CPE-462
Edge Detection in Selected Region

Zikang Sheng
Daren Tay
May 13th, 2021

Pledge: *I pledge my honor that I have abided by the Stevens Honor System.*

Table of contents

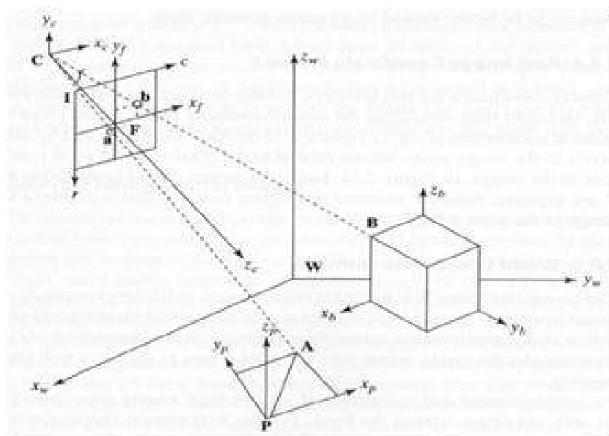
1. Abstract

Due to the nature and position of most security cameras, the resulting security footage is mostly slanted, top-down views which are hard to identify specific parts with human eyes. The objective of this project is to clarify words or specific parts of a photo by using the transforming perspective function of OpenCV and edge detection to make an unrecognizable word or part readable by humans. The project is written in C++ code with the OpenCV library. The entire clarification process is achieved by selecting the area to apply the perspective transform, and then smoothing out the result by performing edge detection and global thresholding. The results are fairly consistent with converting blurry, miniature, or tilted words into identifiable images. The program is able to clarify and edit shots of cars from security footage to make necessary objects such as car plates and luggages easier to identify for users and more.

2. Method

2.1. Perspective Transform

In many real word images, the shooting objects are not necessarily facing the camera. Perspective transform is to transfer a selected region in an image from one state to another. It deals with the conversion of a 3d image into a 2d image.



In this project, the Perspective Transformation functions which are inbuilt in the OpenCV library are used to change the perspective of a given image for getting better insights about the required information.

Mat getPerspectiveTransform(InputArray src, InputArray dst)

- The function calculates a perspective transform from four pairs of the corresponding points.

void warpPerspective(InputArray src, OutputArray dst, InputArray M, Size dsize)

- The function applies a perspective transformation to an image.

2.2. Edge Detection

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness.

A Prewitt edge detector is used in this project. It is first applied to the original image $x[n_1, n_2]$ which will produce two filtered images $G_1(x[n_1, n_2])$ and $G_2(x[n_1, n_2])$.

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Prewitt edge detector

And then combine two filtered images using absolute sum.

$$y[n_1, n_2] = |G_1(x[n_1, n_2])| + |G_2(x[n_1, n_2])|$$

Lastly, use the threshold T from global thresholding function to generate a binary edge image.

$$z[n_1, n_2] = \begin{cases} 255 \ (white) & if \quad y[n_1, n_2] \geq T \\ 0 \ (black) & if \quad y[n_1, n_2] < T \end{cases}$$

2.3. Global Thresholding

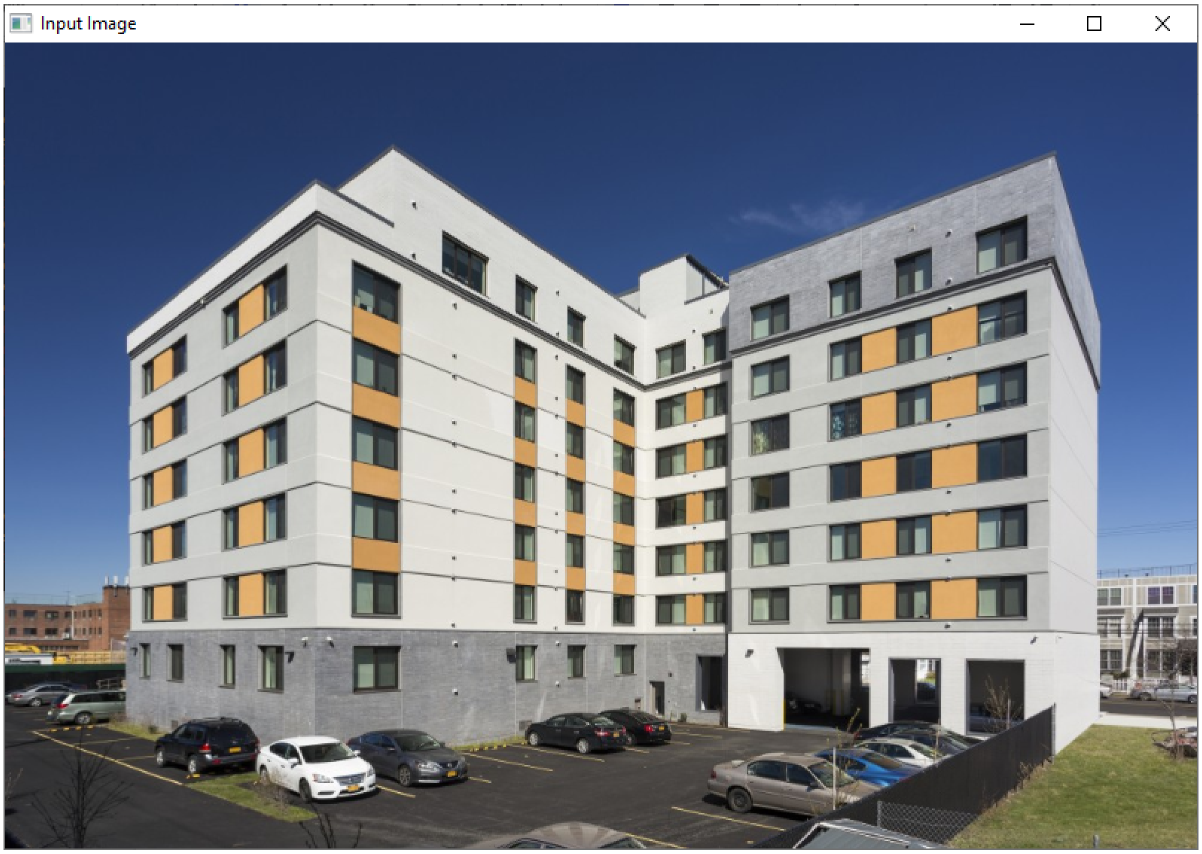
Global thresholding is used when a chosen threshold value depends only on gray-level values and relates to the characteristics of pixels. Interactive global threshold algorithm requires an arbitrary initial T_i. And then the image is classified into two pixel groups (intensity greater than T and intensity less than T) with average intensity of μ_1 and μ_2. New T is the middle point of μ_1 and μ_2. Repeat the steps above and keep updating Ti until $|T - T_i|$ is less than a preset value. The result of thresholding is a binary image, where pixels with intensity value of 255 (white) correspond to objects, whereas pixels with value 0 (black) correspond to the background.

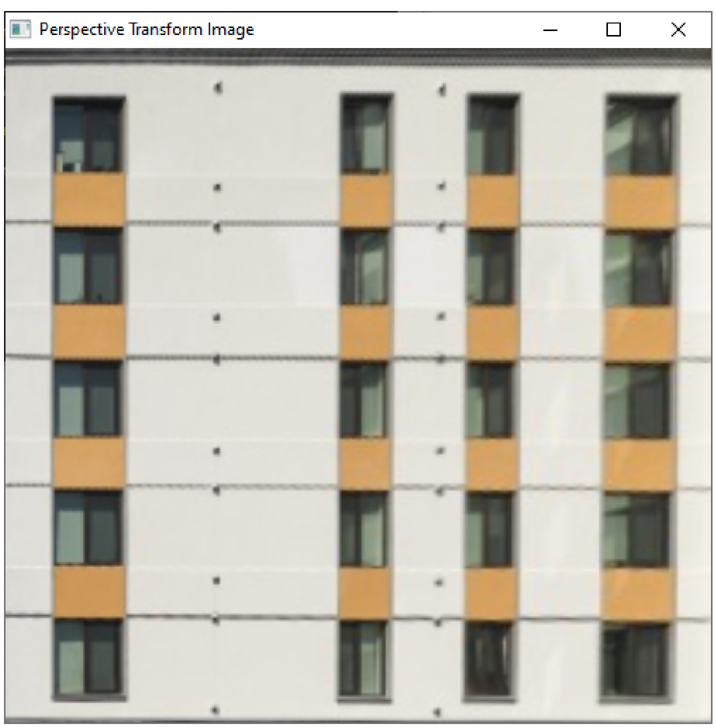
3. Results

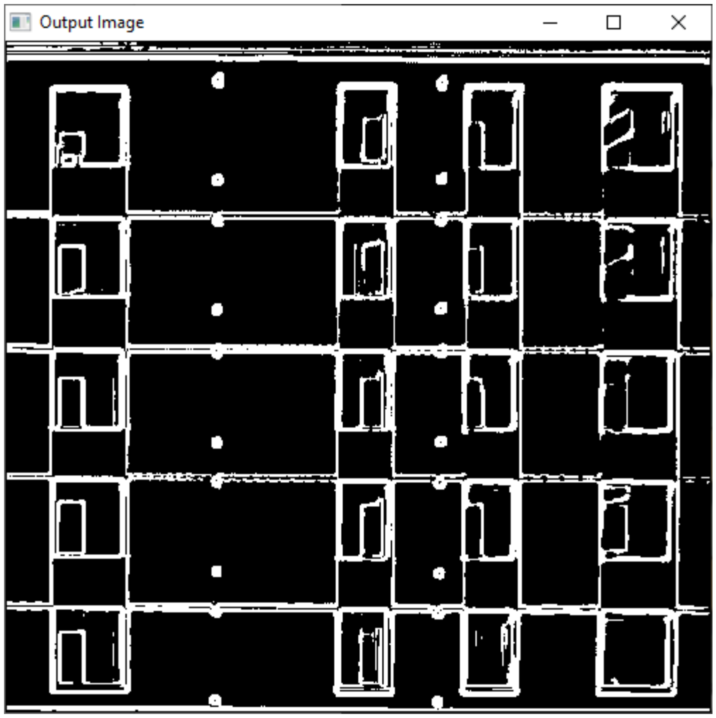

Input Image
ZX
IND
HR26DQ5551
GoMechanic


Perspective Transform Image
IND
HR26DQ5551


Output Image
IND
HR26DQ5551


Input Image


Perspective Transform Image


Output Image

4. Discussion

The results of the project are within the group's expectations with clear cut binary images that highlighted edges and facing the camera. However, when the original object has a large angle difference with the perspective, the resulting image will have a bad width-length ratio depending on the angle. After testing the group discovered that the maximum angle to have a clear outcome would be 30 degrees. The program is able to take parts of an unidentifiable image into a more readable outcome. This project can have a wide variety of uses, and be applied to numerous occasions with security cameras involved or set as a module for most image-altering applications. Some real-life examples would be identifying car plates in security footage, or tracking QR codes on luggages in airport, Implementations of Artificial Intelligence might be able to mitigate the setback of the necessity of manually selecting the area to perform perspective transformation yet it is not a problem if implemented into image-altering applications.

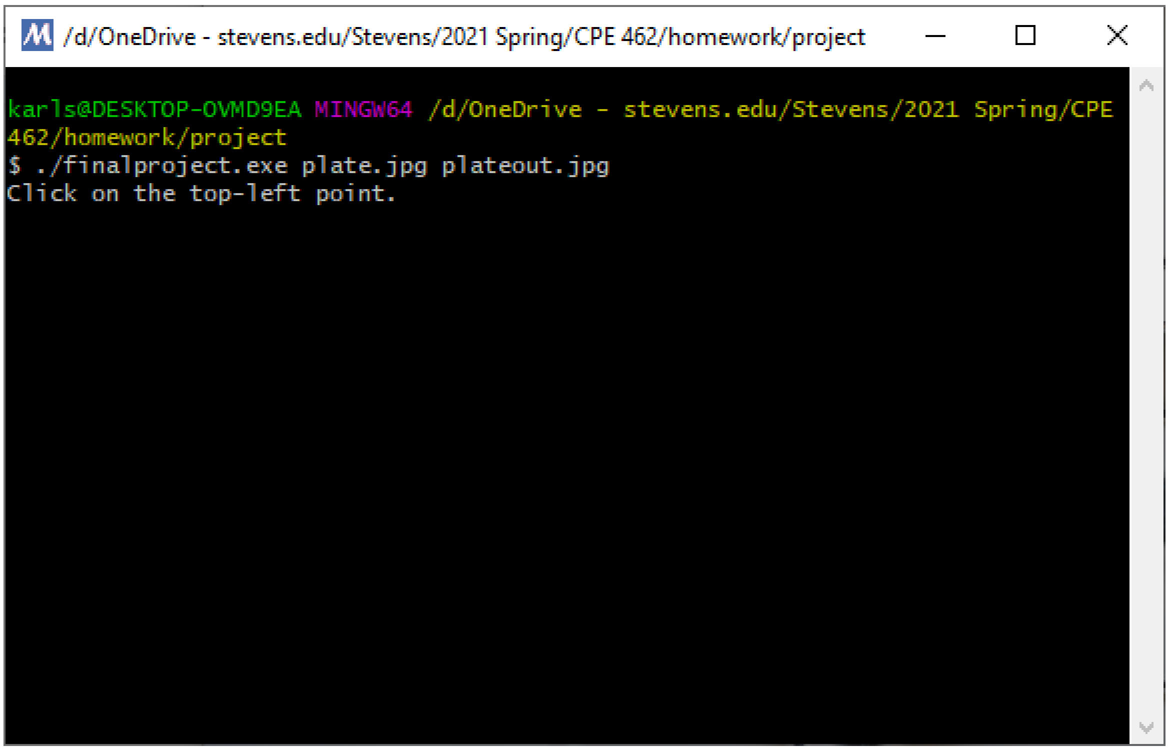
(Example of bad width-length ratio when encountering a large angle)

5. Instruction

1. Compile *FinalProject.cpp.*

2. Run *finalproject.exe* in the command line.

    - Usage: ./finalproject.exe <input filename> <output filename>



3. Follow the instructions in the command window.

    - Select the region in the original image that you want to process.
    - Start with the top-left corner and click on the four corners of the region clockwise.


```
/d/OneDrive - stevens.edu/Stevens/2021 Spring/CPE 462/homework/project

karls@DESKTOP-OVMD9EA MINGW64 /d/OneDrive - stevens.edu/Stevens/2021 Spring/CPE
462/homework/project
$ ./finalproject.exe plate.jpg plateout.jpg
Click on the top-left point.
Points position: (267, 223)
Click on the top-right point.
Points position: (706, 183)
Click on the bottom-right point.
Points position: (731, 290)
Click on the bottom-left point.
Points position: (295, 373)
```

4. Perspective transform image and grayscale image will be displayed on the screen.

5. Output image will be saved to the current folder.

/d/OneDrive - stevens.edu/Stevens/2021 Spring/CPE 462/homework/project

```
karls@DESKTOP-OVMD9EA MINGW64 /d/OneDrive - stevens.edu/Stevens/2021 Spring/CPE
462/homework/project
$ ./finalproject.exe plate.jpg plateout.jpg
Click on the top-left point.
Points position: (267, 223)
Click on the top-right point.
Points position: (706, 183)
Click on the bottom-right point.
Points position: (731, 290)
Click on the bottom-left point.
Points position: (295, 373)
Updated threshold T: 163.53, previous threshold T: 127.50
Updated threshold T: 170.07, previous threshold T: 163.53
Updated threshold T: 173.20, previous threshold T: 170.07
Updated threshold T: 175.05, previous threshold T: 173.20
Updated threshold T: 176.34, previous threshold T: 175.05
Updated threshold T: 177.26, previous threshold T: 176.34

karls@DESKTOP-OVMD9EA MINGW64 /d/OneDrive - stevens.edu/Stevens/2021 Spring/CPE
462/homework/project
$
```

6. Contribution

Zikang Sheng	**Daren Tay**
Coding	Report
Report	Design

7. Source Code

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/opencv.hpp>
#include <cmath>
#include <iostream>
#include <vector>

using namespace std;
using namespace cv;

vector<Point2f> srcPoints;
vector<Point2f> dstPoints;
int   j, k, width, height;
int ** img_in, ** img_out;
Mat image_in, image_in_copy;
double ratio_copy, ratio_trans;

void mousePoints(int event, int x, int y, int flag, void* userdata) {
    if (event==EVENT_LBUTTONUP) {
        circle(image_in_copy, Point2f(x, y), 8, Scalar(0, 255, 0),
FILLED);
        imshow("Input Image", image_in_copy);
        cout << "Points position: (" << x << ", " << y << ")" << endl;
        srcPoints.push_back(Point2f(x, y));
        if (srcPoints.size() == 1) {
            cout << "Click on the top-right point." << endl;
        }
        else if (srcPoints.size() == 2) {
            cout << "Click on the bottom-right point." << endl;
        }
        else if (srcPoints.size() == 3) {
            cout << "Click on the bottom-left point." << endl;
        }
    }
    else if (srcPoints.size() == 4) {
        destroyWindow("Input Image");
    }
}
```

```
void set_srcPoints() {
    for (int i=0; i<4; i++) {
        srcPoints.at(i).x /= ratio_copy;
        srcPoints.at(i).y /= ratio_copy;
    }
}

void set_dstPoints() {
    double x1, x2, y1, y2;
    x1 = sqrt(pow(srcPoints.at(0).x - srcPoints.at(1).x, 2) +
pow(srcPoints.at(0).y - srcPoints.at(1).y, 2));
    x2 = sqrt(pow(srcPoints.at(2).x - srcPoints.at(3).x, 2) +
pow(srcPoints.at(2).y - srcPoints.at(3).y, 2));
    y1 = sqrt(pow(srcPoints.at(0).x - srcPoints.at(3).x, 2) +
pow(srcPoints.at(0).y - srcPoints.at(3).y, 2));
    y2 = sqrt(pow(srcPoints.at(1).x - srcPoints.at(2).x, 2) +
pow(srcPoints.at(1).y - srcPoints.at(2).y, 2));
    width = (int)((x1 + x2) / 2);
    height = (int)((y1 + y2) / 2);
    dstPoints.push_back(Point2f(0, 0));
    dstPoints.push_back(Point2f(width, 0));
    dstPoints.push_back(Point2f(width, height));
    dstPoints.push_back(Point2f(0, height));
}

int global_threshold() {
    double old_T = 255.0/2;
    double new_T = 0.0;
    double delta_T= 0.0;
    double sum1 = 0.0, sum2 = 0.0;
    double count1 = 0.0, count2 = 0.0;
    int a = 1;

    do {
        for (j=0; j<height; j++) {
            for (k=0; k<width; k++) {
                if (img_out[j][k] < old_T) {
                    sum1 += img_out[j][k];
                    count1 += 1;
                }
```

```
                else {
                    sum2 += img_out[j][k];
                    count2 += 1;
                }
            }
        }

        if (count1 == 0) {
            new_T = (old_T / 2 + sum2 / count2) /2;
        }
        else if (count2 == 0) {
            new_T = (sum1 / count1 + old_T / 2) / 2;
        }
        else {
            new_T = (sum1 / count1 + sum2 / count2) / 2;
        }
        printf("Updated threshold T: %.2f, previous threshold T: %.2f\n",
new_T, old_T);

        delta_T = abs(new_T - old_T);
        old_T = new_T;
    } while (delta_T >= a);

    return (int)new_T;
}

void edge_detection() {
    int num = 3;
    int p1[num][num] = {{-1, -1, -1}, {0, 0, 0}, {1, 1, 1}};
    int p2[num][num] = {{-1, 0, 1}, {-1, 0, 1}, {-1, 0, 1}};

    for (j=0; j<height; j++) {
        for (k=0; k<width; k++) {
            if (j==0 || k==0 || j==height-1 || k==width-1) {
                img_out[j][k] = 0;
            }
            else {
                int Gx = 0, Gy = 0;
                for (int m=0; m<num; m++) {
                    for (int n=0; n<num; n++) {
```

```
                        Gx += img_in[j-1+m][k-1+n] * p1[m][n];
                        Gy += img_in[j-1+m][k-1+n] * p2[m][n];
                    }
                }
                img_out[j][k] = abs(Gx) + abs(Gy);
            }
        }
    }

    int t = global_threshold();
    for (j=0; j<height; j++) {
        for (k=0; k<width; k++) {
            if (img_out[j][k] >= t){
                img_out[j][k] = 255;
            }
            else {
                img_out[j][k] = 0;
            }
        }
    }
}

bool initialize_img() {
    img_in = (int**) calloc(height, sizeof(int*));
    if(!img_in)
    {
        return(false);
    }

    img_out = (int**) calloc(height, sizeof(int*));
    if(!img_out)
    {
        return(false);
    }

    for (j=0; j<height; j++)
    {
        img_in[j] = (int *) calloc(width, sizeof(int));
        if(!img_in[j])
        {
```

```
            return(false);
        }

        img_out[j] = (int *) calloc(width, sizeof(int));
        if(!img_out[j])
        {
            return(false);
        }
    }
    return true;
}

void delete_img() {
    for (j=0; j<height; j++)
    {
        free(img_in[j]);
        free(img_out[j]);
    }
    free(img_in);
    free(img_out);
}

int main(int argc, char *argv[]) {

    if(argc != 3) {
        cerr << "ERROR: Insufficient parameters!" << endl;
        return 1;
    }

/**********************************************************************/
/*  Read image                                                        */
/**********************************************************************/
    image_in = imread(argv[1]);
    image_in_copy = imread(argv[1]);
    ratio_copy = min(800.0/image_in.rows, 800.0/image_in.cols);
    resize(image_in_copy, image_in_copy, Size(0,0), ratio_copy,
ratio_copy);
    namedWindow("Input Image");
    imshow("Input Image", image_in_copy);
    setMouseCallback("Input Image", mousePoints, NULL);
```

```
    cout << "Click on the top-left point." << endl;
    waitKey(0);
    set_dstPoints();
    set_srcPoints();

/**********************************************************************/
/*  Perspective transform                                             */
/**********************************************************************/

    Mat image_trans(Size(width, height), CV_64FC1);
    Mat transMat = getPerspectiveTransform(srcPoints, dstPoints);
    warpPerspective(image_in, image_trans, transMat, image_trans.size());

    if (max(width, height) < 500) {
        ratio_trans = min(500.0/image_trans.rows, 500.0/image_trans.cols);
        resize(image_trans, image_trans, Size(0,0), ratio_trans,
ratio_trans);
        width = image_trans.cols;
        height = image_trans.rows;
    }

    namedWindow("Perspective Transform Image");
    imshow("Perspective Transform Image", image_trans);
    waitKey(0);
    destroyWindow("Perspective Transform Image");

/**********************************************************************/
/*  Convert to 2D array                                               */
/**********************************************************************/
    Mat_<uchar> image_gray(width, height);
    cvtColor(image_trans, image_gray, COLOR_BGR2GRAY);
    namedWindow("Grayscale Image");
    imshow("Grayscale Image", image_gray);
    waitKey(0);
    destroyWindow("Grayscale Image");

    while (!initialize_img()) {
        cerr << "Error: Can't allocate memory!" << endl;
        return 1;
    }
```

```
    for (j=0; j<height; j++) {
        for (k=0; k<width; k++) {
            img_in[j][k] = image_gray(j,k);
        }
    }

/**********************************************************************/
/*  Image processing                                                  */
/**********************************************************************/

    edge_detection();

/**********************************************************************/
/*  Save image                                                        */
/**********************************************************************/
    Mat_<uchar> image_out(height, width);
    for (j=0; j<height; j++) {
        for (k=0; k<width; k++) {
            image_out(j, k) = img_out[j][k];
        }
    }

    namedWindow("Output Image");
    imshow("Output Image", image_out);
    waitKey(0);
    destroyWindow("Output Image");

    bool isSuccess = imwrite(argv[2], image_out);
    if (!isSuccess) {
        cout << "Failed to save the image" << endl;
        return 1;
    }

    delete_img();
    return 0;
}
```

https://github.com/karlsheng99/CPE462_ImageProcessing

8. Reference

“Image Transformations - Computer Vision and OpenCV C++,” *YouTube*, 13-Oct-2020. [Online]. Available: https://www.youtube.com/watch?v=nBB2L419EfI&t=1546s. [Accessed: 13-May-2021].

K. A. Abera, K. N. Manahiloh, and M. M. Nejad, “The effectiveness of global thresholding techniques in segmenting two-phase porous media,” *Construction and Building Materials*, 17-Mar-2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950061817304191. [Accessed: 13-May-2021].

niconielsen32, “niconielsen32/ComputerVision,” *GitHub*, 15-Oct-2020. [Online]. Available: https://github.com/niconielsen32/ComputerVision/blob/master/mouse.h. [Accessed: 13-May-2021].

Perspective Transform. [Online]. Available: http://opencvexamples.blogspot.com/2014/01/perspective-transform.html. [Accessed: 13-May-2021].

“Perspective Transformation,” *Tutorialspoint*. [Online]. Available: https://www.tutorialspoint.com/dip/perspective_transformation.htm. [Accessed: 13-May-2021].