**Hausdorff v.1.1**

**Matvei Slavenko**

Charles University in Prague

Faculty of Mathematics and Physics

Obecná matematika, the second semester

NMIN102 Programming II

Summer semester, school year 2016/2017

# Introduction

This programme calculates the Hausdorff distance between two convex polygons. Apart from the actual distance, it also calculates the vectors on which the distance is reached and determines whether the current mutual position of the polygons is optimal or not.

The programme has a simple command-line user interface. The user can set the input and output streams at his/her convenience: the programme could both use files and standard input-output system. The list of commands and their detailed descriptions are available in the internal help module of the programme.

# General Information

Let *A* and *B* be polygons in *R2*. Let us assume that ‘polygon’ means its border with its internal area. These facts are assumed throughout the whole section II.

## Hausdorff deviation and the vectors on which it is reached

The value

is called the *Hausdorff deviation* of the polygon A from the polygon B.

The following example illustrates the idea of the Hausdorff deviation. Let *x* be the point of the polygon *A* that lies as far from the polygon *B* as possible. Then the distance between *x* and *B* is exactly the *Hausdorff deviation* of the polygon *A* from the polygon *B*.

Let and . Let . Then we say that the Hausdorff deviation between *A* and *B* is *reached on the vector* .

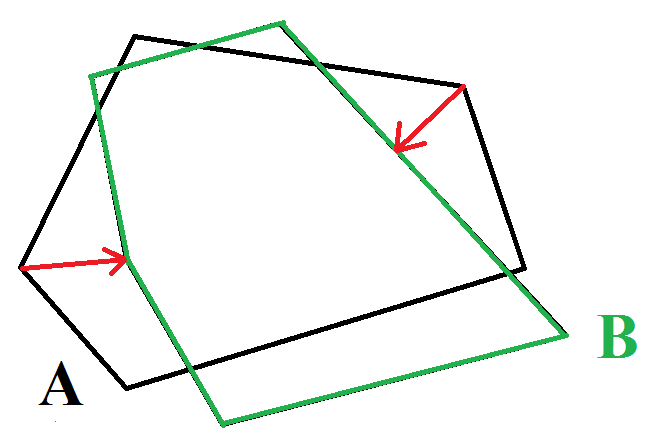


Illustration . The Hausdorff deviation of A from B is reached on the red vectors

## Hausdorff distance and the vectors on which it is reached

The *Hausdorff distance* between *A* and *B* is defined by the following formula:

Let *a* is a point of the polygon *A*, *b* is a polygon of *B*.

Let

Then we say that the vector is one of the *vectors*, on which the Hausdorff distance between *A* and *B* is reached.

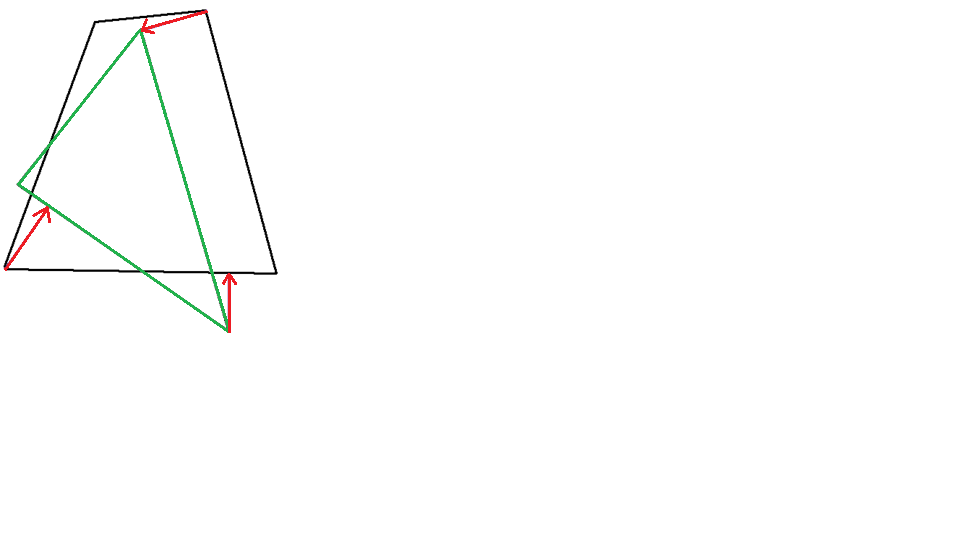


Illustration . The Hausdorff distance between the polygons is reached on the red vectors

## Optimality of the mutual position

Let us define the following function:

Writing *B+x*, we mean the parallel translation of the polygon *B*. We say that the current mutual *position* of the polygons *A* and *B* is *optimal* from the point of the view of the Hausdorff distance, if *(0; 0)* is a global minimum of the function *F(x)*.

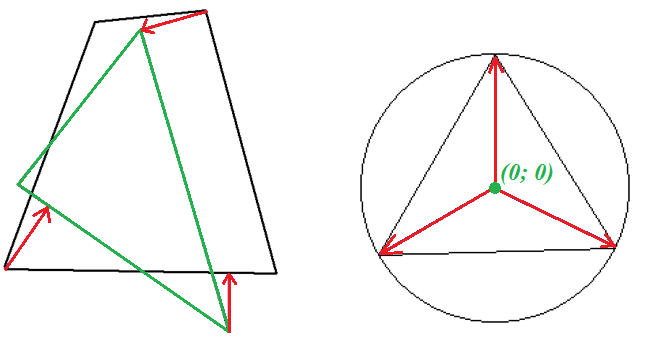
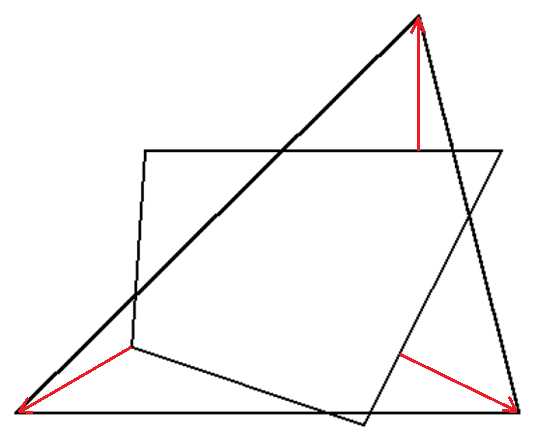


Illustration . The left picture illustrates two polygons and the vectors, on which the Hausdorff distance is reached. The right picture illustrates the *(0; 0)* vector and the convex polygon constructed from the red vectors. *(0; 0)* belongs to the polygon, the position is optimal.

# Actuality and Relevance

The Hausdorff distance is a popular metrics for comparing different subsets in metrical spaces. However, the problem solved by this programme was stated in the scope of the optimal control theory and theory of differential games. In particular, it is used for approximation of attainability domains with convex polyhedrons.

# Data Structures Used

The programme broadly uses the Linked List data structure. In the programme, all polygons are represented as a sequence of their vertices, stored in linked lists.

The data structure’s implementation and the procedures implemented allow turning each linked list to a ring buffer or a priority queue.

# General Ideas of Algorithm

The programme is solving two major tasks. The following section explains the general ideas that are used to solve them.

## Calculation of the vectors on which the Hausdorff deviation is reached

The following fact, proved in [1], allows us to calculate the vectors quite easily.

Let is a vector, on which the Hausdorff deviation of *A* from *B* is reached. Then *a* is a vertex of *A*.

This fact is a basis for the following way of calculating the vectors.

To calculate the vectors on which the Hausdorff deviation between *A* and *B* is reached:

1. For each *a* being a vertex of *A*, calculate the vector, on which the distance between *a* and *B* is reached
2. From the vectors calculated on the first step, select the longest.

## Calculation of the vectors on which the Hausdorff distance is reached

We already know how to calculate the deviation of *A* from *B*, and vice versa. Knowing the solutions of those two tasks, the set of the vectors on which the Hausdorff distance is reached is calculated the following way:

1. Calculate the vectors, on which the Hausdorff deviation of *A* from *B* is reached. Store them to a set *AB*
2. Calculate the vectors, on which the Hausdorff deviation of *B* from *A* is reached. Store them to a set *BA*
3. Swap the vectors in *BA*
4. Join *AB* and *BA*. The resulting set is an answer.

## Defining whether the current position is optimal

The following fact is proved in [1].

Let *S* be a set of vectors, on which the Hausdorff distance between *A* and *B* is reached. The position of *A* and *B* is optimal if and only if the vector *(0; 0)* belongs to the convex envelope of the vectors from *S*.

This fact allows to define whether the current position is optimal within the following steps:

1. Calculate the vectors, on which the Hausdorff distance between *A* and *B* is reached, store them to *S*
2. Construct a convex polygon from the vectors in *S*
3. Define, whether *(0; 0)* is inside of the polygon.

# Main Logical Steps of Programme

## Preparation of the external resources

The programme prepares the external resources: assigns the files on the disk to the proper variables and opens them.

The procedure prepareSources(inPath1, inPath2, outPath: String) is responsible for this.

## Parsing of the polygons

The programme one by one reads tokens from the input streams and constructs the internal representations of the polygons. The procedure readPolygon(target: pLinkedList; var source: Text) is responsible for it.

## Calculation of the Hausdorff distance

The programme calculates the vectors, on which the Hausdorff deviations of *A* from *B* and vice versa are reached. Knowing these vectors, it also calculates the Hausdorff distance.

The procedures polygonPolygonDistanceVectors(target, pol1, pol2: pLinkedList) and hausdorfDistanceVectors(target, pol1, pol2: pLinkedList) are responsible for it.

## Sorting of the Hausdorff distance vectors

On this stage, the programme sorts the vectors calculated on the previous stage. The vectors are sorted by the angle between the OX-axis and the vector. The order is non-descending. This stage allows to create a convex polygon out of these vectors.

Procedure sortByAngle(target, source: pLinkedList) is responsible for actions of this step.

## Optimality test

Using the sorted list of the vectors obtained in step 4, the programme tests the optimality of the current mutual position of the polygons.

The procedure isOptimal(distVecs: pLinkedList) is responsible for it.

## Closing of the external resources

All external resources are closed and freed on this stage. The procedure closeSources() ensures these actions.

See the documentation included to the source code for detailed information about each step, actual implementation and other details. Source code is distributed freely and could be downloaded, for example, from the [GitHub repository](https://github.com/slavenkof/HausdorffPascal) (https://github.com/slavenkof/HausdorffPascal).

# Input Data Format

The single number on the first line specifies the number of points in the polygon. Each of the following lines contains two numbers with an x- and y-coordinates of the point separated by one space. Coordinates are real numbers.

Example:

4

1,1 1

-1 1

-1 -1

1 -1

# Examples of Programme’s Work

Input1: in10.txt

4

2 1

-2 1

-2 -1

2 -1

Input2: in11.txt

4

1 2

-1 2

-1 -2

1 -2

Output: out1.txt

The distance is reached on the following vectors:

Vectors

-1.00000; 0.00000

1.00000; 0.00000

-0.00000; 1.00000

-0.00000; -1.00000

Hausdorff distance: 1.00000

The mutual position of the polygons is optimal:

Input1: in10.txt

4

2 1

-2 1

-2 -1

2 -1

Input2: in20.txt

1

0 0

Output: out2.txt

The distance is reached on the following vectors:

Vectors

-2.00000; -1.00000

2.00000; -1.00000

2.00000; 1.00000

-2.00000; 1.00000

Hausdorff distance: 2.23607

The mutual position of the polygons is optimal: TRUE

Input1: in10.txt

4

1 1

-1 1

-1 -1

1 -1

Input2: in20.txt

1

0 0

Output: out3.txt

The distance is reached on the following vectors:

Vectors

-2.00000; -1.00000

2.00000; -1.00000

2.00000; 1.00000

-2.00000; 1.00000

Hausdorff distance: 2.23607

The mutual position of the polygons is optimal: TRUE

Input1: in40.txt

3

0 200

86.6025403 50

173.2050807 200

Input2: in41.txt

3

213.3974597 550.0

386.6025403 550.0

300.0 700.0

Output: out4.txt

The distance is reached on the following vectors:

Vectors

126.79492; 500.00000

Hausdorff distance: 515.82647

The mutual position of the polygons is optimal: FALSE

Input1: in40.txt

3

0 200

86.6025403 50

173.2050807 200

Input2: in43.txt

3

1.2256532319732116E-5 99.99999291765553

173.20509285653213 99.99999291765553

86.60255255653232 249.99999291765545

Output: out5.txt

The distance is reached on the following vectors:

Vectors

43.30128; -25.00001

0.00000; 49.99999

-43.30126; -25.00000

Hausdorff distance: 50.00001

The mutual position of the polygons is optimal: TRUE

# Ideas for Programme’s Improvement

1. To implement the BASH-friendly working mode;
2. To deal with memory leaks;
3. To implement the analytical and/or numeric algorithms of optimization (see [1]);
4. To implement the analytical way of optimization for nonconvex polygons;
5. To implement the graphical output;
6. To implement a more user-friendly interface.

# Literature

# Lakhtin A.S., Ushakov V.N. Minimization of the Hausdorff distance between convex polyhedrons.// Journal of Mathematic Science, New York, Vol.126, №. 6, 2005. 1553-1560 p.