Playing Card Recognition

Computer Vision (EE4H) — Final Report

Yousef Amar (1095307)
Chris Lewis (1234567)

2014-04-28

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Contents

1 Introduction

This is a paragraph. Here's an example of cross-referencing: please see Review (section 2 on page 2).

1.1 Project Specification

This is a *subsection*. Whitespace is mostly meaningless in LaTeX. Indenting with tabs is just useful for collapsing parts of the document you aren't working on in Sublime; it has no effect on document appearance. Speaking of Sublime, these are some useful packages for working in LaTeX:

1. "LaTeXTools" (better than "LaTeXing") because
   - (a) Doesn't pester you to buy it
   - (b) Has pretty good syntax highlighting and auto completion
   - (c) Auto completion for cross-referencing figures/sections/appendices and citations too
   - (d) Includes LaTeX build system — Ctrl+B to compile; no command line headaches
2. Edit your settings file for really smooth spell check: `http://www.sublimetext.com/docs/3/spell_checking.html`
3. "Origami" for multiple panes in Sublime — useful to have notes open as a clipboard on the side
4. "Increment Selection" (came in handy once or twice)

2 Review

Here's another section. Here's a figure with 0.8 of the line width:

Figure 1: Example of Card

LATEXis pretty good at figuring out what to do with figure. There are a number of options for them too that are cool. Here's me cross-referencing the figure without breaking a sweat: please see figure 1 on page 2.

3 Proposed Method

LaTeXis really good at equations. Here's a simple one from my report:

$$x = \frac{ct}{2} - \epsilon$$

Want some confusion matrices? Here are some matrix examples from my final report:

$$T = \begin{bmatrix} 84 & 200 & 16 \\ 56 & 238 & 11 \\ 251 & 206 & 203 \end{bmatrix} \qquad I = \begin{bmatrix} 160 & 40 & 250 & 27 & 114 \\ 81 & 94 & 14 & 100 & 37 \\ 225 & 52 & 148 & 137 & 111 \end{bmatrix}$$

Here's me sprinkling some math inline: θ, x and y, I_1 and I_2, $|T - I_n|$. Want some graphs? Google pgfplots and prepare to be blown away.

4 Implementation

You can also put a lot of stuff in one figure even on different lines:

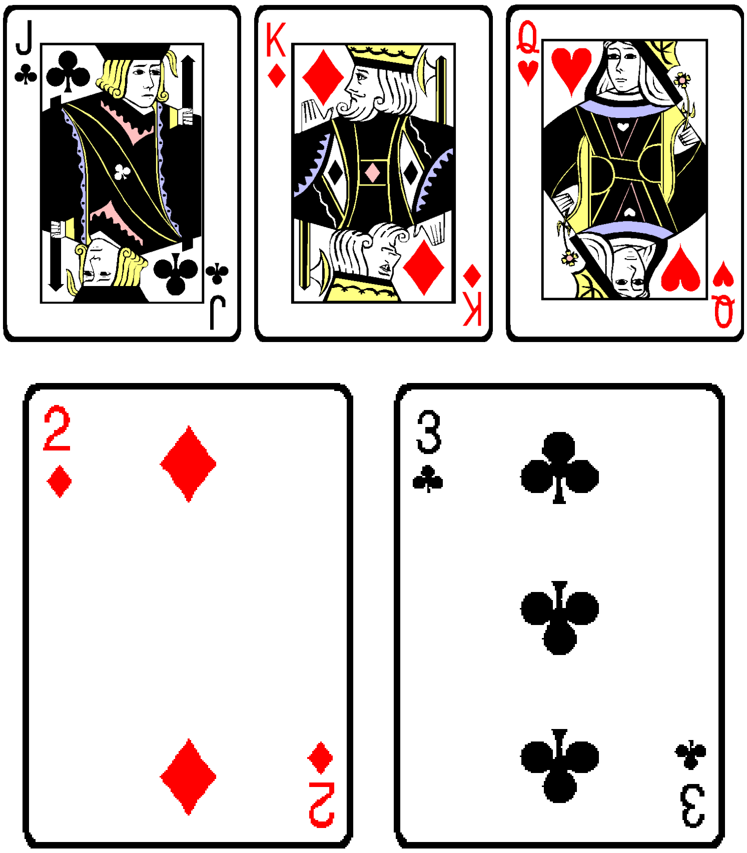

Figure 2: Example of Cards

5 Evaluation

See appendix A.1 on page 8 for an example of code listings. The style is completely customisable of course. Notice also that it's not copied to the *appendices* directory, it's just a symlink to *src*! According to this random book off of Google Scholar, computer vision is pretty cool (Forsyth and Ponce, 2002). I use BibLatex + Biber to compile the bibliography; the order is "latexmk" (or build), "biber report", then "latexmk" again. This only needs to be done if the bibliography changes and then it might as well be done only really at the very end when the report is finished to so don't worry about it. LaTeXing does it automatically but it's not worth it.

6 Conclusion

We've only just scratched the surface! Here, have a mini flow chart:

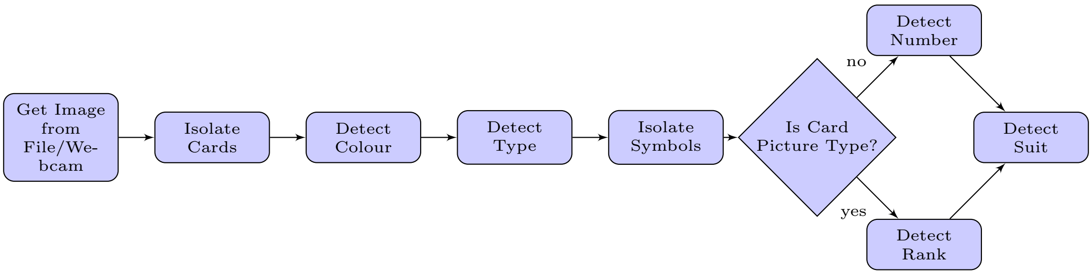

Figure 3: Top-level Flowchart

References

Forsyth, David A and Jean Ponce (2002). *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference.

Appendices

A Source Code Listings

A.1 main.cpp

```
/************************************************************************\
| Main code file for the EE4H Assignment (Playing card recognition)      |
|                                    |
| Authors: Yousef Amar and Chris Lewis                   |
|                                    |
| Dependencies: OpenCV-2.4.8                       |
|       - opencv_core248.dll                   |
|       - opencv_imgproc248.dll                  |
|       - opencv_highgui248.dll                  |
|       - tbb.dll (Built for Intel x64)              |
\************************************************************************/

#include "../include/stdafx.h"

using namespace std;

int process_image(cv::Mat input)
{
  cv::Size input_size = input.size();

  //Check size is greater than zero
  if(input_size.width > 0 && input_size.height > 0)
  {
    //Show image details
    cout << "Image is " << input_size.width << " by " << input_size.height << "
    pixels." << endl << endl;
    //cv::imshow("Input", input);

    //Hack to deal with super large, hi-res images
    if (input_size.width > 1000)
      cv::resize(input, input, cv::Size(1000, 1000*input_size.height/input_size.width));
    if (input_size.height > 500)
      cv::resize(input, input, cv::Size(500*input_size.width/input_size.height, 500));

    //Find cards in image and populate cards vector
    vector<Card> cards;
    if (find_cards(input, &cards)) {
      cerr << "Could not find any cards!" << endl;
      return -4; //No cards found
    }

    for(size_t i = 0; i < cards.size(); i++)
    {
      Card *card = &cards[i];

      //Detect suit colour
      detect_colour(card);

      //Detect suit type (number/picture)
      detect_type(card);

      //Find symbols
      find_symbols(card);

      //Get card value
      if (!card->is_picture_card)
      {
        detect_value_number(card);
      }
      else
      {
        detect_value_picture(card);
      }

      //Find suit
      find_suit_sym(card, 0.95F);
    }

    //Show results until key press
    show_cascade(cards);

  }
  else
  {
    cerr << "Image dimensions must be > 0!" << endl;
    return -2;  //Image size zero code
  }

  //Finally
  cout << "Processing finished successfully!" << endl;
  return 0; //No error code
}


/**
  * Program entry point.
  *
  * Arguments
  *    int argc: Number of arguments
  * char** argv: Array of arguments: [1] - Image path to open
  *
  * Returns
  * int: Error code or 0 if no error occured
  */
int main(int argc, char **argv)
{
  cout << endl << "----------------------------------------------" << endl
         << " EE4H Assignment - Recognising playing cards  " << endl
         << " By Yousef Amar and Chris Lewis               " << endl
         << "----------------------------------------------" << endl << endl;

  //Check image is provided
  if(argc < 2)
  {
    cout << "Arguments error. Check image path/format? Did you mean to use --cam?" <<
    endl;
    return -1;  //Incorrect arguments code
  }

  cv::Mat input;

  bool from_cam = !strcmp(argv[1], "--cam"), should_quit = false;

  if (argc > 2) {
    cout << "Multi mode activated!" << endl;
    multi_mode = !strcmp(argv[2], "--multi");
  }

  do
  {
    if(!from_cam)
    {
      input = cv::imread(argv[1], CV_LOAD_IMAGE_COLOR);
    }
    else
    {
      cv::VideoCapture cap(CV_CAP_ANY);
      cv::waitKey(1000);
      if(!cap.isOpened())
      {
        cerr << "Unable to access webcam" << endl;
        return -3;  //Incorrect arguments code
      }

      cout << "Press space to take a photo" << endl;

      char key;
      do
      {
        key = cvWaitKey(10);

        cap >> input;

        cv::imshow("Webcam", input);

        // Break out of loop if Space key is pressed
      } while (char(key) != 32);

      cv::destroyWindow("Webcam");
    }

    if (process_image(input))
      return EXIT_FAILURE;

    if (!from_cam)
    {
      cout << "Press any key to quit" << endl;
    } else {
      cout << "Press Esc to quit, any other key to try again" << endl;
    }

    should_quit = (from_cam && cv::waitKey(0)==27) || (!from_cam && cv::waitKey(0));

  } while(!should_quit);

  return EXIT_SUCCESS;
}
```