University of Nottingham
UK | CHINA | MALAYSIA

EEEE1002

Computer Vision with OpenCV

Daniel Fallows

Introduction

- What is computer vision and OpenCV?
- How we will use OpenCV in Sessions 6 and 7

- How to approach OpenCV programming
  - Follow along example: Isolate a colour using HSV
  - Add to code: counting pixels

- Example code explanations

- Next week: Pre-session task

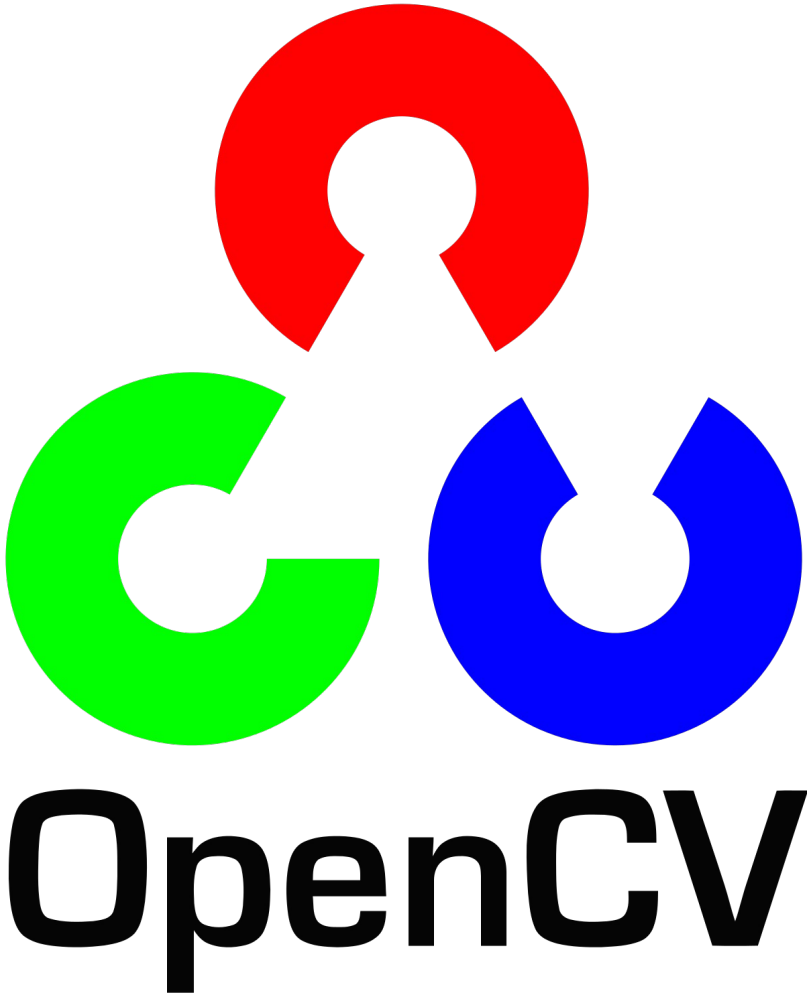
What is Computer Vision?

- The use of software to automatically extract useful information from an image
  - ANPR (automatic number plate recognition)
  - Face unlock
  - Quality control
- Can be used on static images but more common on video feeds
- Requires lots of processing power (i.e. more than an Arduino)

What is OpenCV?

- An open-source multiplatform library to aid the development of image processing and computer vision software

- Contains functions to:
  - Modify images (e.g. size, colour space, etc)
  - Locate features (e.g. edge detection)
  - Perform mathematical operations on images
  - etc



University of Nottingham
UK | CHINA | MALAYSIA

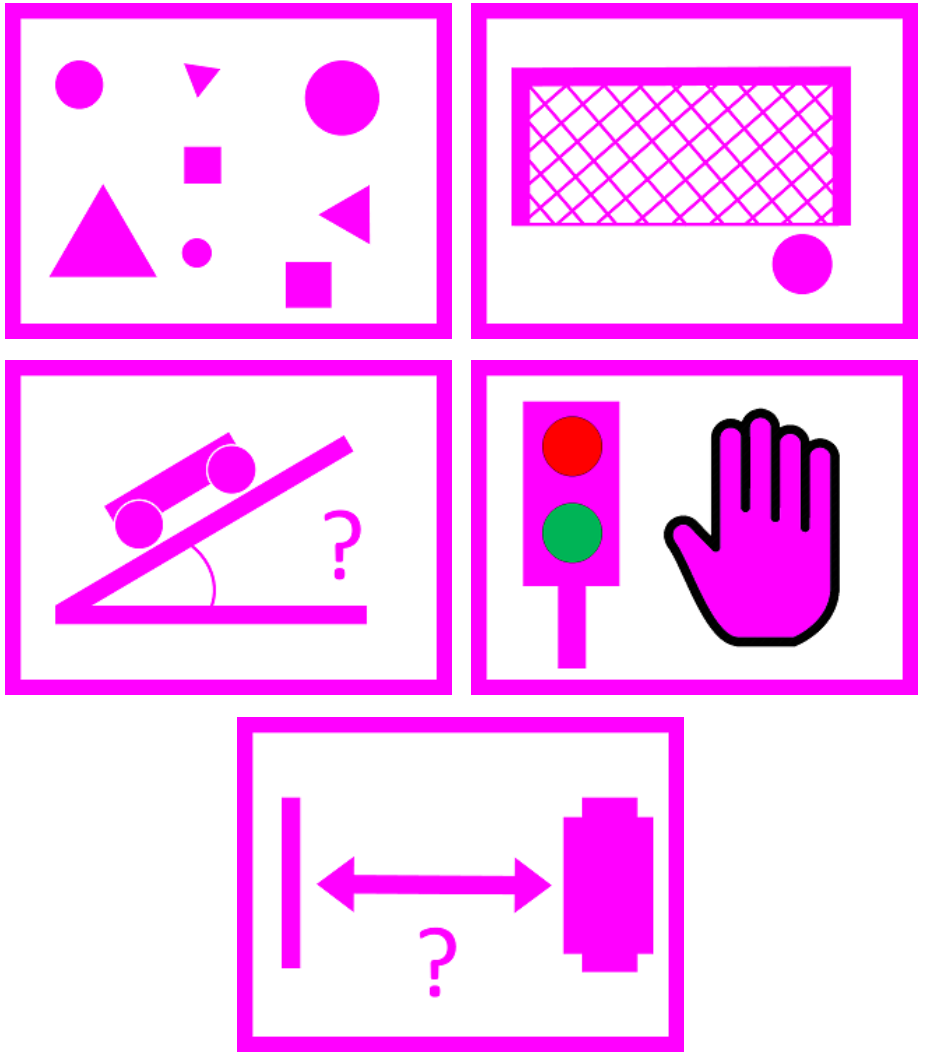
Sessions 6 & 7

OpenCV in the project

- Final challenge will involve 2 parts:
  1. Timed line following
     - Giant track with multiple colour lines
     - Obstacles (e.g. ramps, cones)
     - You will need remote (or automatic) start/stop
     - You will need to accurately time your course and show the result (e.g. LCD, speaker, etc)
  2. Mini challenges
     - Each worth successfully completed worth marks
     - Involve using computer vision

OpenCV in the project

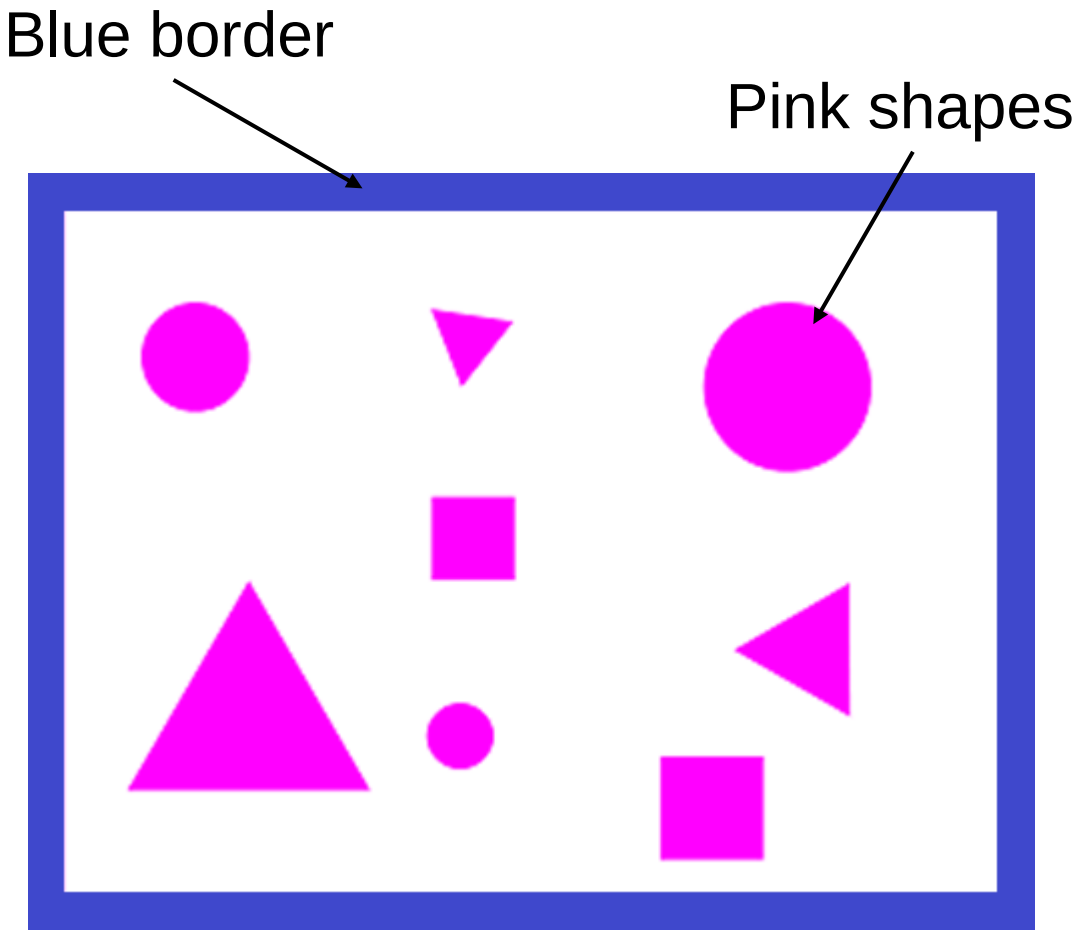
- Mini-challenges will include:
  - Stop light
  - Push a football into a goal
  - Measure the incline of a ramp
  - Count the number of shapes on a wall
  - Measuring distance to an object




University of
Nottingham
UK | CHINA | MALAYSIA
Developing
with OpenCV

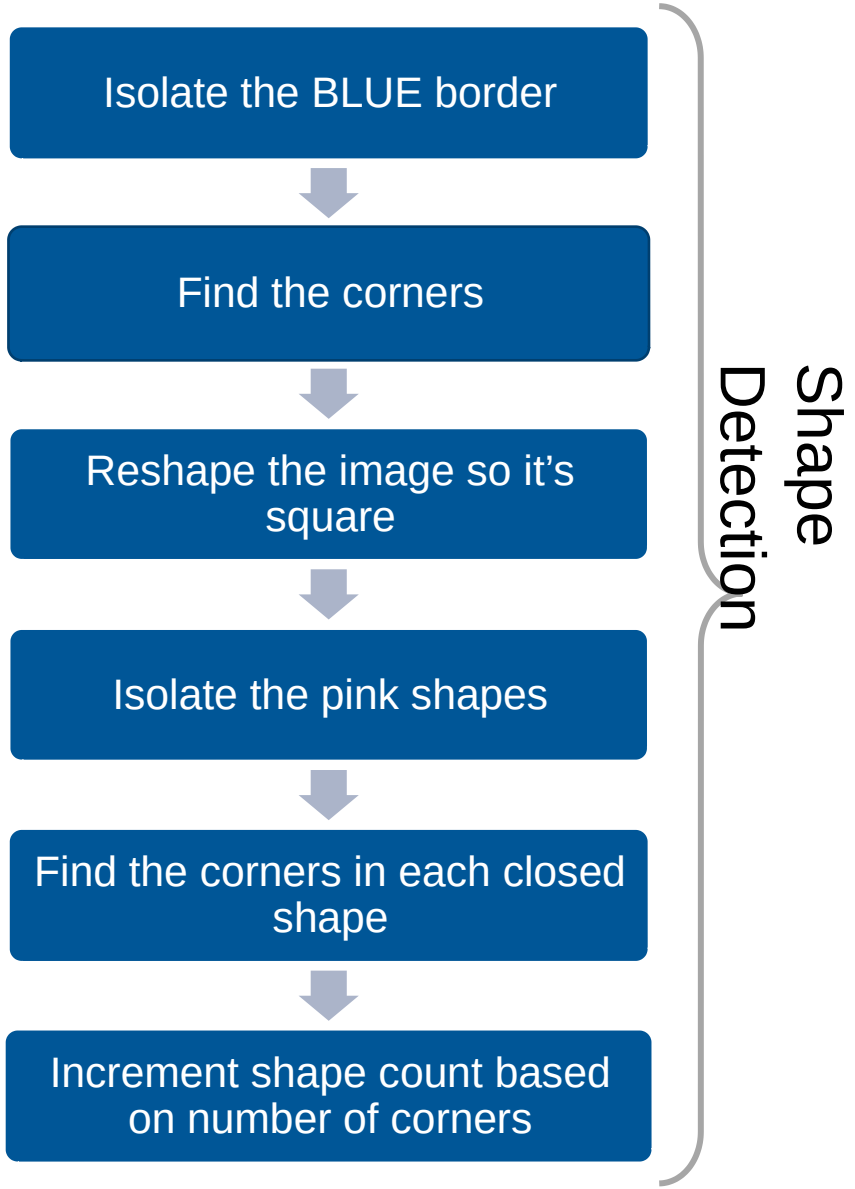
OpenCV development

- Take a sample photo to work on
- Identify key features of the image:
  - Colours
  - Borders
  - Corners
- Isolate these features
- Process the resulting images



Good practise

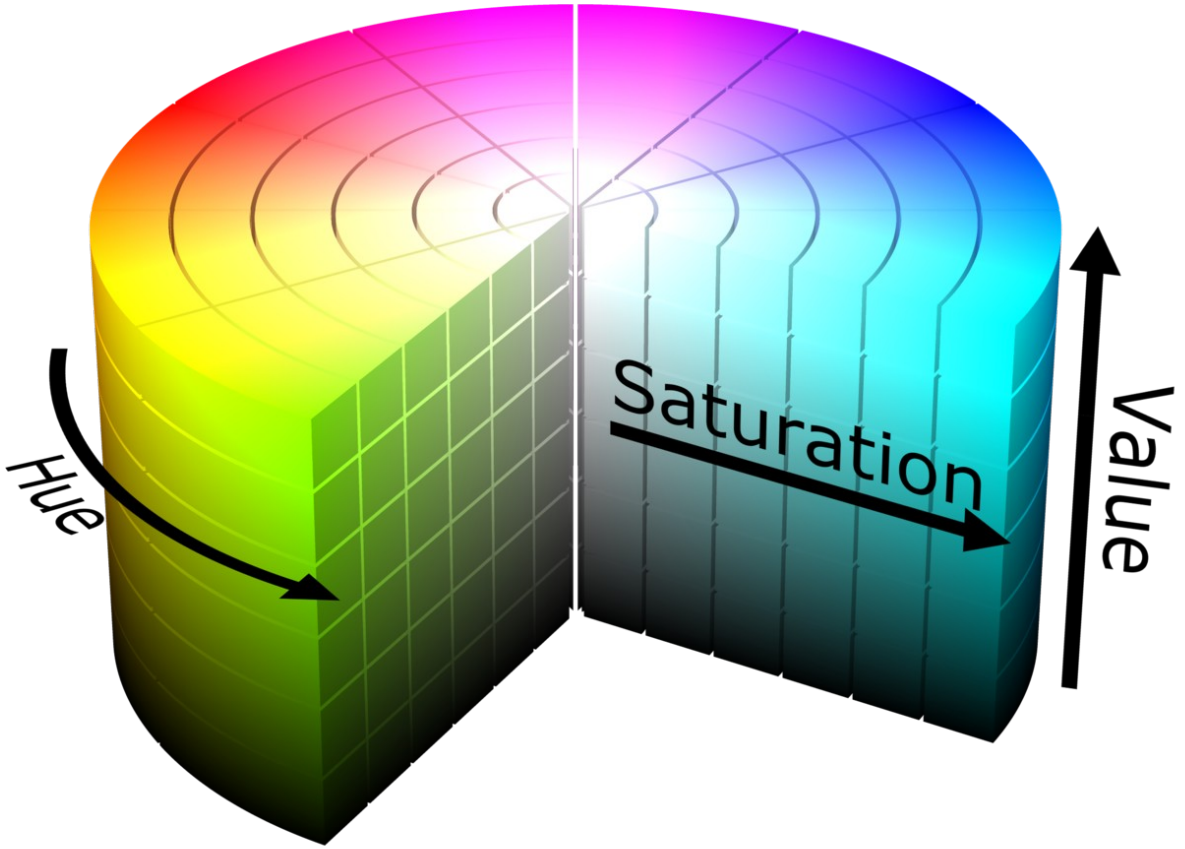
- Start with a flowchart
  - Include how the image should change at each stage
- When testing display the image at each stage
  - Does it look correct? Which bits are highlighted? Does the lighting condition affect the result? Etc…




University of
Nottingham
UK | CHINA | MALAYSIA
Example Code

Example – isolating colour:

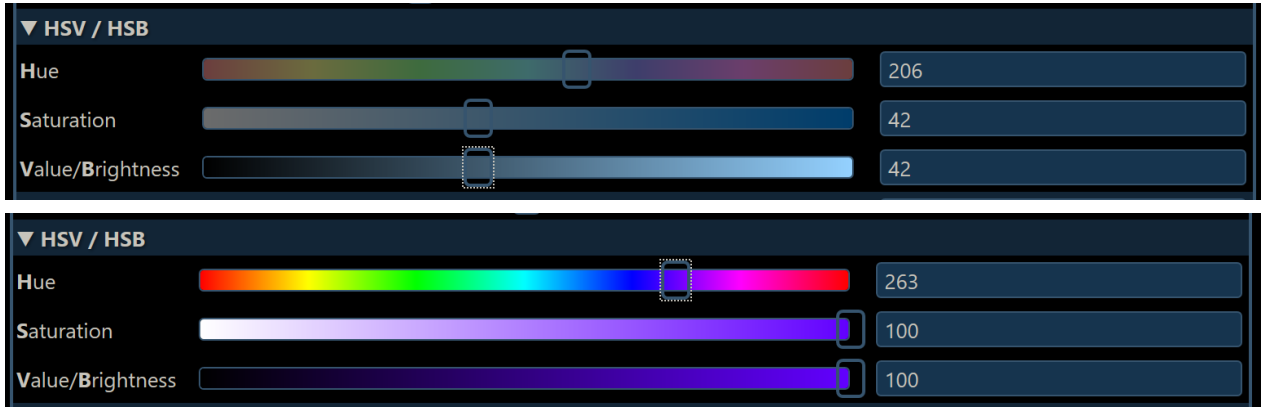
- Each pixel of an image is stored as colour data
  - BGR – Blue, Green, Red intensities
  - HSV – Hue, Saturation, Value
  - Others…
- HSV makes it easy to isolate colours:
  - Hue = colour tone
  - Saturation = intensity of the colour
  - Value = brightness

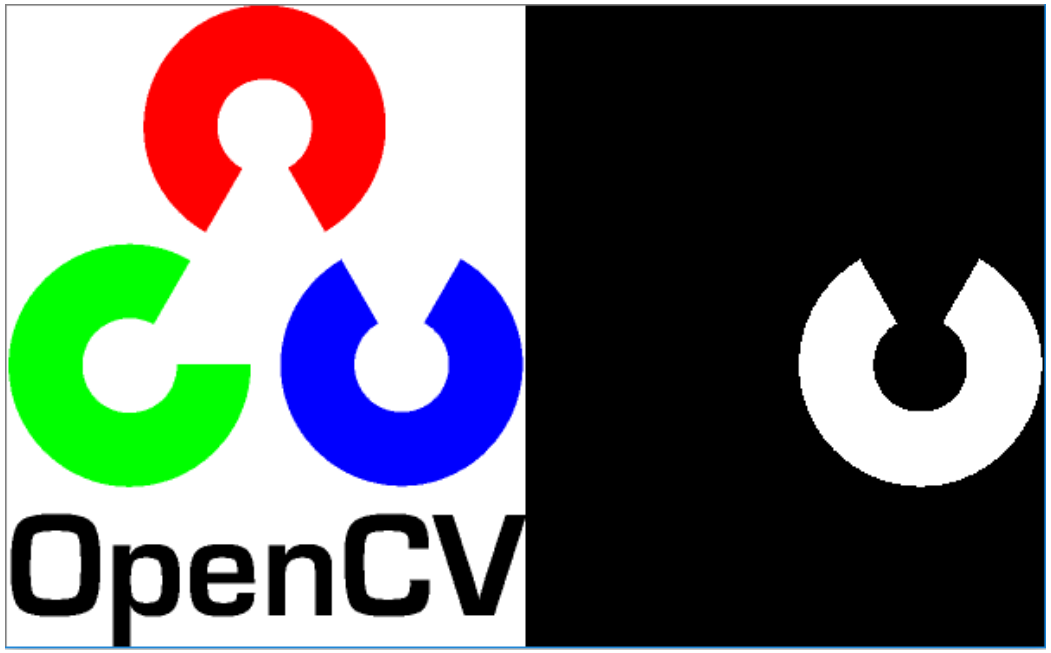


Developing an OpenCV program

Example – isolating colour:

- Let's isolate the blue pixels in the OpenCV logo.
- We need to choose the lower and upper HSV limits...
  - Start with a colour picker tool e.g. http://colorizer.org/
- Use the 'HSVTool' example project to test
  - Note numbers must be scaled

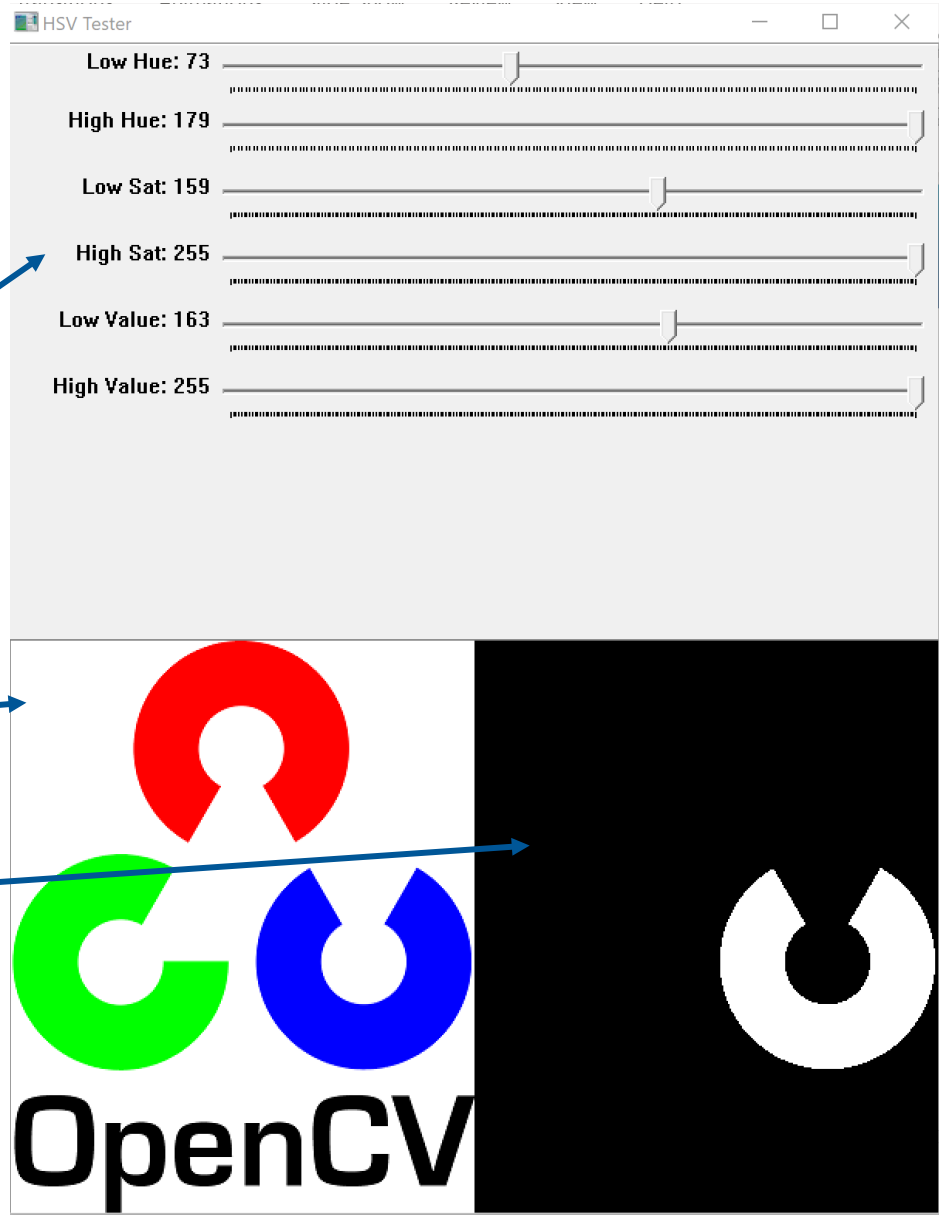




Developing an OpenCV program

Example – HSV Tool:

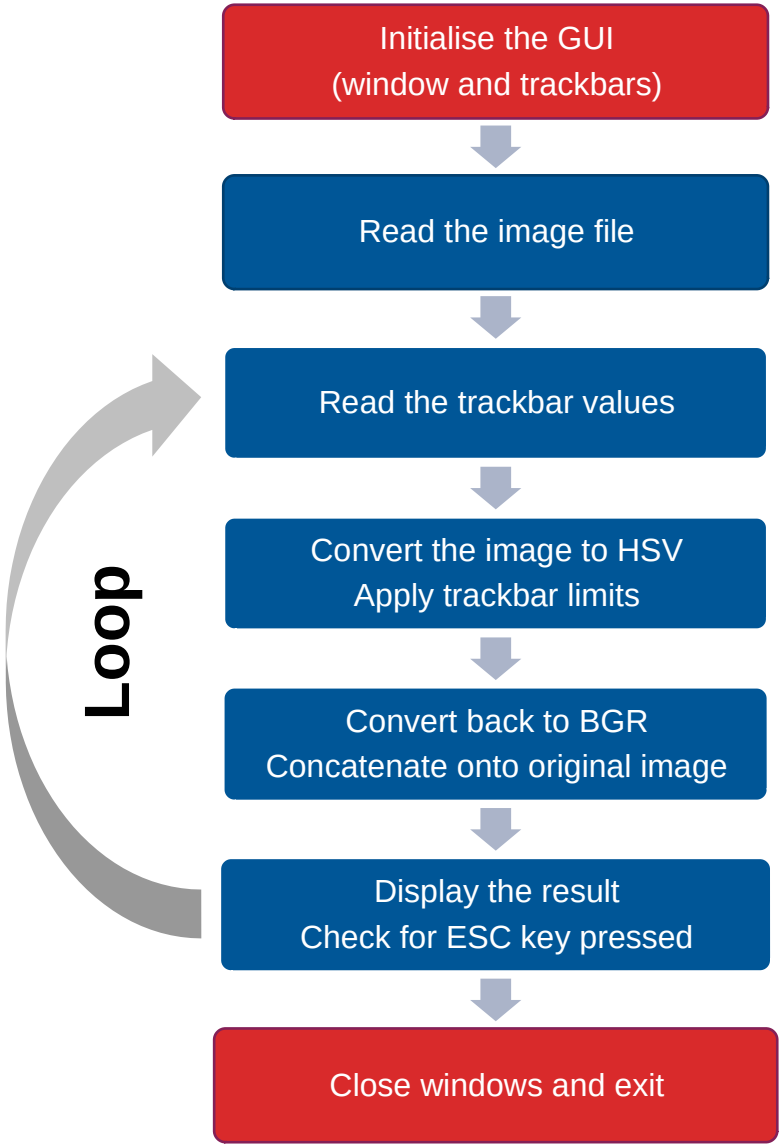
- Use GUI slider controls to isolate a chosen HSV range in an image
  - Slider controls set lower and upper limits
  - Original image is shown
  - Alongside filtered image



HSV Tool Flowchart:

- This is quite a basic program
  - Simple flowchart

- Yours will be much more complex
  - Split tasks into functions
  - Make multiple small flowcharts
  - Test stage by stage and record results
  - Read the lab handouts (these have example of the functions you will need)



HSV Tool Code:

- Included libraries:
  - Standard C/C++ ones for terminal IO
  - OpenCV libraries for image functions
- Select the OpenCV namespace
  - Saves preceding functions with CV::

```
1   /*
2   ********************************************
3   Tool to make it easier to find the HSV v
4   The raw camera feed is shown along with
5   ********************************************
6   */
7
8   // Include files for required libraries
9   #include <stdio.h>
10  #include <iostream>
11  #include "opencv2/core.hpp"
12  #include "opencv2/highgui.hpp"
13  #include "opencv2/imgproc.hpp"
14
15  using namespace cv;
```

HSV Tool Code:

- In main() – create a display window, configure the trackbar (slider) controls, read the input image

```
17    int main( )
18    {
19        namedWindow("HSV Tester");    // Create a GUI window called photo
20
21        int lowH = 0, highH = 179, lowS = 0, highS = 255, lowV = 0, highV = 255;
22
23        createTrackbar("Low Hue", "HSV Tester", &lowH, 179, NULL);       // Create tr
24        createTrackbar("High Hue", "HSV Tester", &highH, 179, NULL);
25
26        createTrackbar("Low Sat", "HSV Tester", &lowS, 255, NULL);
27        createTrackbar("High Sat", "HSV Tester", &highS, 255, NULL);
28
29        createTrackbar("Low Value", "HSV Tester", &lowV, 255, NULL);
30        createTrackbar("High Value", "HSV Tester", &highV, 255, NULL);
31
32        Mat frame = imread("OpenCV_Logo.png"); // Open an image file and store in a
```

HSV Tool Code:

- Main loop
  1. Read the trackbar positions and store in variables

```
34      while(1)    // Main loop to perform image processing
35      {
36          lowH = getTrackbarPos("Low Hue", "HSV Tester");
37          highH = getTrackbarPos("High Hue", "HSV Tester");
38          lowS = getTrackbarPos("Low Sat", "HSV Tester");
39          highS = getTrackbarPos("High Sat", "HSV Tester");
40          lowV = getTrackbarPos("Low Value", "HSV Tester");
41          highV = getTrackbarPos("High Value", "HSV Tester");
42
```

HSV Tool Code:

- Main loop
  1. Read the trackbar positions and store in variables
  2. Create an variable to store our HSV image [frameHSV]
  3. Convert our input image [frame] from BGR to HSV format
  4. Use inRange() to select pixels only within our HSV range
     - [frameHSV] is both the input and output location

```
43          Mat frameHSV;       // Convert the frame to HSV and apply the limits
44          cvtColor(frame, frameHSV, COLOR_BGR2HSV);
45          inRange(frameHSV, Scalar(lowH, lowS, lowV), Scalar(highH, highS, highV), frameHSV);
```

HSV Tool Code:

- Main loop
  3. Convert our input image [frame] from BGR to HSV format
  4. Use inRange() to select pixels only within our HSV range
     - [frameHSV] is both the input and output location
  5. Create a variable to store our comparison image
  6. Convert our HSV image back to BGR
     - Must be the same format to combine
  7. Concatenate the two images together [frame] and [frameHSV]

```
47          Mat comparison;     // Join the two into a single image
48          cvtColor(frameHSV, frameHSV, COLOR_GRAY2BGR);   // In ra
49          hconcat(frame, frameHSV, comparison);
```

HSV Tool Code:

- Main loop
  5. Create a variable to store our comparison image
  6. Convert our HSV image back to BGR
     - Must be the same format to combine
  7. Concatenate the two images together [frame] and [frameHSV]
  8. Display the resulting image in the window

```
51          imshow("HSV Tester", comparison); //Display the image in the window
```

HSV Tool Code:

- Main loop
  8. Display the resulting image in the window
  9. Wait 1ms for a key press and store the value
  10. If the key is 27 (ESC), then break out of the main loop

```
53          int key = cv::waitKey(1);    // Wait 1ms
54
55          key = (key==255) ? -1 : key;     // Check
56          if (key == 27)
57              break;
58      }
```

HSV Tool Code:

- Main loop
  8. Display the resulting image in the window
  9. Wait 1ms for a key press and store the value
  10. If the key is 27 (ESC), then break out of the main loop
- Close the window and exit the program

```
60      destroyAllWindows();
61
62      return 0;
63  }
```

University of Nottingham
UK | CHINA | MALAYSIA

Pre-session task

Next week

Pre-session Task (next week)

Identify the colour of an object:

- Using what's been shown in this session and an additional handout on Moodle
- Create an OpenCV program that can identify the colour of an object in an image
- 6 sample images are provided:

Red

Blue

Green

Red

Green

Blue

Useful Function

Counting pixels:

- countNonZero()

  Count the pixels in a **greyscale** image that have a non-zero value

Variable to store the number of blue pixels

Function call to count the pixels

```
int bluePixels = 0;
bluePixels = countNonZero(frameHSV);
std::cout << "Blue Pixels = " << bluePixels << std::endl;
```

Print the number of pixels on the terminal

Pre-session Task

Assessment:

- Worth 5% of report 3
- Include as an appendix but **must** also be submitted to Moodle by **9am Monday 9th March**
- Complete as an individual piece of work

- Include:
  - A ZIP of the Code::Blocks project
  - A brief (2 page limit) report illustrating the process followed by your code (e.g. a flowchart)