ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №1

За курсом «Обробка та розпізнавання зображень»

Студента групи ПА-19-2

Ільяшенко Єгора Віталійовича

Кафедра комп’ютерних технологій, ДНУ

**Перетворення Хатчінсона**

Трикутник Серпинського

A picture containing text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Папороть Барнслі

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generated

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_1

{

internal class Hatchinson

{

public static void Calculate(ref List<(int, int)> input, int iterations, int count\_of\_transformations, System.Windows.Forms.ListBox loger, Action callback)

{

List<(int, int)> output;

List<AffinTransformation> transformations = new List<AffinTransformation>();

Random rnd = new Random();

loger.Items.Add("TransformationsCreation");

loger.Refresh();

float[,] matrix= new float[2,2];

float[] vector = new float[2];

for (int i = 0; i < count\_of\_transformations; i++)

{

//matrix = new float[2, 2]

//{

// { (float)rnd.NextDouble() \* 2f - 1,(float)rnd.NextDouble() \* 2f - 1},

// { (float)rnd.NextDouble() \* 2f - 1,(float)rnd.NextDouble() \* 2f - 1}

//};

//Tserpynski

{

{

if (i == 0)

matrix = new float[2, 2]

{

{ 0.5f, 0f },

{ 0f, 0.5f }

};

if (i == 1)

matrix = new float[2, 2]

{

{ 0.5f, 0f },

{ 0f, 0.5f }

};

if (i == 2)

matrix = new float[2, 2]

{

{ 0.5f, 0f },

{ 0f, 0.5f }

};

}

{

if (i == 0)

vector = new float[2] { 0f, 100f };

if (i == 1)

vector = new float[2] { 30f, -20f };

if (i == 2)

vector = new float[2] { -30f, -20f };

}

}

/\*

//Папоротник

{

{

if (i == 0)

matrix = new float[2, 2]

{

{ 0f, 0f },

{ 0f, 0.16f }

};

if (i == 1)

matrix = new float[2, 2]

{

{ 0.85f, 0.04f },

{ -0.04f, 0.85f }

};

if (i == 2)

matrix = new float[2, 2]

{

{ 0.2f, -0.26f },

{ 0.23f, 0.22f }

};

if (i == 3)

matrix = new float[2, 2]

{

{ -0.15f, 0.28f },

{ 0.26f, 0.24f }

};

}

{

if (i == 0)

vector = new float[2] { 0f, 0f };

if (i == 1)

vector = new float[2] { 0f, 60f };

if (i == 2)

vector = new float[2] { 0f, 60f };

if (i == 3)

vector = new float[2] { 0f, 44f };

}

}

\*/

//vector = new float[2]

// {

// (float)rnd.NextDouble(), (float)rnd.NextDouble()

// };

transformations.Add(new AffinTransformation(matrix, vector));

}

for (int i = 0; i < iterations; i++)

{

loger.Items.Add($"Iteration: {i + 1}");

loger.Refresh();

output = new List<(int, int)>();

for (int coord = 0; coord < input.Count; coord++)

{

//for (int tr = 0; tr < transformations.Count; tr++)

//{

// int new\_x = (int)(

// transformations[tr].Matrix[0, 0] \* input[coord].Item1 +

// transformations[tr].Matrix[0, 1] \* input[coord].Item2 +

// transformations[tr].Vector[0]);

// int new\_y = (int)(

// transformations[tr].Matrix[1, 0] \* input[coord].Item1 +

// transformations[tr].Matrix[1, 1] \* input[coord].Item2 +

// transformations[tr].Vector[1]);

// output.Add((new\_x, new\_y));

//}

{

int transformationIndex = (int)(rnd.NextDouble() \* transformations.Count);

int new\_x = (int)(

transformations[transformationIndex].Matrix[0, 0] \* input[coord].Item1 +

transformations[transformationIndex].Matrix[0, 1] \* input[coord].Item2 +

transformations[transformationIndex].Vector[0]);

int new\_y = (int)(

transformations[transformationIndex].Matrix[1, 0] \* input[coord].Item1 +

transformations[transformationIndex].Matrix[1, 1] \* input[coord].Item2 +

transformations[transformationIndex].Vector[1]);

output.Add((new\_x, new\_y));

}

}

input = new List<(int, int)>(output);

}

callback?.Invoke();

}

}

internal class AffinTransformation

{

float[,] matrix;

float[] vector;

public float[,] Matrix => matrix;

public float[] Vector => vector;

public AffinTransformation(float[,] matrix, float[] vector)

{

this.matrix = matrix;

this.vector = vector;

}

}

}