considered as a step forward in visualizing annotated DNA sequences. The project that resulted in DNAVis focused on good real-time interaction and the ability to compare datasets and on these two areas DNAVis surpasses existing tools. It has better interaction than existing tools and, partly as a result of the good interaction, gives the user a better ‘feeling’ for the context of the currently visible subdomain of a large dataset. This is very important because the sizes of datasets combined with the limited space available on a computer screen results in relative small subdomains of datasets to be visible at once which can cause disorientation of the user.

With respect to the comparison of datasets DNAVis also performs better than existing tools:

- No existing tool offers the possibility to visualize user-defined similarities as is possible with the matrix view;
- The existing tools that can show a comparison of (parts of) datasets do this in a limited way: the comparisons are not interactive and there is no support for multiple levels of detail.
- Comparing datasets in DNAVis is not limited to comparing only two datasets; multiple views can be shown and linked in order to make a comparison between more datasets possible.

So, as described in the preceding sections of this chapter, DNAVis still has its limitations, but with respect to interactive visualization and comparison of datasets it is a useful improvement over existing tools.
Appendix A

Requirements

This appendix lists the requirements that were given in chapter 3. For an extensive explanation of the requirements, see that chapter.

1. (Sequence size) Datasets with sequences of arbitrary length must be admissible.

2. (Number of annotations) Datasets with an arbitrary number of annotations must be admissible.

3. (Own datasets) It must be possible for the user to use his/her own datasets.

4. (Annotation attributes) For a dataset it must be possible to show the attributes of each annotation. The regions and type of the annotations are the most important attributes.

5. (Multiple annotations) If there is a set of annotations that are near each other it must be possible to see how their locations and internal structures relate to each other. Annotations are near each other if their regions overlap or there are at most $k$ bases between them, where $k \in \mathbb{N}$ may vary depending on the needs of the user.

6. (Bases) It must be possible to view the individual bases of a subsequence of the whole DNA sequence.

7. (Overview) When showing a subdomain $s$ of dataset $D$, an overview of $D$ must be visible that highlights $s$.


9. (Flexibility) The user must be enabled to interactively specify which (types of) annotations must be shown and re-organize the scene so that it is the most effective for the current task (which varies). Results of changed options must be immediately visible to the user.
10. **(Base comparison)** It must be possible to visually compare the bases of at least two datasets with each other.

11. **(Annotation comparison)** It must be possible to visually compare the annotations and their internal and relative structures of at least two datasets with each other.

12. **(Dataset comparison)** It must be possible to visually compare at least two whole datasets that makes it possible for the user to identify interesting (similar) regions at a global level.
Appendix B

Implementation

This appendix describes the platform on which DNAVis was developed (section B.1) and explains the object-oriented design of the tool (section B.2). These are both useful for programmers that create extensions for DNAVis.

B.1 Platform

DNAVis was developed on two different machines running Linux. Table B.1 lists the hardware and operating systems of these machines. DNAVis was programmed in C++ and its source was compiled with the GNU C Compiler (GCC) [W13] on multiple Linux distributions (see table B.1) in order to create runnable binary files. Using Cygwin [W14], DNAVis was also successfully compiled and tested on computers running Microsoft Windows.

For the implementation of the graphical user interface (GUI) we used FLTK (Fast Light Toolkit, [W15]). The main advantages of FLTK are the facts that it is lightweight, it supports 3D graphics via OpenGL, and it is very portable (no code needed to be changed for DNAVis to compile on Cygwin rather than real Linux). FLTK comes with a user-interface builder called FLUID which was used to create the GUI parts of DNAVis outside the canvas.

For the graphics on the DNAVis canvas we used OpenGL [W16] with GLU (OpenGL Utility Library) and GLUT (OpenGL Utility Toolkit). GLU comes

<table>
<thead>
<tr>
<th></th>
<th>PC 1</th>
<th>PC 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Pentium 4</td>
<td>AMD Athlon</td>
</tr>
<tr>
<td></td>
<td>2.40 GHz</td>
<td>1.0 GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>256 MB DDR</td>
<td>512 MB SDRAM</td>
</tr>
<tr>
<td>GPU</td>
<td>NVidia GeForce 4 MX 440</td>
<td>NVidia GeForce 2 MX</td>
</tr>
<tr>
<td>Operating System</td>
<td>Red Hat Linux 8.0</td>
<td>Slackware Linux 8.0,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>later 8.1 and 9.0</td>
</tr>
<tr>
<td>Compiler</td>
<td>GCC 3.2</td>
<td>GCC 2.95.3,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>later 3.2.2</td>
</tr>
<tr>
<td>FLTK version</td>
<td>1.1.2</td>
<td>1.1.3</td>
</tr>
</tbody>
</table>

Table B.1: Hardware and software used for the development of DNAVis.
with OpenGL and both GLU and GLUT are platform independent.

The source code of DNAVis is spread over 78 C++ files, of which 39 are header files. Each of these header files specifies one of the 39 classes of DNAVis, of which the most important ones are discussed in section B.2. All source files together contain 11271 lines of text, of which approximately a third are comment lines and the rest are actual lines of code. The DNAVis source tree (source only, thus without compiled files) takes up about 780 kilobytes of disk space.

B.2 System architecture

This section describes the various C++ classes that were created for DNAVis. It contains some figures that show classes, their relations, variables, and functions in UML (Unified Modeling Language) class diagrams. Note that these class diagrams are not complete because that would make them too complicated. Many of the functions and variables that classes contain are left out to increase the clarity of the figures. All object variables are private, but many of them have get and set functions that can be used to inspect or change them. For example: the range variable of instances of DNAVisSubCanvas (see section B.2.3) can be inspected and changed with the functions getRange(): double and setRange(newrange: double): void.

Section B.2.1 describes Observers and Observables. These classes are described first because many of the other classes subclass them. The way DNAVis handles datasets is explained in section B.2.2. Finally, in section B.2.3, the implementation of views in DNAVis is discussed. For a more detailed description of all classes and functions, see the HTML documentation (generated with Doxygen [W17]) that comes with the DNAVis source.

### B.2.1 Observers

Many of the objects which are described in the following sections need to monitor or observe other objects for various reasons. Because of this, the Observer and Observable classes were introduced (see figure B.1).

![Figure B.1: UML class diagram with the Observer and Observable classes.](image-url)

Classes that must be able to monitor objects must implement the Observer interface. An Observer has a notify() function that is called by the observed objects if an interesting change occurred. notify() has the following parameters:

- **observable** A pointer to the object that notified the event;
- **event** An integer value to identify the type of event that occurred;
argument A float value that further specifies the event that occurred. This parameter is optional and depends on the type of the event.

Classes that can be observed must subclass the Observable class. This class has functions addObserver() and removeObserver() for adding and removing an observer to an observable object. The function notifyObservers() is available for the subclasses of Observable and notifies the observing objects. If this function is called it calls the Observer’s notify() function and passes on the event and (optional) argument parameters supplied to notifyObservers().

B.2.2 Data

This section explains how the datasets as specified in section 3.1 are handled. First, the class for representing datasets is handled and then it is explained how datasets are read from files.

Datasets

Datasets are represented by instances of the AnnotatedDNA class (see figure B.2). An AnnotatedDNA object representing dataset $D = (N, S, A)$ holds:

- a string name identifying the name $N$ of $D$;
- a string sequence that represents the DNA sequence $S$;
- a set of annotations $A$ on DNA sequence $S$;
- a set of similarities between $D$ and other datasets $D_1 \ldots D_N$ ($N > 0$) which may include $D$. 
Similarities are stored separately from other annotations for performance reasons: when drawing a comparison matrix only the similarities are needed and when drawing a bar view only the ‘normal’ (non-similarity) annotations are needed. AnnotatedDNA has functions that can be used for obtaining the following data:

1. Subsequences of DNA sequence $S$;
2. Annotations in a specified subdomain of $D$;
3. Similarities between two subdomains of datasets, of which at least one is a subdomain of $D$.

Annotations are represented by Annotation objects which have all the attributes described in section 3.1.2. Because a similarity is a special kind of annotation, the Similarity class is a subclass of Annotation.

AnnotatedDNA has a subclass AnnotatedDNASelection that can handle the selection (and deselection) of annotations in the dataset. Currently it is mainly used for selecting the annotation(s) on which the user clicks and getting the attributes of the clicked annotation(s) to show them in the text area of the DNAVis window (figure 5.1 on page 39). But if a more advanced selection (see section 6.3) would be implemented in the user interface, the functionality is already available in the AnnotatedDNASelection class. AnnotatedDNASelection is a subclass of Observable and it notifies its observers if the current selection is updated.

Loading of data

Reading of data from files is done by subclasses of the FileReader class (see figure B.3). FileReader implements basic functionality for opening and reading text files. It has three subclasses which are involved in the reading of datasets:

1. FAReader reads a DNA sequence from a FASTA (see appendix C.1) file;
2. GFFReader reads annotations from GFF (see appendix C.2) files;
3. **AnnotatedDNAReader** opens a special file called *datasets* which contains a list of datasets and lists the FASTA and GFF files that contain these datasets (see appendix C.3). These datasets can then be requested to the *AnnotatedDNAReader*, which uses *FAReader* and *GFFReader* objects to read the DNA sequences and annotations for these datasets. Datasets are returned as *AnnotatedDNA* objects.

Because DNAVis can have multiple datasets loaded at once, the following needs to be kept track of:

- Which datasets are available (on disk)?
- Which datasets have been loaded (into memory)?
- Which (loaded) datasets are being used (for visualization in a view)?

This is handled by a **DatasetHandler** object, which also notifies the relevant parts of the GUI when (un)loading of a dataset starts or finishes and when a dataset become free, which means that it is not being used and thus the user must be given the option to unload it in order to free memory.

**B.2.3 Views and canvasses**

As described in chapter 5, DNAVis has a canvas that can contain multiple views which each visualize a dataset or a comparison between two datasets. Currently there are two types of views: the bar view and the matrix view. These are implemented by the **BarView** and **MatrixView** classes which are both subclasses of the **View** class (see figure B.4). The DNAVis canvas is implemented by **DNAViewCanvas** which has functions for:

1. Adding views to the canvas;
2. Selecting a view;
3. Moving the selected view up and down in order to change the order of the views on the canvas;
4. Deleting the selected view;
5. Linking and unlinking a view to the selected view.

Views can also contain other views. This is currently the case for the **MatrixView** which contains two **BarViews**.

**DNAViewCanvas** is a subclass of **FLGLMetaCanvas** that adds view functionality. **FLGLMetaCanvas** (see figure B.5) implements a metacanvas that can have multiple subcanvasses.

**Definition 1 (Metacanvas)** A metacanvas is a canvas that can contain subcanvasses.

**Definition 2 (Subcanvas)** A subcanvas represents a rectangular area of a metacanvas which shows its own visualization and has its own user interaction.
Figure B.4: UML class diagram with view-related classes.
Subcanvasses are represented by subclasses of GLSubCanvas. Currently there are three non-abstract subcanvas classes:

1. The **OverviewSubCanvas** renders the dataset overview of the bar view and handles user interaction in the overview bar (see section 5.1.2) of the bar view;

2. The **BarSubCanvas** renders the remaining bars of the bar view (including the annotation bars) and handles the user interaction in that part of the bar view;

3. The **MatrixSubCanvas** draws the comparison matrix of the matrix view and handles the user interaction in that part of the matrix view.

**FLGLMetaCanvas** does not only provide metacanvas functionality, but also works as a link between the (OpenGL) visualizations of the subcanvasses and the (FLTK) GUI layer that implements menus, buttons, etc. When adding a new subcanvas to the **FLGLMetaCanvas** it is optional to supply a subclass of **FLSubCanvasGUI** that implements a GUI for setting options (such as enabling and disabling of perspective walls in a **BarSubCanvas**) for that subcanvas. If a subcanvas is selected that has a GUI associated with it, this GUI is shown in the view-specific options part of the DNAVis window (see figure 5.1 on page 39).
Appendix C

File formats

Here we describe the formats of the files that DNAVis uses. Section C.1 describes the FASTA file format, which is used to store DNA sequences. In section C.2 the General Feature Format is described which is used for the storage of annotations, and in section C.3 the format of a special datasets file is described, which is used to define datasets.

C.1 FASTA

The following description of the FASTA file format is an edited version of a description that can be found on several websites, including [W18].

A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol in the first column. An example sequence in FASTA format is:

>III
ACGCCGGCCAGTGAATTGTAATACGACTCATTAGGCCGGAATTGGGCCTCTTAGATGCA
TGCTGAGGCCGGCCAGTGTGATGGATATCTGCAGAATTCGCCCTTCCCTAAATCCCTAA
ACCCCTAAAATCCCTAAACCCCTAAACCCCTAAACCCCTAAATCCCTAATCCCTAAAA
CCATAATCCCTAAATCCCTAAATCCCTAAATCCCTAAATCCCTAAATCCCTAAATCCCTAAATCCCTAA
CTAGACCCTAATCTTTAGTTCTTCAGACCTAAATCCCTAAATCCCTAAATCCCTAAATCCCTAAATCCCTAA
AAATCCCTAATGCTAAATCTCTAAATCCCTAATGGCCCAAATTTTTGCTTTGATTTGTTGTTGATTTG
AGGAATGTGCTTTTTCTTCTTCTTCTCTCTGTGTGTGTGTAGATTTTGTTTGTTTGGTTTGGTTTGGTTTGGTTTGG
TAGGTTGATTTTTGCGTGGTGTTTGTGACTCATATGGTTTGATTGGAGTTTGTTTCTGG

This sequence shows the beginning of chromosome 3 of Arabidopsis thaliana. Of course, actual input file are much longer.

The supported codes representing nucleic acids in a sequence are:

<table>
<thead>
<tr>
<th>Nucleic Acid</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>--&gt; adenosine</td>
</tr>
<tr>
<td>C</td>
<td>--&gt; cytidine</td>
</tr>
<tr>
<td>G</td>
<td>--&gt; guanine</td>
</tr>
<tr>
<td>T</td>
<td>--&gt; thymidine</td>
</tr>
<tr>
<td>U</td>
<td>--&gt; uridine</td>
</tr>
<tr>
<td>M</td>
<td>--&gt; A C (amino)</td>
</tr>
<tr>
<td>S</td>
<td>--&gt; G C (strong)</td>
</tr>
<tr>
<td>W</td>
<td>--&gt; A T (weak)</td>
</tr>
<tr>
<td>B</td>
<td>--&gt; G T C</td>
</tr>
<tr>
<td>D</td>
<td>--&gt; G A T</td>
</tr>
</tbody>
</table>
Even though U is an acceptable letter it is not a “DNA base” and it does not occur in the datasets we have for testing the application. R, Y, K, M, S, W, S, W, B, D, H and V occur only very rarely (less than \(6 \cdot 10^{-5}\) percent of the bases) in our test datasets.

C.2 General Feature Format

Parts of the description of the file format below are derived from the “GFF (General Feature Format) Specifications Document” on [W19].

DNAVis uses GFF files for the definition of annotations. GFF is a text based format and each line of text represents a so-called GFF record which represents a feature. Each GFF record \(r\) contains the following fields, which are divided by tabs in a GFF line:

**seqname** The name of a DNA sequence \(S\) that \(r\) applies to;

**source** The source of \(r\). This may be, for example, the program used to predict the feature;

**feature** The feature type name, such as “Gene”;

**start** The start position of the region that \(r\) applies to in sequence \(S\); 

**end** The end position of the region that \(r\) applies to in sequence \(S\);

**score** A floating point value. What it means is not specified so it depends on the dataset. It is possible not to specify a score by using a dot (‘.’) for the value of this field;

**strand** One of ‘+’, ‘-’ or ‘.’, where ‘+’ means \(r\) applies to the forward strand, ‘-’ means \(r\) applies to the reverse strand, and ‘.’ indicates the strand is not relevant;

**frame** One of ‘0’, ‘1’, ‘2’ or ‘.’, where ‘.’ means the frame is not relevant. ‘0’ means the region is “in frame” and ‘1’ or ‘2’ mean that the second or third base in the region are the first base of a codon. This value is not used in DNAVis.

At the end of a line in a GFF file, after the fields listed above, one or more ‘attributes’ may be listed. Each attribute consists of a *tag* and one or more *values*. Multiple attributes are separated by semicolons (‘;’) and multiple values are separated by spaces. Tags are identifiers of type \([A-Za-z][A-Za-z0-9_]\)*. Values are either of the same type or free text values quoted with double quotes, where non-printing characters are represented by their C (UNIX) style representation (e.g. ‘\n’ for newlines). Two (imaginary) example GFF lines are the following:
These two lines define annotations $\beta$ and $\gamma$ of section 3.1.4 (see page 19).

DNAVis reads GFF files and translates a collection of GFF records into a collection of annotations. The dataset an annotation belongs to is derived directly from the ‘seqname’ field of the GFF record. The ‘source’, ‘feature’ and ‘strand’ fields of a GFF record relate directly to the source, type and strand of the annotation and the ‘begin’ and end ‘fields’ of the record translate into a region.

Since a GFF record has only one ‘begin’ and ‘end’ field, and an annotation can have multiple regions, multiple GFF records are used to specify an annotation with more than one regions. If this is the case, these GFF records are defined directly under each other in the GFF file and they all have at least one ‘attribute’ of which the values correspond. Annotation $\alpha$ of section 3.1.4 is specified by the following three GFF lines:

II SourceX Gene 3 29 . + . Sequence "Gene001"; Note "Example"
II SourceX exon 3 13 . + . Sequence "Gene001"
II SourceX exon 20 29 . + . Sequence "Gene001"

As is shown here, the ‘attributes’ of GFF record, which translate into properties of an annotation can be used for various purposes, including identifying multiple GFF records that define an annotation together, and specifying a dataset name and region for a similarity (the specification of annotation $\gamma$ above). See [W19] for a full specification of GFF.

### C.3 Dataset specification

As specified in section 3.1, each dataset consists of a name, a DNA sequence, and a set of annotations on that DNA sequence. In sections C.1 and C.2 the file formats for the storage of DNA sequences and annotations were described. Here we describe how these are combined into datasets.

DNAVis expects to find a file called datasets that is located in the same directory as the DNAVis executable. This datasets file specifies the datasets that can be used in DNAVis. The datasets file is a plain text file and defines one dataset on each line in the following fields:

- **name** The name of the dataset;
- **seqfile** The location and name of the FASTA file from which the DNA sequence can be read;
- **annfile** The location and name of a GFF file from which annotations can be read.

There is one ‘name’ and one ‘seqfile’ field, and there are one or more ‘annfile’ fields. Fields are separated by spaces and/or tabs. FASTA and GFF files can define DNA sequences and annotations for multiple datasets, but if a dataset is
read in DNAVis, only those DNA sequences and annotations are read of which
the dataset name equals the one specified by the `name` field. Two example
lines of the `datasets` file are:

```
II  data/II.fa  data/II.gff  data/chr2.rnaFold  data/repeat_50.gff
Mit data/Mit.fa  data/Mit.gff data/repeat_50.gff
```

As can be seen here, parts of multiple datasets may be stored in the same files,
but different parts of those files are read depending on the name of the dataset
(here ‘II’ and ‘Mit’) that is being read.
Bibliography


[W16] OpenGL, the most widely used and supported 2D and 3D graphics application programming interface (API). http://www.opengl.org/
