**Department of Electrical and Computer Systems Engineering**

**Monash University**

**TRC3500 Sensors and Artificial Perception**

**Calculating Blob Statistics**

**Introduction**

In this exercise, you will write a short program in C++ to capture an image from a USB camera, convert the image to grayscale, and apply a threshold to produce a binary image. You will then do a connectivity analysis to discover the blobs in the image. Afterwards, for each blob, you will calculate the 0th, 1st and 2nd order moments of its area and indicate the centre of area and axis of minimum moment of area by displaying them superimposed on the original image.

In this exercise, you will be exposed to a number of introductory concepts in image capture and processing using state-of-the-art software packages that are publicly available.

**Equipment and Software**

The tools that you will require are as follows:

* A USB camera
* A Windows PC
* C++ compiler suite including the OpenCV packages

**Theory**

The unit’s notes provide the background on blob statistics calculations. Make sure that you are fully conversant with this material before you begin.

**Recommended Procedure**

1. Capture a sharp image of the pictures in the example files indicated in the marking guide. If you wish, you can use the starter code provided in Appendix A. (Tip: if you don’t want to print the images, point the webcam at a computer monitor!).
2. Convert the image to grayscale.
3. Convert the image to black and white (binary). To do this, you need to work out a threshold for the image.
4. Do a connectivity analysis to determine the number of blobs in the image.
5. Do the following for each blob:
   1. Calculate and report (in a Windows terminal) its centre of area and its axis of minimum moment of inertia, and
   2. Modify the original image displayed in the image window to indicate the blob centre and axis of minimum moment of inertia (draw a cross to indicate the blob centre and a line through the centre of the cross indicating the axis).
6. Identify similarly-shaped components and label each component type with a category letter. Total the number of components in each category and print the result into the console window.

**Appendix A: OpenCV/C++ Starter Code**

*This code is available for download from the* [*TRC3500 Github repository*](https://github.com/TRC3500/Project1-blobs)

#include <opencv2/opencv.hpp>

#include <iostream>

/\* Ahmet 13-Feb-2024

Sample program for capturing and displaying a frame, and printing the pixel

values at a given position.

To compile, copy the following lines into a file, say "compile.bat", and

save it in the directory where this file is stored. Then, run it in a Windows terminal.

g++ -o opencv\_cam opencv\_cam.cpp -std=c++17^

-I "C:\msys64\mingw64\include\opencv4"^

-L "C:\msys64\mingw64\bin"^

-lopencv\_core-409 -lopencv\_highgui-409 -lopencv\_imgcodecs-409^

-lopencv\_imgproc-409 -lopencv\_videoio-409

\*/

int main() {

cv::VideoCapture cap(0); // On my laptop "0" is the built-in camera.

if (!cap.isOpened()) {

std::cerr << "Error opening the camera!" << std::endl;

return -1;

}

int x = 100; int y = 200; // I am interested in this pixel

cv::Mat frame;

cap >> frame;

if (frame.empty()) {

std::cerr << "No frame captured?" << std::endl;

return -1;

}

cv::imshow("Frame", frame);

cv::Vec3b pixel = frame.at<cv::Vec3b>(y, x);

std::cout << "Pixel values at (" << x << ", " << y << "): ";

std::cout << "B: " << static\_cast<int>(pixel[0]) << " ";

std::cout << "G: " << static\_cast<int>(pixel[1]) << " ";

std::cout << "R: " << static\_cast<int>(pixel[2]) << std::endl;

cv::waitKey(0);

cap.release();

cv::destroyAllWindows();

return 0;

} // main()