Goncalo Mendes

Computer Vision

Assignment 5

In this assignment, we applied what we went over in class about Image Registration in an example with 2-D panels with points on the same plane that represented a shape.

Firstly, the goal was to derive the math to develop a matrix equation for minimizing the cost in image registration (without shear) when the correspondences between the points in two images I1 and I2 are known.

To do that, we used the Error equation shown below and derived the expressions in respect to each one of the model parameters and equaled that to zero.

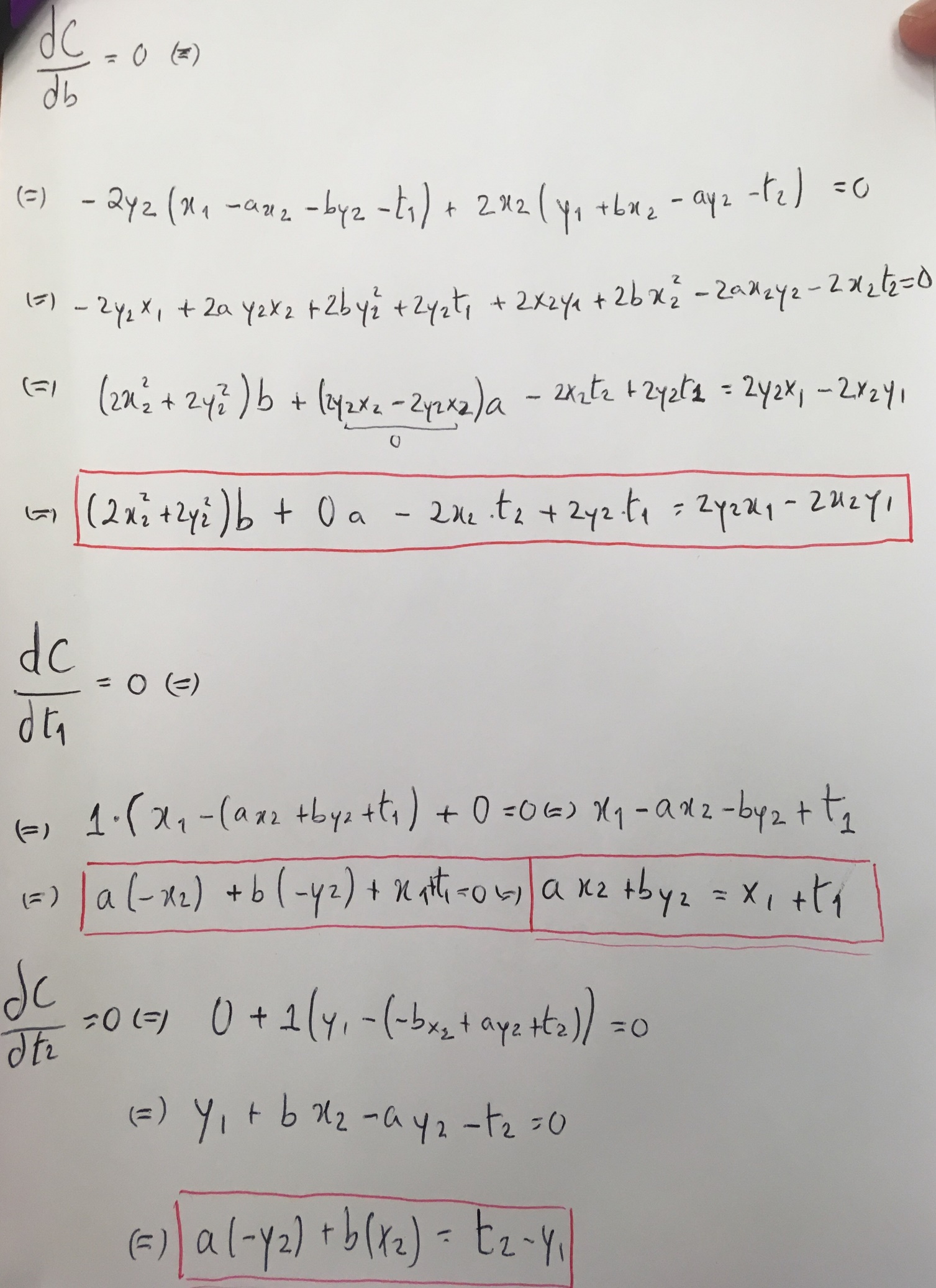
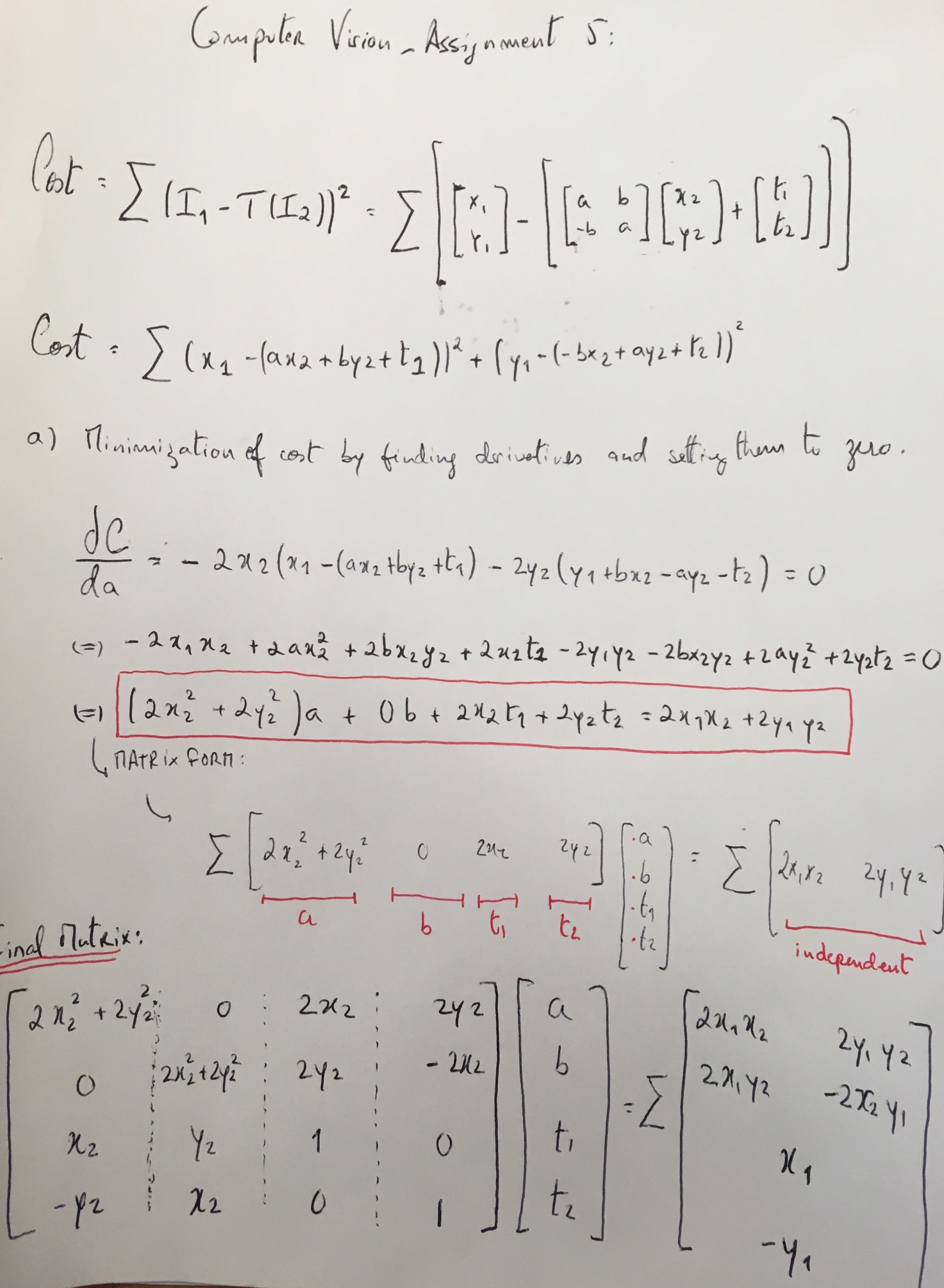
To code the math below, we created a transformation class that saves the parameters and an apply transformation method that basically multiplies the input matrix by the model parameters to obtain the new shape.

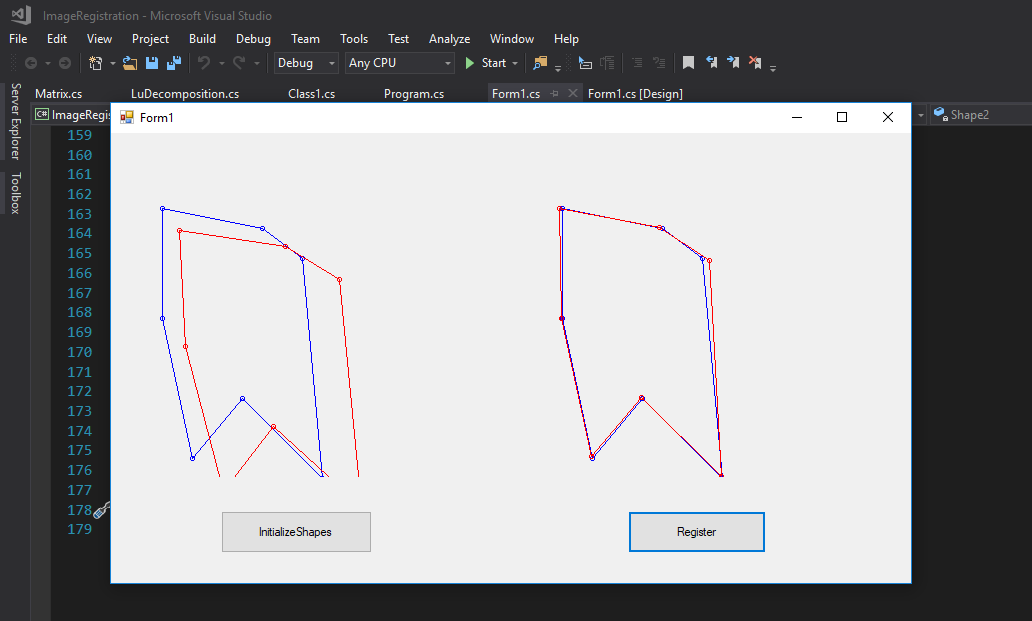
An idea gotten from the base code I used, was to do this process point by point of the shape, which in a picture would be going pixel by pixel which works easily. For this, one basically accesses the points inside each shape, and apply the transformation.

An interesting detail is that in order to fill both shapes (original and transformed) with data, we have to input some random parameters into the transformation class just so we can apply it and have the second shape be generated and later altered using the linear least squares optimization of the cost function.

Then the transformations were drawn by having each shape at a different color, so we could see the image registration product.

Below is the math derived, the source code and the screenshots of the resulting portrait of the shapes before and after applying the image registration algorithm.





using Mapack;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

List<Point> Shape1 = new List<Point>();

List<Point> Shape2 = new List<Point>();

private void btnInitializeShapes\_Click(object sender, EventArgs e)

{

}

private void InitializeShapes\_Click(object sender, EventArgs e)

{

Shape1.Clear();

Shape2.Clear();

Point p1a = new Point(20, 30);

Point p2a = new Point(120, 50);

Point p3a = new Point(160, 80);

Point p4a = new Point(180, 300);

Point p5a = new Point(100, 220);

Point p6a = new Point(50, 280);

Point p7a = new Point(20, 140);

Shape1.Add(p1a); Shape1.Add(p2a);

Shape1.Add(p3a); Shape1.Add(p4a);

Shape1.Add(p5a); Shape1.Add(p6a);

Shape1.Add(p7a);

Transformation T2 = new Transformation();

T2.A = 1.05; T2.B = 0.05; T2.T1 = 15; T2.T2 = 22;

for (int i = 0; i < Shape1.Count; i++)

{

Point temp = new Point();

temp = ApplyTransformation(T2, Shape1[i]);

Shape2.Add(temp);

}

Shape2[2] = new Point(Shape2[2].X + 10, Shape2[2].Y + 3); // change one point

Pen pBlue = new Pen(Brushes.Blue, 1);

Pen pRed = new Pen(Brushes.Red, 1);

Graphics g = panShape1.CreateGraphics();

DisplayShape(Shape1, pBlue, g);

DisplayShape(Shape2, pRed, g);

}

void DisplayShape(List<Point> Shp, Pen pen, Graphics g)

{

Point? prevPoint = null; // nullable

foreach (Point pt in Shp)

{

g.DrawEllipse(pen, new Rectangle(pt.X - 2, pt.Y - 2, 4, 4));

if (prevPoint != null)

g.DrawLine(pen, (Point)prevPoint, pt);

prevPoint = pt;

}

g.DrawLine(pen, Shp[0], Shp[Shp.Count - 1]);

}

private void Form1\_Load(object sender, EventArgs e)

{

}

Point ApplyTransformation(Transformation Trf, Point p)

{

//List<Point> shifted = new List<Point>();

Matrix input = new Matrix(2, 1); //contains x1,y1, point from original shape

Matrix coef = new Matrix(2, 2); //contains matrix with a and b coefficients formulated

Matrix t = new Matrix(2, 1); //contains t1, t2, components.

Matrix result = new Matrix(2, 1); //contains x2, y2 - the shifted result point for every case

input[0,0] = p.X;

input[1,0] = p.Y;

coef[0,0] = Trf.A;

coef[0,1] = Trf.B;

coef[1,0] = Trf.B\*(-1);

coef[1,1] = Trf.A;

t[0,0] = Trf.T1;

t[1,0] = Trf.T2;

result = coef\*input + t; // Matrix Equation to apply transformation

Point shifted = new Point ((int)result[0,0], (int)result[1,0]);

return shifted;

}

void LeastSquaresOptimization(List<Point> Shp1, List<Point> Shp2, Transformation T2)

{

Matrix Sum = new Matrix(4, 4); //contains components of the calculated derivatives

Matrix z = new Matrix(4, 1); //contains result of loose derivatives.

for (int i = 0; i < Shp1.Count; i++)

{

int x1 = Shp1[i].X;

int x2 = Shp2[i].X;

int y1 = Shp1[i].Y;

int y2 = Shp2[i].Y;

Sum[0, 0] += 2 \* x2\*x2 + 2 \* y2\*y2;

Sum[1, 0] += 0;

Sum[2, 0] += 2 \* x2;

Sum[3, 0] += 2 \* y2;

Sum[0, 1] += 0;

Sum[1, 1] += 2 \* x2 \* x2 + 2 \* y2\*y2;

Sum[2, 1] += 2 \* y2;

Sum[3, 1] += 2 \* x2 \* (-1);

Sum[0, 2] += 2 \* x2;

Sum[1, 2] += 2 \* y2;

Sum[2, 2] += 2;

Sum[3, 2] += 0;

Sum[0, 3] += 2 \* y2;

Sum[1, 3] += 2 \* x2 \* (-1);

Sum[2, 3] += 0;

Sum[3, 3] += 2;

z[0, 0] += 2 \* x2 \* x1 + 2 \* y1 \* y2;

z[1, 0] += 2 \* x1 \* y2 - 2 \* x2 \* y1;

z[2, 0] += 2 \* x1;

z[3, 0] += 2 \* y1;

}

Matrix SumInv = Sum.Inverse; //inverse of Sum (A)

Matrix result = SumInv \* z; //contains result of equation with model parameters a,b,t1,t2

T2.A = result[0, 0];

T2.B = result[1, 0];

T2.T1 = result[2, 0];

T2.T2 = result[3, 0];

}

private void button1\_Click(object sender, EventArgs e)

{

Transformation Tf = new Transformation();

List<Point> Trf = new List<Point>();

LeastSquaresOptimization(Shape1, Shape2, Tf);

for (int i = 0; i < Shape1.Count; i++)

{

Point temp = new Point();

temp = ApplyTransformation(Tf, Shape2[i]);

Trf.Add(temp);

}

Pen pBlue = new Pen(Brushes.Blue, 1);

Pen pRed = new Pen(Brushes.Red, 1);

Graphics g = panShape2.CreateGraphics();

DisplayShape(Shape1, pBlue, g);

DisplayShape(Trf, pRed, g);

}

}

}

Conclusion

The Mathematics background was reviewed derived and understood through this assignment for future applications and it would be interesting to also have an example applied on a picture in the future, as there are certainly limitations on the algorithm coded for the points in the shapes. An example on this was provided in class but since it is a step towards the final goal of computer vision applications, it should be solidified and practiced, which I will try to.