Computer Vision

License Plate Recognition

Klaas Dijkstra - Jaap van de Loosdrecht

1 September 2012

j.van.de.loosdrecht@tech.nhl.nl, k.dijkstra@tech.nhl.nl, jaap@vdlmv.nl

High demand for software engineers with computer vision experience

- **Quality control and quality insurance become more important**
  - **Efficient and effective**
  - **Recall actions**

- **Availability of low cost vision solutions**
  - **PC + camera**
  - **Intelligent camera**
  - **Cellphone**

- **Using libraries (No development from scratch)**

- **Structured testing**

- **Evaluation of large datasets**

- **Applications**
  - **Industry**
  - **Gaming**
  - **Surveillence**
  - **Augmented reality**

Demo

- **Framework**
- **Find License Plate**
- **Find Characters**
- **Match Plate**
- **Lexicon**

Overview

- **Licenseplate recognition competition**
  - **Goal**
  - **Planning**
  - **Example**
  - **Rules**

- **Matching**
  - **Not using a lexicon**
  - **Using a lexicon**
  - **Rejecting classification results**

- **Framework**
  - **VisionLab**
  - **Components**
  - **Finding the license plate**
  - **Finding the characters**
  - **Reading the licenseplate**

- **C# Application**
  - **User interface**
  - **UML()**

- **Appendix**
  - **Generating the pattern matcher file**

Goal

Next week (homework):

- **Take 50 photo's of license plates (each student)**
- **Taken with sensible white balance and exposure**
- **Different Angles / Lighting conditions**
- **All Dutch rectangular car license plates (yellow)**
- **One licenseplate per image (fully visible)**
- **File format: XXXXXX.jpg example: RVLG20.jpg, FBCG13.jpg, FBCG13-2.jpg, FBCG13-3.jpg etc.**

Before the end of the course (Deliverables)

1) **C# software for reading license plates**
2) **Report, with focus on the creativity of your solution**

Planning

Homework:

Week 1 – 4	**: Theory and assignments**
Week 5	**: Finish *LicencePlateMatcher.FindPlate()* in c#**
Week 6	**: Finish *LicencePlateMatcher.FindCharacters()* in c#**
Week 7	**: Fully functional**

Extra

- **Questions**
- **Receive final set**
- **Receive minimum score**

Midterm week

- **Final competition**
- **Report**
- **No tweaking possible!**

Examples

Rules

Not using the lexicon!

Scoring:

- **1 point for each correctly matched licenseplate**
- **0 points for each unrecognized licenseplate**
- **10 penalty points for each incorrectly recognized licenseplate**

Grading:

1) **Based on competencies of the course**
2) **Result of the competition on a selection of the photos**
3) **Minimum number of points will be determined after the selection of photos has been made**
4) **+1 for 1st and 2nd place (For the whole group)**

Matching

No Lexicon

Approach

1. **Match every character**
2. **Take character with the lowest error**
3. **Calculate confidence**

Applications:

- **Toll roads**
- **Speed camera's**

No Lexicon

1st	0	0.07	3	0.10	M	0.21	Z	0.08	H	0.58	Z	0.10
2nd	O	0.18	5	0.22	H	0.22	2	0.16	N	0.20	2	0.16
3nd	D	0.20	S	0.24	N	0.28	E	0.23	M	0.27	1	0.24

Green = Correct
Red = Incorrect
Blue = Not matched
Gray = Error

Confidence:

0.61	0.61	0.18	0.59	0.75	0.44

03MZHZ is incorrect !

Lexicon

Approach

1. **Match every character**
2. **Match license plate to possible license plates**
3. **Take plate with the lowest error for a whole word**
4. **Calculate confidence**

Applications:

- **Limited entry**
- **Camp sites**
- **Car parks**

Lexicon

1st	0	0.07	3	0.10	M	0.21	Z	0.08	H	0.58	Z	0.10
2nd	O	0.18	5	0.22	H	0.22	2	0.16	N	0.20	2	0.16
3nd	D	0.20	S	0.24	N	0.28	E	0.23	M	0.27	1	0.24

Confidence: 0.44

Green = Correct
Red = Incorrect
Blue = Not matched
Gray = Error

03**M**ZHZ is not a word in the lexicon

03**H**ZHZ is chosen instead, which is correct !

Rejection

1. **Reject confidences below a certain value to detect mismatches.**
2. **Check *ground-truth* agains the *match result***
3. **Count instances falling in each category**
4. **Score = True Positives - (10 * False Positives)**

Confusion matrix		*Check with ground-truth*	
		Correct	Incorrect
Match result	Accept	**True Positive**	**False Positive**
	Reject	**False Negative**	**True Negative**

Confidence threshold

Confidence threshold is a tunable parameter:

- Decreasing will move **False Negatives** to **True Positives** which is <u>good</u>
- Decreasing will move **True Negatives** to **False Positives** which is <u>bad</u>
- The opposite is true for increasing
- When trying to improve the overall performance (**True Positives** <u>and</u> **True Negatives**) the method or parameters have te be improved

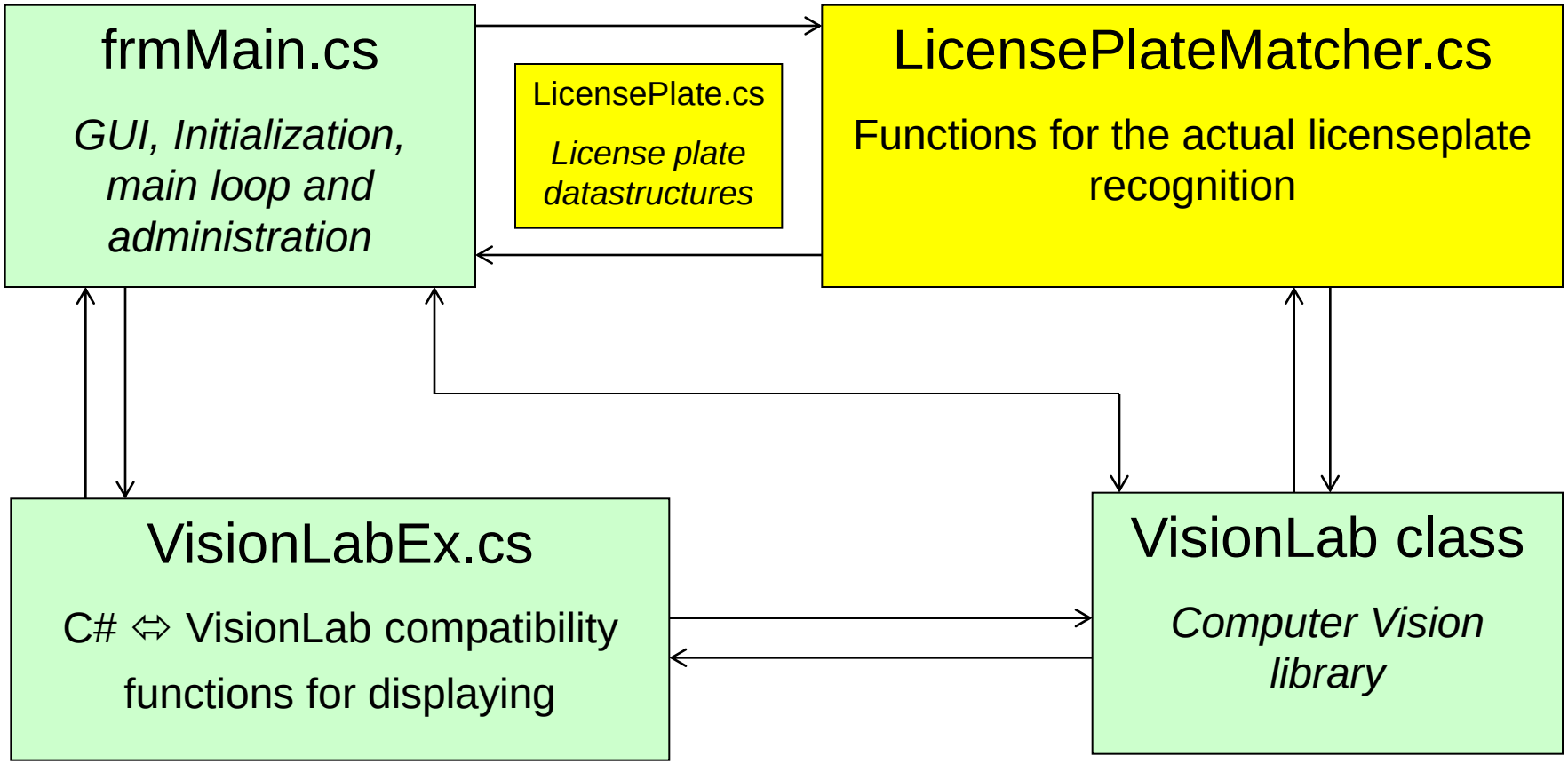
Framework

VisionLab

VisionLab is used for image processing (www.vdlmv.nl):

- Image processing algorithms
- Pattern matching
- Neural networks
- Genetic algorithms
- Algorithms written in ANSI C++
- OpenMP
- OpenCL
- Portable software:
  - Windows, Linux and Android
  - x86, x64, ARM and PowerPC

Components


frmMain.cs
GUI, Initialization, main loop and administration
LicensePlate.cs
License plate datastructures
LicensePlateMatcher.cs
Functions for the actual licenseplate recognition
VisionLabEx.cs
C# ⇔ VisionLab compatibility functions for displaying
VisionLab class
Computer Vision library

LicensePlate.cs

```
public class LicenseCharacter {
    public LicenseCharacter(string character, double error, double confidence)
    public string character()
    public double error()
    public double confidence()
    public new string ToString()
}

public class LicensePlate {
    public LicensePlate()
    public double confidence
    public List<LicenseCharacter> characters
    public string getLicensePlateErrorsString()
    public string getLicensePlateString()
    public new string ToString()
}
```

LicensePlateMatcher.cs

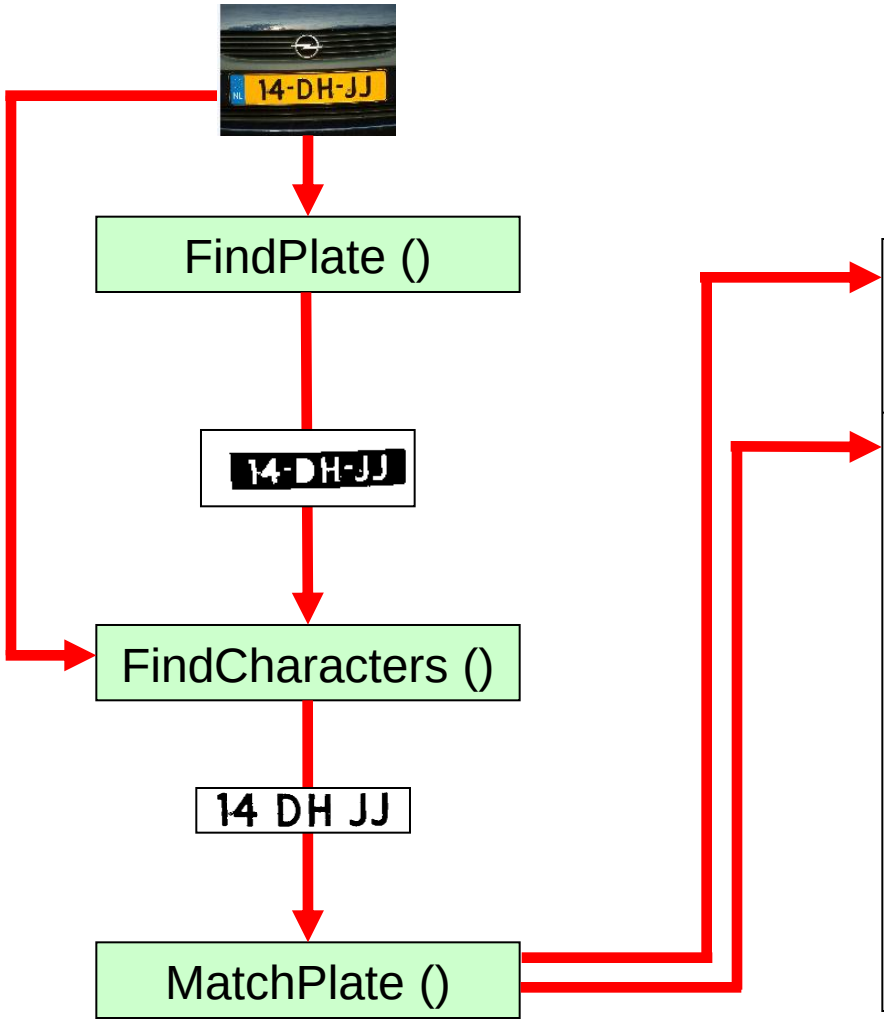
```
public class LicensePlateMatcher {

    public static bool FindPlate(                RGB888Image plateImage,
                                                 ref Int16Image binaryPlateImage)

    public static bool FindCharacters(           RGB888Image plateImage,
                                                 Int16Image binaryPlateImage,
                                                 ref Int16Image labeledRectifiedPlateImage)

    public static bool MatchPlate(               Int16Image binaryRectifiedPlateImage,
                                                 BlobMatcher_S16 matcher,
                                                 ClassLexicon lexicon,
                                                 ref LicensePlate result,
                                                 ref LicensePlate lexiconResult)
}
```

Production phase license recognition



14DHJJ	**Lexicon result**
0.7	Lexicon confidence
140HJJ	**Match result**
0.6	Confidence char 1
0.7	Confidence char 2
0.3	Confidence char 3
0.8	Confidence char 4
0.7	Confidence char 5
0.8	Confidence char 6

Finding the license plate

public static bool FindPlate ()

Description:

Find the largest license plate in the image

1. Segment using ThresholdHSVchannels
2. Remove blobs which are not license plates

Input:

```
//Original image
RGB888Image plateImage
```

Output:

```
//Segmented license plate
ref Int16Image binaryPlateImage
```

Return:

```
//License plate found?
bool
```

find_plate.jls

```
//Copy script selected image (F6)
copy %currentimage OriginalImage

//Convert from RGB888Image to HSV888Image
FastRGBToHSV OriginalImage OriginalImage

//************************************//
//*** Exercise: Find license plate **//
//************************************//

//Threshold HSV
ThresholdHSVchannels OriginalImage LicensePlateBin Int32Image 21 50 100 255 100 255

//Remove small blobs
RemoveBlobs LicensePlateBin EightConnected Area 1 5000 UseX

display LicensePlateBin

//Sum all pixels
$sum = SumIntPixels LicensePlateBin
if $sum > 0 then
  return true
else
  return false
endif
```

```
public static bool FindPlate( RGB888Image plateImage,
                              ref Int16Image binaryPlateImage ){

    const int c_threshold_h_min = 21;
    const int c_threshold_h_max = 50;
    const int c_threshold_s_min = 100;
    const int c_threshold_s_max = 255;
    const int c_threshold_v_min = 100;
    const int c_threshold_v_max = 255;
    const int c_remove_blobs_min = 1;
    const int c_remove_blobs_max = 5000;
    HSV888Image plateImageHSV = new HSV888Image();
    //Convert to RGB to HSV
    VisionLab.Convert(plateImage, plateImageHSV);

    //*******************************//
    //** Exercise:                 **//
    //**   adjust licenseplate     **//
    //**   segmentation            **//
    //*******************************//

    //Threshold HSV image
    VisionLab.Threshold3Channels(   plateImageHSV, binaryPlateImage,
                                    c_threshold_h_min, c_threshold_h_max,
                                    c_threshold_s_min, c_threshold_s_max,
                                    c_threshold_v_min, c_threshold_v_max
    //Remove blobs with small areas
    VisionLab.RemoveBlobs( binaryPlateImage, Connected.EightConnected,
                           BlobAnalyse.BA_Area,
                           c_remove_blobs_min, c_remove_blobs_max);
    plateImageHSV.Dispose();
    //Return true, if pixels found
    return (VisionLab.SumIntPixels(binaryPlateImage) > 0);
}
```

Solutions brainstorm for finding the license plate

1. **Tune *ThresholdHSVChannels***
   Find darkest and brightest yellow license plates
   Analyse HSV values, apply values, test values

2. **Tune *RemoveBlobs***
   Find smallest and largest licensplate
   Analyse *Area*, apply criteria, test criteria

3. **Add additional criteria**
   Add *RemoveBlobs* line using *HeightWithRatio* as a feature
   Add *RemoveBlobs* using additional features

4. **Add additional segmentation functions**
   Use a different *Threshold* values for each function

5. **Etcetera**
   This will score points

Finding the characters

public static bool FindCharacters ()

Description:

Locates the characters of the license plate

- **Warp image (Rectify)**
- **Segment characters**
- **Remove blobs which are to small (Lines between characters)**

Input:

```
//Original image
RGB888Image plateImage
//Segmented license plate
Int16Image binaryPlateImage
```

Output:

```
//Image containing binary six characters
ref Int16Image binaryCharacterImage
```

Return:

```
//Function executed successfully
bool
```

find_characters.jls

```
//copy script selected image (F6) and second selected image (F5)
copy %currentimage OriginalImage
copy %secondimage LicensePlateBin

//Find corner points of the licenseplate
$found = FindCornersRectangle LicensePlateBin EightConnected 0.5 Landscape &$tab
if $found then
  //Warp (rectify) licenseplate
  Warp OriginalImage binaryCharacterImage ForwardT          $tab[0] $tab[1] $tab[2] $tab[3]
                                                            100 470 0
                                                            NearestPixelInterpolation

  //**********************************************//
  //*** Exercise: Find license plate characters **//
  //**********************************************//

  Convert binaryCharacterImage binaryCharacterImage Int16Image
  //Automatic threshold finds black letters
  ThresholdIsodata binaryCharacterImage DarkObject
  //Remove all blobs connected to the border
  RemoveborderBlobs binaryCharacterImage EightConnected AllBorders
  //Remove all blobs with an area less than 400
  RemoveBlobs binaryCharacterImage EightConnected Area 1 400 UseX

  display binaryCharacterImage
  return true
else
  return false
endif
```

Solutions brainstorm for finding characters

1. **Try *FindCornersRectangeSq* operator/function**
   Instead of *FindCornersRectangle*

2. **Tune *RemoveBlobs***
   Find smallest and largest licenseplate
   Analyse *Area*, apply criteria, test criteria

3. **Add additional criteria**
   Add *RemoveBlobs* using additional features

4. **Add additional segmentation functions**
   Use a few different *Threshold* values and methods (manual vs. automatic)

5. **Use a different color space**
   Use HSV

6. **Use binary morphological filters to dilate or erode blobs**
   Try making the blobs more like the characters in the .pm file

7. **Etcetera**
   This will score points

Reading the license plate

public static bool MatchPlate ()

Description:

Read the license plate

Input:

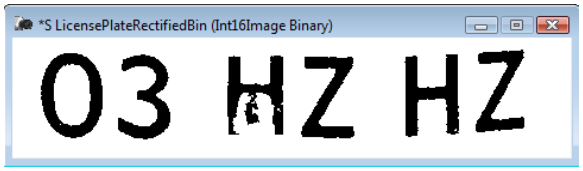
```
//Rectified license plate image containing six characters
Int16Image labeledRectifiedPlateImage
BlobMatcher_Int16 matcher      //initialized blobmatcher
ClassLexicon lexicon           //initialized lexicon
```

Output:

```
//Result by the blob matcher
ref LicensePlate result
//Result by the lexicon
ref LicensePlate lexiconResult
```

Return:

```
//six characters found
bool
```




```
03HZHZ 0.61
03MZHZ 0.67 0.61 0.18 0.59 …
```

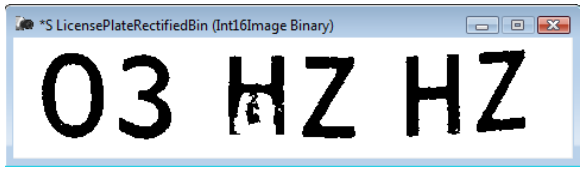
```
//Copy third selected image (Operator->Select 3rd)
copy %thirdimage LicensePlateRectifiedBin
$lpwd = lpwd
cwd $lpwd
PM_ReadFromFile PatternMatcher lic_fonts.pm  //Read pattern matcher file
//Analyse blobs locations
copy LicensePlateRectifiedBin LicensePlateRectifiedLabel
labelblobs LicensePlateRectifiedLabel EightConnected
$maxBlobIndex = BlobAnalysisArray LicensePlateRectifiedLabel &$tab    SortDown TopLeft UseX
                                                                       Height TopLeft Width
if $maxBlobIndex !== 5 then //Check if 5 characters were found
  return false
endif
//Begin matching in a for loop
$confidences = ""
$matches = ""
for $i = 0 to $maxBlobIndex do
  VarToArray &$tab[$i] &$elm
  $label = $elm[0]
  $h = $elm[1]
  $tl = $elm[2]
  $w = $elm[3]
  $x = getnthfromvector 1 $tl
  $y = getnthfromvector 2 $tl
  ROI LicensePlateRectifiedBin LicensePlateRectifiedBinROI $x $y $h $w
  $bestMatch = PM_BestMatch LicensePlateRectifiedBinROI PatternMatcher -0.5 0.5 //Match
  $patternId = GetNthWord 1 $bestMatch  //Get pattern ID
  $confidence = GetNthWord 2 $bestMatch //Get confidence
  $patternName = PM_PatternName PatternMatcher $patternId //Convert pattern ID to confidence
  $confidences = concat $confidences $confidence
  $matches = $matches . $patternName
endfor
//Return results
$result = concat $matches $confidences
return $result
```

Solutions brainstorm for reading the license plate

1. **Adapt pattern matcher (.pm file)**
   Analyse which characters occur in real license plates
   Use a different font in the .pm file

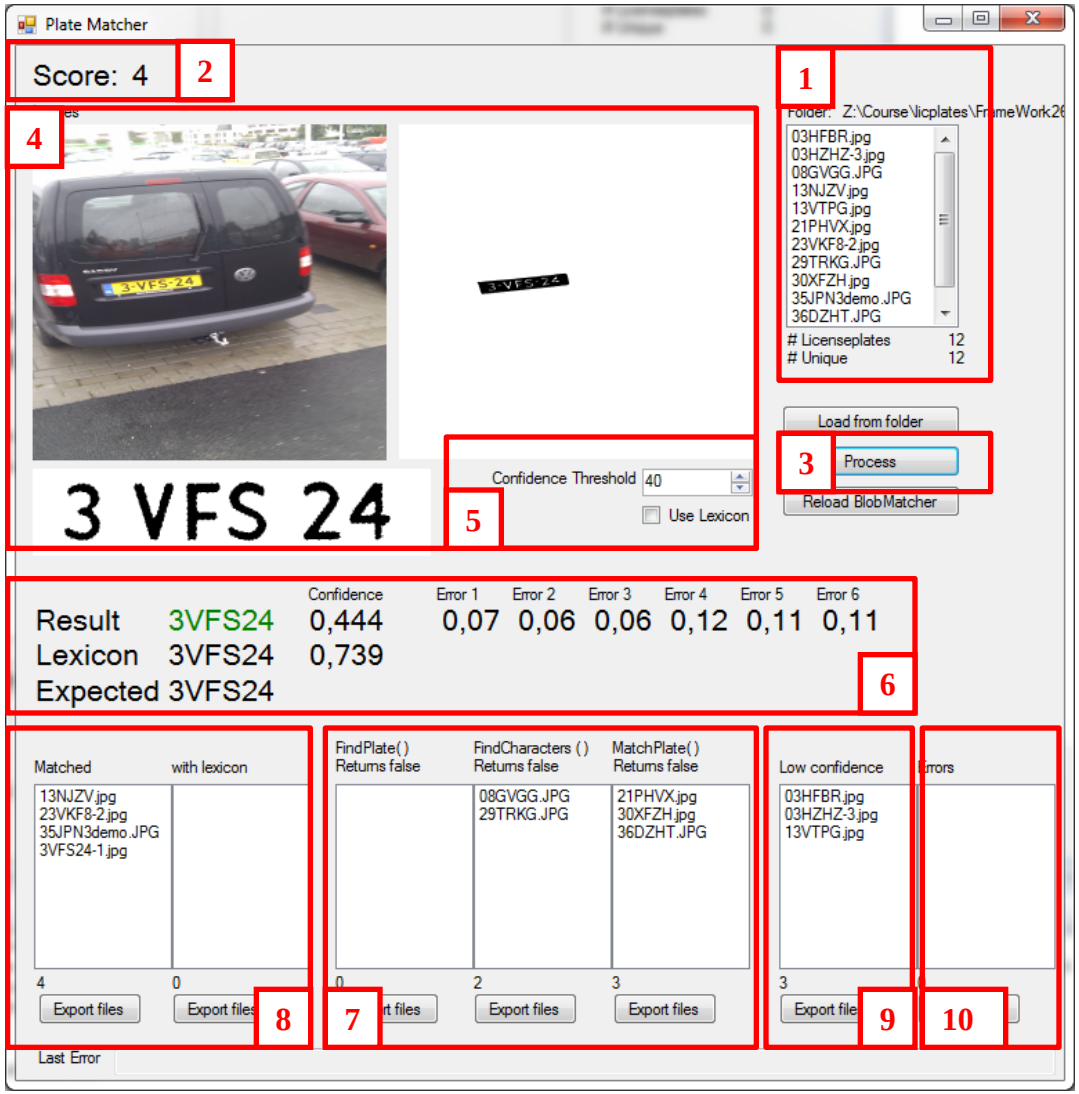
2. **Check or correct license plate grammer**
   D3-HF-BR is not very likely to be a license plate, while 03-HF-BR is

3. **Etcetera**
   This will score points

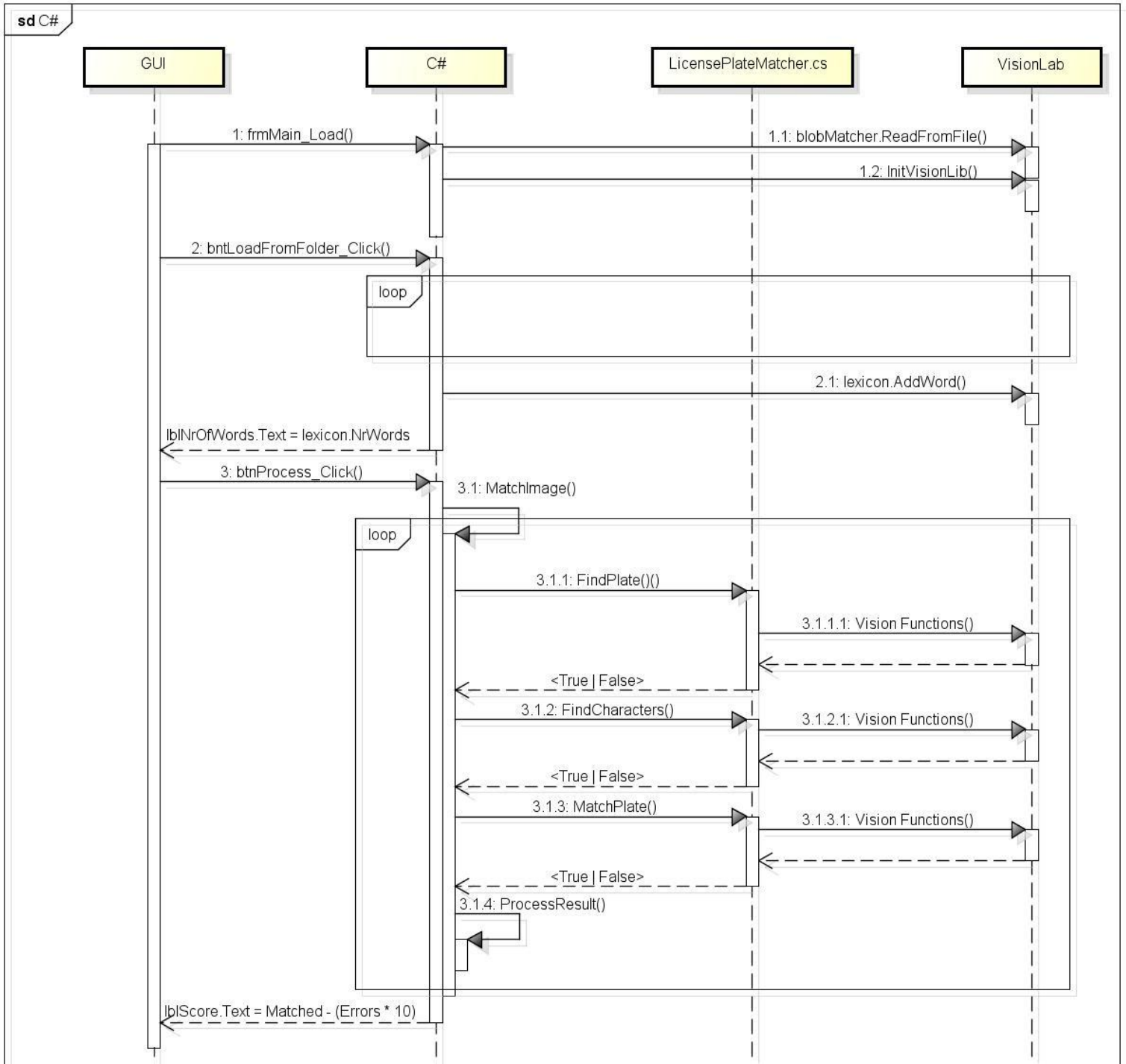



```
03HZHZ 0.61
03MZHZ 0.67 0.61 0.18 0.59 …
```

Framework



1. Images to be processed
2. Current score
3. Begin processing
4. Output images from LicensePlateMatcher.cs functions
5. Mark as low confidence below this number divided by 10
6. Match result from current licenseplate
7. Functions return for these licenseplates
8. Correct match and high confidence
9. Confidence is too low
10. Confidence is high, but match is wrong


sd C#
GUI
C#
LicensePlateMatcher.cs
VisionLab
1: frmMain_Load()
1.1: blobMatcher.ReadFromFile()
1.2: InitVisionLib()
2: bntLoadFromFolder_Click()
loop
2.1: lexicon.AddWord()
lblNrOfWords.Text = lexicon.NrWords
3: btnProcess_Click()
3.1: MatchImage()
loop
3.1.1: FindPlate()()
3.1.1.1: Vision Functions()
<True | False>
3.1.2: FindCharacters()
3.1.2.1: Vision Functions()
<True | False>
3.1.3: MatchPlate()
3.1.3.1: Vision Functions()
<True | False>
3.1.4: ProcessResult()
lblScore.Text = Matched - (Errors * 10)

Appendices

gen_lic_pm.jls (VisionLab)

Idea:

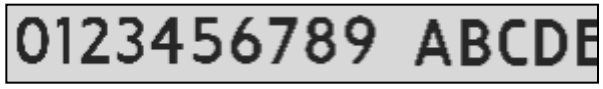
Generate the license plate *pattern matcher*

Input:

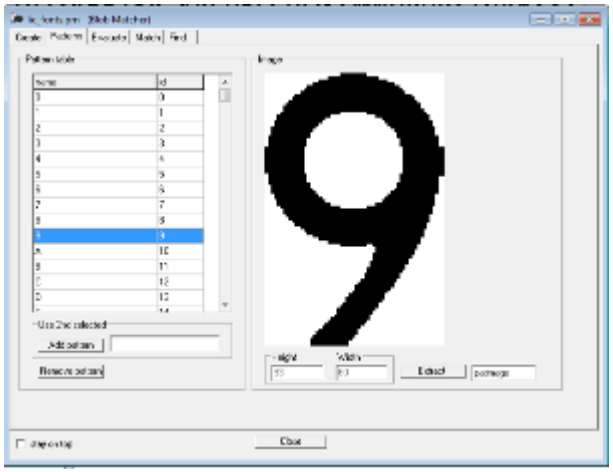
lic_fonts.jl => Image contaning the characters

Return:

lic_fonts.pm => Pattern matcher containing the characters



Lic_fonts.jl



Lic_fonts.pm

```
PM_CreateBlobMatcher pm Int16Image 60 1 20 0
$names = 0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
VarToArray &$names &$nameTab
lread allPats lic_fonts.jl
copy allPats allPatsB
ThresholdIsodata allPatsB DarkObject
$nrNums = LabelBlobs allPatsB EightConnected
$maxBlob = BlobAnalysisArray allPatsB &$tab SortDown TopLeft UseX Height TopLeft Width
for $i = 0 to $maxBlob do
  VarToArray &$tab[$i] &$elm
  $label = $elm[0]
  $h = $elm[1]
  $tl = $elm[2]
  $w = $elm[3]
  $x = getnthfromvector 1 $tl
  $y = getnthfromvector 2 $tl
  ROI allPats roi $x $y $h $w
  copy roi roiB
  Threshold roiB 0 100
  PM_AddPattern roiB pm $nameTab[$i]
endfor
PM_WriteToFile pm lic_fonts.pm
PM_Delete pm
```