University of Ljubljana, Faculty of Electrical Engineering
Laboratory of Imaging Technologies

Biomedical Informatics

LABORATORY PRACTICE

Tomaž Vrtovec

2012

University of Ljubljana, Faculty of Electrical Engineering
Laboratory of Imaging Technologies

Biomedical Informatics

LABORATORY PRACTICE

Tomaž Vrtovec

2012

Kazalo

University of Ljubljana, Faculty of Electrical Engineering
Master's study programme in Electrical Enginnering (level 2)
Biomedical Engineering - Biomedical Informatics (64209) - Laboratory practice

Lab Work 1: Introduction to Matlab

Instructions

The purpose of this introductory lab work is to learn how to handle the variables and use the basic commands and function within the Matlab programming environment. Your task is to develop an algorithm for sorting random numbers according to the principle of bubble sort.

1. Generate a vector of unsorted numbers by using the function randperm(), where the input argument N represents the number of elements in the vector that can be assigned integer values between 1 and N, while the output argument is a row vector of unsorted numbers.

2. Write a function for sorting random numbers according to the principle of bubble sort:

   ```
   function oVector = bubbleSort(iVector),
   ```

   where the input argument `iVector` is the vector of unsorted numbers, while the output argument `oVector` is the vector of sorted numbers. Both vectors are of course of the same size.

3. Write a function for displaying the obtained results:

   ```
   function displayResults(iVector, iColor, iName),
   ```

   where the input argument `iVector` is an arbitrary chosen vector of numbers that you wish to display, `iColor` is the display color (e.g. 'r' for red, 'b' for blue, etc.), and `iName` is the title of the display figure. Use the function `plot()` and adapt the display by commands `axis` and `grid`, and functions `title()`, `xlabel()` and `ylabel()`.

4. Verify the bubble sort algorithm with vectors of size $N = 50$ and $N = 1000$.

Questions

The report includes handwritten (without using the computer) answers to the following questions, while the required images can be printed and enclosed in the report.

1. Enclose a picture that shows the position of each element in the vector against the value of that element before and after sorting, but for the vector of size $N = 50$.

2. Enclose a picture that shows the position of each element in the vector against the value of that element before and after sorting, but for the vector of size $N = 1000$.

University of Ljubljana, Faculty of Electrical Engineering
Master's study programme in Electrical Enginnering (level 2)
Biomedical Engineering - Biomedical Informatics (64209) - Laboratory practice

Lab Work 2: Electrocardiography (ECG)

Instructions

Electrocardiography (ECG) is a non-invasive measurement of the electircal activity of the heart over time as detected by electrodes attached to the skin surface. The graphical representation of such measurement is an electrocardiogram (also ECG), which consists of typical periodical signals that describe the heart beat and that are composed of the P wave, QRS complex (Q wave, R wave and S wave) and T wave, and of intermediate PQ and ST segments. Your task is to retrieve typical ECG data from a publicly available ECG database and determine some of the basic parameters of the heart beat by applying time-domain analysis.

1. Go to the webpage PhysioNet (http://www.physionet.org) and choose "PhysioBank" → "Signal Archives" → "ECG", and then the ECG database "European ST-T Database". After that save the chosen ECG data, namely the description file (*.hea) and the data file (*.dat). By using the program ecg2mat_eng.m, convert the saved ECG data from 212 format (*.dat) into Matlab binary format (*.mat) while setting the start observation time to 0 s and the end observation time to 5 s.

2. Load the obtained binary data into the Matlab environment by using the function `load()`. The loaded data represents a structure (e.g. `ecg`) that consists of the description of both ECG signals(`ecg.desc1` and `ecg.desc2`), both signal data (`ecg.signal1` and `ecg.signal2`) and the corresponding time (`ecg.time`) over which the signal samples were acquired. Choose the signal that represents the 4th precordial (thoracic) lead (V_4) and display it on screen.

3. Write a function for searching of the peaks of the Q wave, R wave in S wave in the QRS complex:

   ```
   function [oIdxQ, oIdxR, oIdxS] = detectWavePeaksQRS(iData),
   ```

   where the input argument `iData` represents the vector of data samples, while the output arguments `oIdxQ`, `oIdxR` and `oIdxS` represent the vectors of locations (indices) of the peaks of the Q wave, R wave and S wave in the vector of data samples. Take into account that the R wave is the maximum in the area where the values are larger than 50% of the range of ECG data, while the Q wave and the S wave represent the first minimum on the left and right side of the R wave, respecitvely. Display the obtained locations and corresponding values of the wave peaks on screen into the same figure as the ECG signal.

4. Write a function for searching of the peaks of the P and T wave:

   ```
   function [oIdxP, oIdxT] = detectWavePeakPT(iData, iIdxQ, iIdxS),
   ```

   where the input argument `iData` represents the vector of data samples, while `iIdxQ` and `iIdxS` represent the locations (indices) of the peaks of the Q wave and S wave, respectively, in the vector of data samples. The output arguments `oIdxP` and `oIdxT` represent the locations (indices) of the peaks of the P wave and Y wave, respectively, in the vector of

data samples. Take into account that the P wave occurs on the left side of the Q wave at a location that is from the Q wave distant at most for one third of the distance between the peak of the Q wave and the peak of the previous S wave. Similarly, take into account the T wave occurs on the right side of the S wave at a location that is from the S wave distant at most for one third of the distance between the peak of the S wave and the peak of the next Q wave. Display the obtained locations and corresponding values of the wave peaks on screen into the same figure as the ECG signal.

5. Write a function to determine the heart beat on the basis of a selected signal wave:

   ```
   function [oHB_avg, oHB_std, oHF_avg, oHF_std] = computeHeartBeat(iTime, iIdx),
   ```

   where the input argument `iTime` represents the vector of time values over which the data samples were acquired, while `iIdx` represents the locations (indices) of the peaks of the selected wave in the vector of data samples. The output arguments `oHB_avg` and `oHB_std` represent the mean period and the corresponding standard deviation, respectively, of the occurence of the peak of the selected wave (in seconds), while `oHF_avg` and `oHF_std` represent the mean frequency and the corresponding standard deviation, respectively, of the occurence of the peak of the selected wave (in hertzs).

Questions

The report includes handwritten (without using the computer) answers to the following questions, while the required images can be printed and enclosed in the report.

1. Set the start observation time to 0 s and the end observation time to 5 s (experiment 1), 50 s (experiment 2) and 500 s (experiment 3). For each experiment, enclose the picture of the ECG signal and the peaks of the Q, R and S wavevs in the QRS complex, and the peaks of the P and T wave.

2. For each experiment and for each wave, write down the mean frequencies and the corresponding standard deviations (in units of beats per minute) of the occurence of the peak of the wave. Which wave has the most variable frequency? What was the basis for your answer? What is the cause of the variability?

3. Why is the size of the data file (`*.dat`) exactly 5.400.000 bytes?

Additional Problems

The answers to the following questions do not need to be enclosed in the report, but should serve for a better understanding of the topic.

Enclose the picture of the selected ECG signal, displayed on a "virtual" ECG graph paper where you take into account the standardized block size. You can set the coordinate axis ratio by using the Matlab function `dascpect([dx dy 1])`, where `dx` and `dy` represent the size of the block in x and y directions, respectively. For a reasonable display, set the start observation time to 0 s and the end observation time to 5 s.

University of Ljubljana, Faculty of Electrical Engineering
Master's study programme in Electrical Enginnering (level 2)
Biomedical Engineering - Biomedical Informatics (64209) - Laboratory practice

Lab Work 3: The DICOM Standard

Instructions

The DICOM standard (digital imaging and communications in medicine) is established among the manufacturers of medical imaging equipment as well as within healthcare institutions for storage and review of medical images.

The given file imageXY.dcm contains an image, acquired by the magentic resonance (MR) imaging technique. According to the specifications of the DICOM standard, extract the selected data elements from the file. To determine the data elements you can use the group tags in the file dcmGroups.mat, the element tags in the file dcmElements.mat and the tag descriptions in the file dcmRepresentations.mat. Load the files by using the Matlab function `load()`, each of the loaded structures `S` contains a variable `S.code` with the code and a variable `S.desc` with the description.

1. Write a function for loading the DICOM file:

   ```
   function oData = loadDicomFile(iPath),
   ```

   where the input argument `iPath` represents the path (folder and filename) to the DICOM file. The output argument `oData` represents the data in the DICOM file in the form of a row vector, individually for each byte of the data. Use Matlab fucntions `fopen()`, `fread()` and `fclose()` to read the data, where you take into account the data type `'uint8'` (unsigned 8-bit integer).

2. Write a function for retrieving the data element from the data in the DICOM file:

   ```
   function oElement = getDicomDataElement(iData, iTag),
   ```

   where the input argument `iData` represents the data in the DICOM file, while `iTag` represents the tag of the data element in the form of (group,element) = (gggg,eeee). The output argument `oElement` is in the form of a structure that contains the following variables, which correspond to the data element:

   - `oElement.idx` is the index of the beginning of the data element in the vector of DICOM data, e.g. `585`.
   - `oElement.tag` is the tag of the data element in the form of (group,element) = (gggg,eeee), e.g. `'0008,0023'`.
   - `oElement.VR` is the value representation (VR) of the data element, e.g. `'DA'` (represents the date).
   - `oElement.VL` is the value length (VL) of the data element in bytes, e.g. 8.
   - `oElement.value` is the value of the data element, e.g. `'20120516'`.

   To convert numbers from decimal to hexadecimal numeral system, or vice versa, use the Matlab functions `dec2hex()` and `hex2dec()` (e.g. `dec2hex(35) = '23'` and `dec2hex(35, 4) = '0023'`; `hex2dec('23') = 35` and `hex2dec('0023') = 35`).

   To convert numbers to text strings use the Matlab function `char()` (e.g. `char(68) = 'D'` and `char([68, 65]) = 'DA'`).

3. Adjust the output argument `oElement.value` of the function `getDicomDataElement()` so that it will match the value represenation (VR) given by `oElement.VR`. For this purpose, write a function:

```
function oValue = convertDicomValue(iValue, iVR),
```

where the input argument `iValue` is the value, while `iVR` is the value representation of the data element. The output argument `oValue` represents the adequatly converted value of the data element. Consider the following value representation codes: AS, CS, DA, DS, LO, PN, SH, ST, TM, UL and US.

Example: If the value representation code is `oElement.VR = 'DA'` and the value is `oElement.value = [50, 48, 49, 50, 48, 53, 49, 54]`, then the adequate conversion is by using the Matlab function `char()` to produce the date (code DA) `oElement.value = '20120516'` in the form of YYYYMMDD.

4. Display the image by using the enclosed function `displayDicomImage(iData)`, where the input argument `iData` represents the data in the DICOM file.

Questions

The report includes handwritten (without using the computer) answers to the following questions, while the required images can be printed and enclosed in the report.

1. Write down the filename of the selected DICOM file.

2. Retrieve the data elements that are stored in the group Study Information. Write down the values for the Image Date, Modality, Manufacturer and Institution Name. For each value, write down also the location (index) of the beginning of the data element in the vector of DICOM data, the tag in the form of (gggg,eeee), the value representation (VR) and the value length (VL, in bytes).

3. Retrieve the data elements that are stored in the group Patient Information. Write down the values for the Patient's Name, Patient's Birth Date, Patient's Sex, Patient's Age and Patient's Weight. For each value, write down also the location (index) of the beginning of the data element in the vector of DICOM data, the tag in the form of (gggg,eeee), the value representation (VR) and the value length (VL, in bytes).

4. Retrieve the data elements that are stored in the group Imag Information. Write down the values for the Rows, Columns, Bits Allocated, Window Center and Window Width. For each value, write down also the location (index) of the beginning of the data element in the vector of DICOM data, the tag in the form of (gggg,eeee), the value representation (VR) and the value length (VL, in bytes).

5. Enclose the picture of the image that is stored in the selected DICOM file.

6. What is the size (in bytes) of ... :

   - ... the selected DICOM file?
   - ... the image data?
   - ... all data except the image data?
   - ... all data elements?

   Please provide a suitable explanation for the answers.

7. Write down the text that is stored between byte 129 and byte 132 in the selected DICOM file. What can you conclude on the basis of the obtained text?

Additional Problems

The answers to the following questions do not need to be enclosed in the report, but should serve for a better understanding of the topic.

Adjust the output argument `oElement` of the function `getDicomDataElement()` so that you determine also:

- `oElement.group` is the name of the tag group that can be accesses through the file dcmGroups.mat. The obtained structure determines the groups, e.g. at location 6 is `G.code{6} = '0028'` and the corresponding description is `G.desc{6} = 'Image Information'`.
- `oElement.element` is the name of the tag that can be accessed through the file dcmElements.mat. The obtained structure determines the elements, e.g. at location 96 is `E.code{96} = '0028,0100'` and the corresponding description is `E.desc{96} = 'Bits Allocated'`.

University of Ljubljana, Faculty of Electrical Engineering
Master's study programme in Electrical Enginnering (level 2)
Biomedical Engineering - Biomedical Informatics (64209) - Laboratory practice

Lab Work 4: Cryptography

Instructions

Criptography is a scientific field of the practice and study of techniques for secure communication, related to various aspects of data security, privacy and confidentiality. Basing on a simple algorithm and secret key, encrypt and decrypt the given text, and transform the results into a useable form according to the Base64 scheme encoding.

1. Write a function for the encryption of a text message according to the secret key:

   ```
   function oText = encryptText(iText, iKey),
   ```

   where the input argument `iText` represents the plaintext, and `iKey` represents the encryption key. The output argument `oText` represents the ciphertext. To encrypt, sum the numerical values of the plaintext and of the encryption key, and if necessary, repeat the the encryption key to the length of the plaintext. Then encode the ciphertext with the Base64 scheme, for which you write a function:

   ```
   function oText = encodeBase64(iText),
   ```

   where the input argument `iText` represents the initial text, while the output argument `oText` represents the encoded text.

   To determine the numerical values for text characters, use the Matlab function `unicode2native()`, while to determine the text characters for numerical values, use the Matlab function `native2unicode()`. To convert between the decimal and binary base use Matlab functions `dec2bin()` and `bin2dec()`. The numerical values and corresponding text characters for the Base64 encoding scheme can be obtained from the file tableBASE64.mat (e.g. the structure `BASE64`), where `BASE64.charCode` represents numerical values and `BASE64.charSymbol` represents the corresponding text characters (e.g. `BASE64.charCode(13) = 12` and `BASE64.charSymbol(13) = 'M'`.

2. Write a function for the decryption of a text message according to the secret key:

   ```
   function oText = decryptText(iText, iKey),
   ```

   here the input argument `iText` represents the ciphertext, and `iKey` represents the decryption key. The output argument `oText` represents the plaintext. To decrypt, subtract the numerical values of the decryption key from the numerical values of the ciphertext, where if necessary, you repeat the length of the decryption key to reach the length of the ciphertext. Before decryption, the ciphertext has to be decoded with the Base64 scheme, where the padding bytes are represented by the character '=', for which you write a function:

   ```
   function oText = decodeBase64(iText),
   ```

   where the input argument `iText` represents the initial text, while the output argument `oText` represents the decoded text.

   The procedure is reversed when compared to the encryption, therefore you can use the same Matlab function as described under 1.

3. Write a function that loads the text message from a given text file:

```
function oText = loadText(iPath),
```

where the input argument `iPath` represents the path (folder and filename) to the text file, while the output argument `oText` represents the text message in the form of a row vector.

The data can be read by using Matlab functions `fopen()`, `fread()` and `fclose()`, and by taking into account the data type 'uint8' (unsigned 8-bit integers), which can be transformed into text characters by using the Matlab function `native2unicode()`.

Questions

The report includes handwritten (without using the computer) answers to the following questions, while the required images can be printed and enclosed in the report.

1. Load the plaintext from the file plainText_eng.txt, then encrypt it with the above describe algorithm by using the string 'encryption' for the secret key, and finally encode it according to the Base64 scheme. Write down the plaintext and the obtained ciphertext.

2. Load the ciphertext from the file cihperText_eng.txt, then decode it according to the Base64 scheme, and finally decrypt it with the above described algorithm by using the string 'decryption' for the secret key. Write down the ciphertext and the obtained plaintext.

3. What is the length (in bytes) of . . . :

   - . . . the plaintext in file plainText_eng.txt?
   - . . . the encrypted and encoded plaintext?
   - . . . the ciphertext in file cipherText_eng.txt?
   - . . . the decoded and decrypted ciphertext?

   Provide adequate arguments for your answers in connection to the lenght of the initial message (plaintext or ciphertext).

4. Why is it useful to encode the results according to the Base64 scheme? Provide adequate arguments for your answer.

Additional Problems

The answers to the following questions do not need to be enclosed in the report, but should serve for a better understanding of the topic.

The file secretText_eng.txt contains a chipertext that was encrypted with the above described algorithm by using an unknown secret key. By using the brute force search (trying out all possible combinations), decrypt the secret text and find the secret key, if you know that the plaintext contains the word "Janez" and that the length of the secret key is 3 alphabet characters (lower case or upper case). To search for the string of characters within a text you can use the Matlab function `strfind()`.

University of Ljubljana, Faculty of Electrical Engineering
Master's study programme in Electrical Enginnering (level 2)
Biomedical Engineering - Biomedical Informatics (64209) - Laboratory practice

Lab Work 5: Binary Classification

Instructions

To test the hypothesis that one of the diseases of the heart and cardiovascular system is related to the sistolic (maximum) blood pressure, the researchers developed four different diagnostic tests (methods) for measuring the blood pressure. All tests can measure the blood pressure in the range between 70 mmHg and 170 mmHg (lower or higher values are not detected), and were applied to measure the blood pressure in 2268 subjects, among which one half was diseased (1134 subjects) and one half was healthy (1134 subjects). The results were stored in the file labData.mat (e.g. structure `D`), in which the variable `D.refData` represents the reference data (value of 0 represents a healthy subject, value of 1 represents a diseased subject), while the variable `D.testData` represents the results of diagnostic tests (blood pressure in mmHg), with each row `i` corresponding to a subject and each column `j = 1...4` corresponding to a diagnostic test:

	D.refData	D.testData			
i	(i,1)	(i,1)	(i,2)	(i,3)	(i,4)
⋮	⋮	⋮	⋮	⋮	⋮
71	0	85.5	98.5	114.8	115.0
72	1	144.2	70.6	113.8	115.0
73	0	93.4	145.6	117.9	115.0
⋮	⋮	⋮	⋮	⋮	⋮

1. Write a function for the binary classification of the results:

   ```
   function [oTP, oTN, oFP, oFN] = classifyData(iThreshold, iTestData, iRefData),
   ```

   where the input argument `iThreshold` represents the classification threshold, `iTestData` is the vector of the results of the diagnostic test, and `iRefData` is the vector of the reference data. Output arguments `oTP`, `oTN`, `oFP` and `oFN` represent, respectively, the number of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) results.

   Classify the data according to `tNum` different thresholds by equally distributing them between the minimal threshold `tMin` = 70 mmHg and maximal threshold `tMax` = 170 mmHg.

2. Write a function for computing the ratios that evaluate the performance of the binary classification:

   ```
   function function [oTPR, oTNR, oFPR, oFNR] = computeRates(iTP, iTN, iFP, iFN),
   ```

   where input arguments `iTP`, `iTN`, `iFP` and `iFN` represent, respectively, the number of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) results. Output arguments `oTPR`, `oTNR`, `oFPR` and `oFNR` represent, respectively, the true positive rate (TPR or sensitivity), true negative rate (TNR or specificity), false positive rate (FPR or non-specificity) and false negative rate (FNR or non-sensitivity).

3. Write a function for computing the remaining values that also evaluate the performance of the binary classification:

   ```
   function function [oPPV, oNPV, oFDR, oACC] = computeValues(iTP, iTN, iFP, iFN),
   ```

   where input arguments `iTP`, `iTN`, `iFP` and `iFN` represent, respectively, the number of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) results. Output arguments `oPPV`, `oNPV`, `oFDR` and `oACC` represent, respectively, the positive predictive value (PPV), negative predictive value (NPV), false discovery rate (FDR) and classification accuracy (ACC).

4. Write a funciton for displaying the classification results:

   ```
   function drawResults(iX, iY, iTitle, iLabelX, iLabelY),
   ```

   where input arguments `iX` and `iY` represent, respectively, the values on x and y axes of the display, `iTitle` represents the title of the display, and `iLabelX` and `iLabelY` represent, respectively, the labels on x and y axes of the display.

   The input arguments `iX` and `iY` should be matrices, where each row represents a different classification threshold and each column represents a different diagnostic test. Display the classification results for all diagnostic tests in the same coordinate system, and display the legend with generic labels (`data1`, `data2`, `data3`, `data4`) by using the Matlab function `legend('Location', 'BestOutside')`.

5. Write a function for computing the area under the curve (AUC):

   ```
   function oAUC = computeAUC(iX, iY),
   ```

   where input arguments `iX` and `iY` represent the curve in the form of values on x and y axes, respectively, of the display, while the output argument `oAUC` represents the area under the curve, computed according to the trapezoidal rule.

Questions

The report includes handwritten (without using the computer) answers to the following questions, while the required images can be printed and enclosed in the report.

1. Classify the results according to `tNum = 20` different thresholds and enclose the images showing the course of TPR, TNR, FPR and FNR against the classification threshold.

2. Classify the results according to `tNum = 20` different thresholds and enclose the images showing the course of PPV, NPV, FDR and ACC against the classification threshold.

3. Classify the results according to `tNum = 20` different thresholds and enclose the image showing the ROC for all diagnostic tests, and write down the corresponsing AUC values. Basing on the AUC values, evaluate the success of each diagnostic test.

4. According to the obtained courses and curves, determine the optimal classification threshold for each diagnostic test. Provide adequate arguments for your choices.

5. Which diagnostic test proved to perform best? What can you conclude about the test with the completely diagonal ROC? What can you conclude about the test with the almost diagonal ROC? Provide adequate arguments for your answers.

6. Set the classification threshold to 120 mmHg. For each diagnostic test, write down the contingency table, and report the TPR, TNR, FPR and FNR, as well as the PPV, NPV, FDR and ACC.

Additional Problems

The answers to the following questions do not need to be enclosed in the report, but should serve for a better understanding of the topic.

Write a function for computing the classification threshold that results in a chosen classification sensitivity:

```
function [oThreshold, oTPR, oFPR] = computeThreshold(iTPR, iTestData, iRefData),
```

where the input argument `iTPR` represents the chosen true positive rate (TPR or sensitivity), `iTestData` is the vector of the results of the diagnostic test, and `iRefData` is the vector of the reference data. The output argument `oThreshold` represents the threshold that results in the chosen classification sensitifity, `oTPR` represents the actually achieved classification sensitivity, while `oFPR` represents the actually achieved classification non-specificity.

Write down the resulting classification thresholds as well as the actually achieved classification sensitivity and non-specificity when choosing classification sensitivities of 90.0%, 95.0% and 97.5%.

University of Ljubljana, Faculty of Electrical Engineering
Master's study programme in Electrical Enginnering (level 2)
Biomedical Engineering - Biomedical Informatics (64209) - Laboratory practice

Lab Work 6: Electronic Health Record

Instructions

The electronic health record (EHR) is a concept of systematic collection of data related to health of individual patients in an electronic form. Suppose that we have an EHR encoded with the extensible markup language (XML), where the record notation is simplified (tags do not contain attributes, the name of the tag is not repeated on the same level, the value of the tag is always a text string), e.g. as:

```
<medications>
      <medication2>
            <name>Ultop</name>
            <fullname>Ultop 10 mg hard capsules</fullname>
            <usage>1 capsule each 24 hours</usage>
            <code>040762</code>
            <dateOfPrescription>
                  <dd>15</dd>
                  <mm>06</mm>
                  <yyyy>2012</yyyy>
            </dateOfPrescription>
            <dateOfExpiration>
                  <dd>06</dd>
                  <mm>07</mm>
                  <yyyy>2012</yyyy>
            </dateOfExpiration>
      </medication2>
</medications>
```

The notation in the form of a structure that is equivalent to the above mentioned XML is:

```
medications.medication2.name = 'Ultop'
medications.medication2.fullname = 'Ultop 10 mg hard capsules'
medications.medication2.usage = '1 capsule each 24 hours'
medications.medication2.code = '040762'
medications.medication2.dateOfPrescription.dd = '15'
medications.medication2.dateOfPrescription.mm = '06'
medications.medication2.dateOfPrescription.yyyy = '2012'
medications.medication2.dateOfExpiration.dd = '06'
medications.medication2.dateOfExpiration.mm = '07'
medications.medication2.dateOfExpiration.yyyy = '2012'
```

By applying XML parsing, your task is to convert the given XML record into the form of a Matlab structure, then add a chosen sub-structure to the obtained structure, and finally convert the resulting structure back into an XML record.

1. Load the XML record from the file sampleEHR_eng.xml by using Matlab functions `fopen()`, `fread()` and `fclose()`.

2. Write a function for converting an XML record into the form of a structure:

```
function oStruct = xml2struct(iXml, iStruct, iTag),
```

   where the input argument `iXml` represents the XML record, `iStruct` is the current structure, while `iTag` is a cell structure with the names of all XML tags up to the current tag. The output argument `oStruct` represents the new structure.

   The function implementation has to enable recursive calls in the case of nested XML tags, where the initial function call is `EHR = xml2struct(XML, struct, [])`. Recursion is a code programming technique, where the solution to the given problem consists of multiple solutions to individual sub-problems. In practice, the function call is executed within the function itself by using different arguments and ensuring a stop criterion.

   By using this function, convert the XML record into the form of a structure.

3. Add the sub-structure `medications.medication2`, given in the introduction above, to the obtained structure.

4. Write a function for converting a structure into the form of an XML record:

```
function oXml = struct2xml(iStruct, iXml, iOffset),
```

   where the input argument `iStruct` represents the structure, `iXml` is the current XML record, while `iOffset` is the whitespace offset of the current XML tag. The output argument `oXML` represents the new XML record.

   The function implementation has to enable recursive calls in the case of nested XML tags, where the initial function call is `XML = struct2xml(EHR, '', '')`. To move to a new line in the XML record, use the ASCII code 10, i.e. `char(10)`.

   By using this function, convert the structure into the form of an XML record.

5. Save the XML record into the file updatedEHR.xml by using Matlab functions `fopen()`, `fwrite()` and `fclose()`.

Questions

The report includes handwritten (without using the computer) answers to the following questions, while the required images can be printed and enclosed in the report.

1. List the advantages and disadvantages of having data stored in the form of an XML record. Provide adequate arguments for each advantage and disadvantage.

2. Write down a recursive version of the function for computing the factorial of the natural number N:

   ```
   function oValue = fact(iValue),
   ```

   where the input argument `iValue` is the natural number N, and the output argument `oValue` is the factorial of N, i.e. $N!$, defined as:

   $$N! \quad = \quad N \cdot (N-1) \cdot (N-2) \cdot \ldots \cdot 2 \cdot 1 \quad = \quad \prod_{i=1}^{N} i \,.$$

3. Write down an iterative version of the function from question No. 2.

Additional Problems

The answers to the following questions do not need to be enclosed in the report, but should serve for a better understanding of the topic.

Generalize functions `xml2struct()` and `struct2xml()` so that they will enable the decoding of the attribute `type` within each tag, which represents the type of the tag value, e.g.:

- tag `<id type="number">123456</id>` should represent a number, i.e. `EHR.id = 123456`,
- tag `<dd type="string">12</dd>` should represent a text string, i.e. `EHR.dd = '12'`.

In the case the attribute does not exist for a specific tag, then the tag contains a nested XML record. To validate the generalized functions, use the XML record in the file sampleEHR_attr_eng.xml.

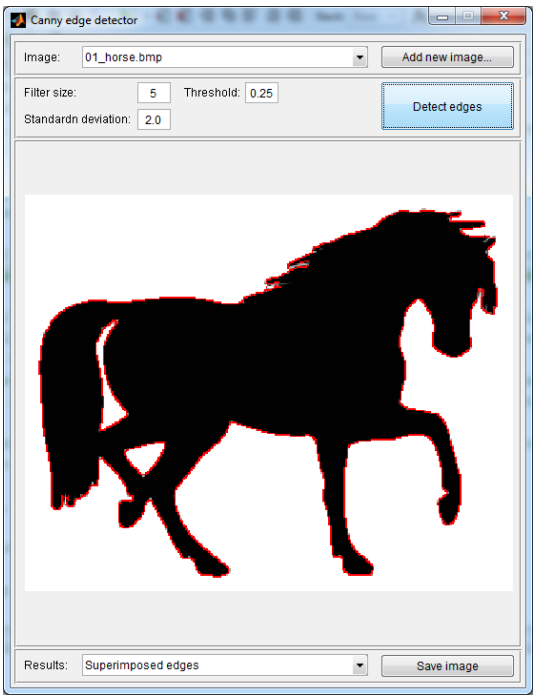
University of Ljubljana, Faculty of Electrical Engineering
Master's study programme in Electrical Enginnering (level 2)
Biomedical Engineering - Biomedical Informatics (64209) - Laboratory practice

Lab Work 7: Image Edge Detection

Instructions

Image edge detection is one of the most frequently applied concepts in the field of image processing and analysis, and the Canny edge detector is one of the most frequently applied techniques.

Your task is to implement a simplified Canny edge detector by using the given graphical user interface (GUI) that allows loading of a chosen grayscale image, selecting the edge detection parameters (size and standard deviation of the Gauss gradient filter, and the value for thresholding), and saving of the obtained results.



1. The given file getGradientFilter.m contains the declaration of the function for generating the gradient operator of the Gaussian function:

```
function [oFilterX, oFilterY] = getGradientFilter(iN, iSigma),
```

where the input argument `iN` represents the size $N = 2M + 1$, and `iSigma` represents the standard deviation σ of the gradient operator of the Gaussian function of size $N \times N$, defined by the equation:

$$\nabla N(x,y) = \nabla e^{-\frac{x^2+y^2}{2\sigma^2}} = \begin{bmatrix} -\frac{x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \\ -\frac{y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \end{bmatrix} = \begin{bmatrix} -x \\ -y \end{bmatrix} \frac{1}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} .$$

Output arguments `oFilterX` and `oFilterY` represent the gradient operators $C_x(i,j)$ and $C_y(i,j)$ of the Gaussian function in x and y direction, respectively.

Replace the blind implementation in the given function with a working implementation, so that the output arguments will actually represent the gradient operators of the Gaussian function according to the input arguments, which can be selected in the GUI.

2. The given file imageFiltering.m contains the declaration of the function for image filtering:

```
function [oMapX, oMapY] = imageFiltering(iImage, iFilterX, iFilterY),
```

where the input argument `iImage` represents the image, while `iFilterX` and `iFilterY` are the operators $C_x(i,j)$ and $C_y(i,j)$ for image filtering in x and y direction, respectively.

Output arguments `oMapX` and `oMapY` represent the filtered input image $g_x(x, y)$ and $g_y(x, y)$ in x and y direction, respectively. For each picture element (x, y) perform filtering as:

$$g(x, y) = \sum_{i=-M}^{M} \sum_{j=-M}^{M} c(i, j) f(x + i, y + j) .$$

Replace the blind implementation in the given function with a working implementation, so that the output arguments will actually perform image filtering according to the input arguments, obtained by using the function `getGradientFilter()`.

3. The given file nonMaximaSuppression.m contains the declaration of the function for suppressing non-maximal values:

```
function [oMapX, oMapY] = nonMaximaSuppression(iMapX, iMapY),
```

where input arguments `iMapX` and `iMapY` represent the image gradients $g_x(x, y)$ and $g_y(x, y)$ in x and y direction, respectively, while output arguments `oMapX` and `oMapY` represent the image gradient $g'_x(x, y)$ and $g'_y(x, y)$ in x and y direction, respectively, with suppressed non-maximal values. Perform the suppression of non-maximal values by removing a picture element if the magnitude of the gradient of at least one neighboring picture element, from the point of view of the discrete gradient, is larger than the magnitude of the gradient of the observed picture element.

Replace the blind implementation in the given function with a working implementation, so that the output arguments will actually represent the image gradient with suppressed non-maximal values according to the input arguments, obtained by using the function `imageFiltering()`.

4. The given file imageThresholding.m contains the declaration of the function for image thresholding:

```
function oMap = imageThresholding(iMapX, iMapY, iThreshold),
```

where input arguments `iMapX` and `iMapY` represent the image gradient $g'_x(x, y)$ and $g'_y(x, y)$ in x and y direction, respectively, with suppressed non-maximal values, while `iThreshold` represents the threshold T on the interval between 0 and 1. The output argument `oMap` represents the thresholded values $u(x, y)$. Perform the thresholding by assigning the value of 1 to picture elements with values above the thresold, and the value of 0 to the remaining picture elements:

$$u(x, y) = \begin{cases} 1; & \text{if } f(x, y) > T , \\ 0; & \text{if } f(x, y) \leq T . \end{cases}$$

Replace the blind implementation in the given function with a working implementation, so that the output argument will actually represent the thresholded image according to the input arguments, obtained by using the function `nonMaximaSuppression()`, while the threshold can be selected in the GUI.

Validate the implemented technique for image edge detection on images 01_horse.bmp, 02_brain.bmp and 03_vertebra.bmp. Save the results as JPEG images by adequate selection and pressing the button "Save image" in the GUI.

Questions

The report includes handwritten (without using the computer) answers to the following questions, while the required images can be printed and enclosed in the report.

1. For image 02_brain.bmp enclose the picture of the original image, the pictures of the gradient operator in x and y directions, the pictures of image gradients in x and y directions, the picture of image gradient magnitude, the picture of image gradient magnitude with suppressed non-maximal values, the picture of the thresholded image and the picture with superimposed edges. Write down the selected values for filter size and standard deviation as well as the selected value for thresholding.

2. For image 03_vertebra.bmp enclose the picture of the original image, the pictures of the gradient operator in x and y directions, the pictures of image gradients in x and y directions, the picture of image gradient magnitude, the picture of image gradient magnitude with suppressed non-maximal values, the picture of the thresholded image and the picture with superimposed edges. Write down the selected values for filter size and standard deviation, as well as the selected threshold.

3. Provide adequate arguments for you selection of values for the size and standard deviation of the filter in the form of the gradient operator of the Gaussian function, as well as for the selected threshold.

4. How was the implemented technique simplified in comparison to the "true" Canny edge detector?

Additional Problems

The answers to the following questions do not need to be enclosed in the report, but should serve for a better understanding of the topic.

Modify the function `imageThresholding()`, so that the input argument `iThreshold` the upper threshold, while the lower threshold is defined as the half of the upper threshold. Then perform the hysteresis thresholding, which retains all picture elements with the grayscale intensities above the upper threshold, while the picture elements with the grayscale intensities between the lower and upper threshold are retained only in the case when the grayscale intensities of the neighboring picture elements are above the upper threshold:

```
>>  % hysteresis image thresholding
>>  function oMap = imageThresholding(iMapX, iMapY, iThreshold)
>>      % upper and lower threshold
>>      iHighT = iThreshold;
>>      iLowT = iHighT/2;
>>      ...
```

University of Ljubljana, Faculty of Electrical Engineering
Master's study programme in Electrical Enginnering (level 2)
Biomedical Engineering - Biomedical Informatics (64209) - Laboratory practice

Lab Work 8: Sequence Alignment

Instructions

Your task is to implement the Needleman-Wunsch algorithm for the determination of the global optimal alignment of sequences (e.g. nucleotides in DNA or amino acids in proteins). The algorithm is a special case of the technique known as dynamic programming, which is used to solve relatively complex problems by dividing them into easier sub-problems.

1. Write a function for the determination of the score matrix M and trace matrix T for alignment of sequences a and b:

   ```
   function [oScrM, oTrcM] = computeMatrices(iSeqA, iSeqB, iSubS, iSubM, iGapP),
   ```

   where input arguments `iSeqA` and `iSeqB` represent sequences a and b, respectively, `iSubS` represents the symbols in the selected order (e.g. `'AGCT'`), `iSubM` represents the substitution matrix S with symbols in the same order, and `iGapP` represents the gap penalty P. Output arguments `oScrM` and `oTrcM` represent the score and trace matrices M and T, respectively.

   The score matrix M has to be initialized with zeros, while the trace matrix T has to be initialized with initial directions. The score $M(i,j)$ and trace $T(i,j)$ are then defined by the following recursive formula:

   $$M(i,j) = max \begin{cases} M(i-1,j-1) + S(a(i),b(j)) & \nwarrow \text{ (diagonal direction or D)} \\ M(i,j-1) + P & \leftarrow \text{ (left direction or L)} \\ M(i-1,j) + P & \uparrow \text{ (upward direction or U)} \end{cases}$$

   In the case that multiple values are simultaneously maximal, the trace matrix T is assigned only one direction by taking into account the pre-defined order, i.e. first the diagonal direction ($\nwarrow$ or D), then the left direction ($\leftarrow$ or L) and finally the upward direction ($\uparrow$ or U).

2. Write a function for the determination of the optimally aligned sequences:

   ```
   function [oSeqA, oSeqB] = computeSequences(iSeqA, iSeqB, iTrcM),
   ```

   where input arguments `iSeqA` and `iSeqB` represent sequences a in b, respectively, and `iTrcM` represents the trace matrix T. Output arguments `oSeqA` and `oSeqB` represent the optimally aligned sequences a and b, respectively.

   The optimally aligned sequences can be found by tracing the directions in the trace matrix T, starting from the last element in the matrix.

3. Write a function for computing the score of the optimal alignment:

   ```
   function oScr = computeScore(iSeqA, iSeqB, iSubS, iSubM, iGapP),
   ```

   where input arguments `iSeqA` and `iSeqB` represent optimally aligned sequences a and b, respectively, `iSubS` represents the symbols in the selected order (e.g. `'AGCT'`), `iSubM` represents the substitution matrix S with symbols in the same order, and `iGapP` represents the gap penalty P. The output argument `oScr` represents the optimal alignment score.

Validate the algorithm implementation by finding out the optimal alignment of two nucleotide (DNA) sequences a = GGATCGA and b = GAATTCAGTTA, for which the solution is given by a_{opt} = GGA-TC-G–A and b_{opt} = GAATTCAGTTA with the optimal alignment score of 3 when using substitution matrix S and gap penalty P:

S	A	G	C	T
A	2	−1	−1	−1
G	−1	2	−1	−1
C	−1	−1	2	−1
T	−1	−1	−1	2

$P = -2$

S^*	A	G	C	T
A	2	1	−1	−1
G	1	2	−1	−1
C	−1	−1	2	1
T	−1	−1	1	2

$P^* = 0$

Questions

The report includes handwritten (without using the computer) answers to the following questions, while the required images can be printed and enclosed in the report.

1. Determine the optimal alignment of nucleotide (DNA) sequences a = ACA and b = CGACT by using the substitution matrix S and gap penalty P. Write down the score matrix M, the trace matrix T (with arrows and marked optimal trace) and the score of the optimal alignment.

2. Determine the optimal alignment of nucleotide (DNA) sequences a = CTCTAGCATTAG and b = GTGCACCCA by using the substitution matrix S and gap penalty P. Write down the score of the optimal alignment.

3. Determine the optimal alignment of sequences from Question 1 by using the substitution matrix S^* and gap penalty P^*. Write down the score matrix M, the trace matrix T (with arrows and marked optimal trace) and the score of the optimal alignment.

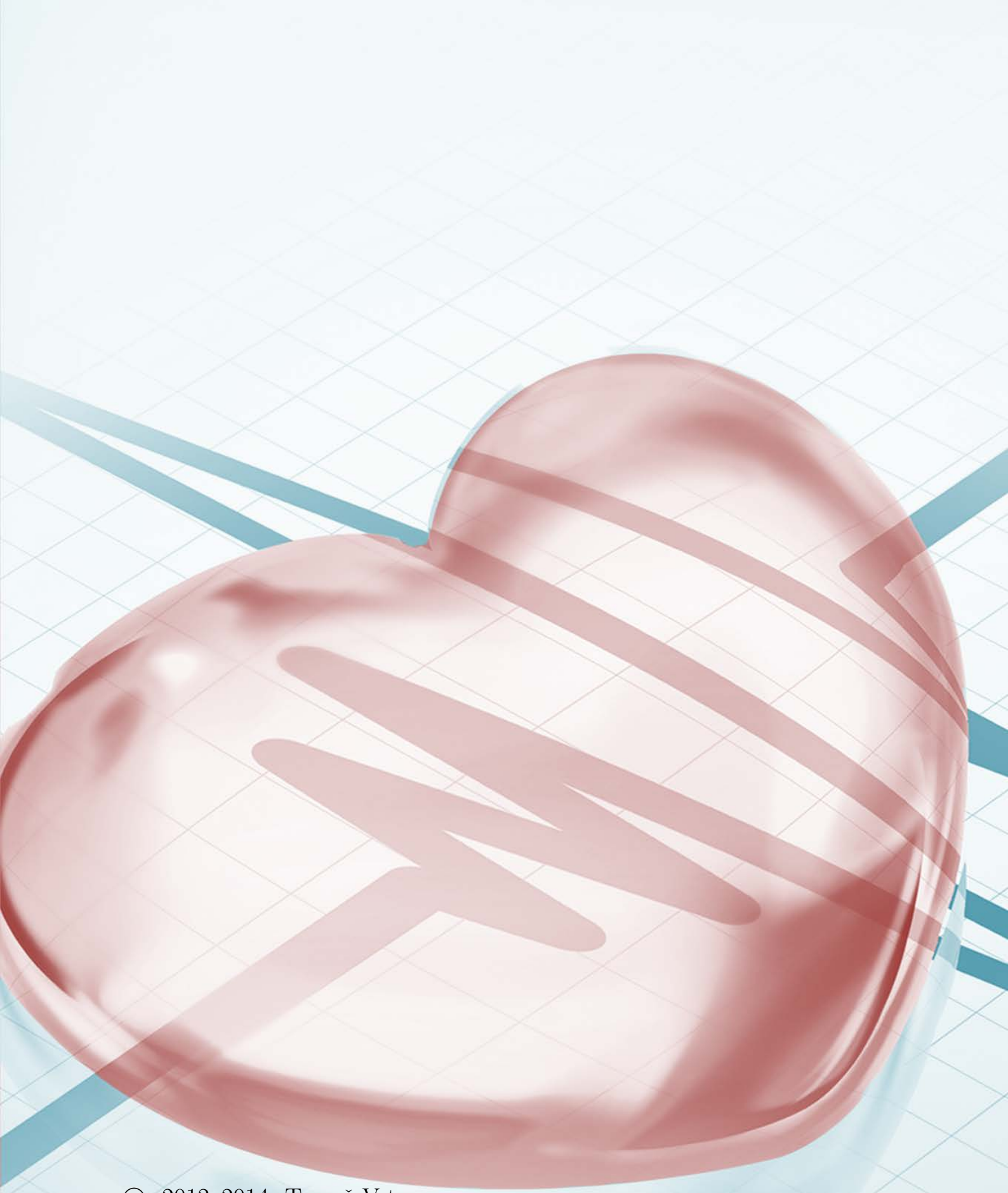
4. Determine the optimal alignment of sequences from Question 2 by using the substitution matrix S^* and gap penalty P^*. Write down the score of the optimal alignment.

5. What kind of simplifications were adopted for this implementation?

Additional Problems

The answers to the following questions do not need to be enclosed in the report, but should serve for a better understanding of the topic.

For this algorithm implementation, we ignored the fact that there may exist multiple optimally aligned sequences that result from multiple choices in the trace matrix T due to multiple maximal values in the recursive formula for the score matrix M. Modify the existing implementation so that it will allow the determination of all existing optimally aligned sequences.



http://lit.fe.uni-lj.si/BMI/
http://lit.fe.uni-lj.si/gradivo/BMI-LabVaje-eng.pdf