Efficient Hausdorff distance metric calculation algorithm for human position comparison to template and its performance and precision research

Karolis Ryselis, Tautvydas Petkus
Department of Software Engineering
Kaunas University of Technology
Kaunas, Lithuania
e-mail: Karolis.Ryselis@ktu.edu, Tautvydas.Petkus@ktu.edu

***Abstract*–This article presents efficient algorithm to calculate Hausdorff distance metric and heuristic algorithm that maximizes amount of intersecting points in two point sets very quickly. Performance and precision research of those algorithms is presented. It is shown that algorithm is applicable for real time position tracking using "Kinect" device.**

Keywords–Kinect; Hausdorff distance; pattern matching

I. INTRODUCTION

"Kinect" sensor is a motion tracking device presented by "Microsoft". It is one of the most popular motion tracking devices used at home. "Kinect's" application programmable interface presents external applications with current human position [1]: skeleton stream that consists of processed human joint position information; depth stream, which is a two-dimensional matrix with data about every pixel's distance from the sensor to the closest object in its direction; body index matrix which assigns every pixel to a tracked player's body. Skeleton stream is processed by "Kinect" and presents external application with information about human body parts. Therefore, it is applicable when information about separate body parts is needed. However, it is shown by some researches [2] that skeleton stream is only accurate when the user is visible by "Kinect" comfortably and the user is standing. Skeleton is warped under different conditions.

Specialized algorithms are needed to process data from other data streams and compare them to a template. All data streams present the user with large amount of data compared to preprocessed skeleton stream. Thus, they require efficient algorithms to process them if real time or near real time tracking is required.

The easiest and simplest to process data stream is body index stream. If only one human is tracked this stream is easily deduced to a binary matrix where 1 represents a pixel that belongs to human body, 0 – that is does not. Hausdorff distance metric is a good metric for this kind of data [3]. Say that $A = \{a_1, a_2, \dots, a_n\}$ and $B = \{b_1, b_2, \dots, b_m\}$ are finite sets of points. Then Hausdorff distance is defined as follows:

$$H(A, B) = \max(h(A, B), h(B, A)), \text{ kur} \quad (1)$$

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\| \quad (2)$$

Here $\|a - b\|$ could be any distance metric between points a and b, e, g., Euclidean norm or L_2 norm. Semantically this metric shows the distance to the most distance point from the other set corresponding to a chosen distance metric.

Looking at (2) formula it is trivial to see that direct calculation of the metric has computational complexity of $O(nm)$ because all points of both sets must be compared. If $m \approx n$, complexity approaches to $O(n^2)$ and is not applicable for real time use.

II. HAUSDORFF DISTANCE CALCULATION METHODS

A. *Yanir Taflev's method*

Yanir Taflev presents an open source solution to compute Hausdorff distance metric [4]. This algorithm does not calculate the distance itself, it produces Hausdorff distance matrices instead. Each point of such matrix shows the distance to the closest point from set B for each point of set A. Hausdorff distance is then equal to the maximum value of both distance matrices.

Hausdorff distance matrix is calculated by constructing initial matrix first. A value of 0 is assigned to every point of set B and ∞ for all other points. For all points of set B the matrix is traversed in squares of increasing size and a value of square edge length is assigned if the value of that point is not less that the length of square edge. Chebyshev distance is computed because of such matrix construction.

Algorithm is implemented in C# language and is used in C# framework "Shape Matching Framework".

B. *"Elastic Search" method*

"ElasticSearch" client created by "Vivid Solutions" is an open source Java project [5]. It used discrete Hausdorff distance implementation. The computed distance is approximate. The algorithm is not given by "Vivid Solutions". The implementation is used to compare geometries in "Elastic Search" engine internally.

C. *Princeton University's method*

Princeton University's resolving library "RgResolve" uses Hausdorff algorithm's Java implementation [6]. Algorithm relies on assumption that all points are sorted either clockwise or counterclockwise. Points are then analyzed in this order. Data is analyzed as a list of points instead of two-dimensional

matrix, which is different from Yanir Taflev's implementation. The distance is calculated using Euclidean norm.

III. PROPOSED HAUSDORFF DISTANCE CALCULATION METHOD

The human could be in any part of "Kinect's" frame. The result of the algorithm must not depend on where the human is standing. A transformation of human's silhouette is needed before computing Hausdorff distance. It must move the silhouette to the position where it would intersect with the template the most.

These requirements must be met by efficient Hausdorff distance calculation algorithm:

1. Execution time is close to 30 ms with "Kinect's" body index stream data
2. Algorithm must find an optimal linear transformation that makes the template and user's silhouette be oriented in such a way that the amount of intersecting points is maximum and the Hausdorff distance is minimum

A. *Proposed algorithm to find the optimal transformation*

Say that we have two finite sets of points $A = \{a_1, a_2, \dots, a_n\}$ and $B = \{b_1, b_2, .., b_m\}$. Let us assume that the human will be oriented vertically in the frame, i.e., the normal vector of the base that the user is standing on is vertical. This assumption eliminates the need of rotation transformation. Two linear transformations need to be made: scale and shift. Say that the center point of set B is b_m. The shape will be scaled using this point as the center. Say that optimal scale coefficient is s, optimal shift transformation vector - $\vec{t}$. Then the transformed point set is

$$B_t = \{b_{t1}, b_{t2}, \dots, b_{tn}\} = b_m + \frac{(b_1 - b_m + \vec{t})}{s}, b_m + \frac{(b_2 - b_m + \vec{t})}{s}, \dots, b_m + \frac{(b_n - b_m + \vec{t})}{s} \quad (3)$$

After the transformation set B_t should maximize cardinality function

$$i(A, B_t) = |A \cap B_t| \quad (4)$$

Precise calculation of such function is computationally intensive, so heuristic algorithm is used instead.

A fast and quite accurate heuristic for scale transformation is to compute scale coefficient as the ratio of heights of the two shapes. If template and user are in similar positions, e.g., both are standing, both silhouettes could be made the same height. Then

$$s = \frac{h_A}{h_B}, \text{ where} \quad (5)$$

$$h_A = \max_{a \in A} a_y - \min_{a \in A} a_y, \quad (6)$$

where a_y- coordinate in vertical axis.

A fast to compute shift transformation algorithm is to shift one of the two silhouettes using a vector that is equal to difference between centroids of both shapes. Centroid is defined as

$$A_C = \left(\frac{\sum_{a \in A} a_x}{n}; \frac{\sum_{a \in A} a_y}{n}\right) \quad (7)$$

Then

$$b_i = C_B + \frac{(b_i - C_B + C_B - C_A)}{s} \quad (8)$$

If silhouettes of template and human belong to the same sequence of movement, we can assume that these transformation parameters will stay constant during the process of matching. Thus, they can be calculated in the beginning and used later. This should speed up the tracking even more. Then

$$t = C_{B1} - C_{A1} \quad (9)$$

$$s = \frac{h_2}{h_1} \quad (10)$$

$$b_i = C_B + \frac{(b_i - C_B + t)}{s} \quad (11)$$

B. *Proposed Hausdorff distance calculation algorithm*

Proposed algorithm is based on Yanir Taflev's ideas, but is better optimized for speed. The use of this algorithm eliminates the need to convert data from two-dimensional matrix to other formats, e.g., a list of points. Chebyshev distance is used as distance metric. Firstly, a two-dimensional matrix is set up. All points with coordinates that belong to the silhouette are set to 0, others – with ∞. A point queue is constructed. Initially it consists of all points that belong to the silhouette. A point is taken from the queue and 8 surrounding equidistant points are analyzed. The distance from the original points to those surrounding points is always 1, so the distance from the other silhouette to those points cannot be larger than the original point's distance increased by one. If this value is less than already computed value for this point, this value is updated and the point is added to the queue. Distance is calculated for all points this way. All points that do not belong to the other set are then removed because they do not contribute to Hausdorff distance. Elements of this matrix show the differences between the silhouettes. Two such matrices must be calculated, so the computation of them can be made parallel – their computation is independent.

To reduce the amount of calculation matrix edges without silhouette points are cut. The resulting cut matrix is minimum rectangle matrix that contains all the points that belong to silhouette.

Common maximum of both matrices is Hausdorff distance. Algorithm pseudocode is given below.

```
matrices ← template matrix and user matrix
find common left upper corner of matrices
find common right lower corner of matrices
∀ matrix ∈ matrixes
    do initialize queue and ret
    while queue ≠ ∅
        do p ← take from queue
        val ← get p value from ret
        val ← val + 1
        ret ← min(r, val) ∀ adjacent point
r to p
set points of res that do not belong to
opposing matrix to 0
hausdorff distance = max(max(res[1]),
max(res2))
```

IV. METHOD OF RESEARCH

Two properties of the algorithm are evaluated: performance and precision.

To evaluate performance different types of frames are set up. The frames differ in size and distance between silhouettes. Used frame sizes are 10% to 300% "Kinect's" frame size in steps of 10%. For each frame size, real human silhouette is analyzed. Performance is measured when the human and the template are in similar parts and in opposite parts of the frames. Performance is compared to other existing algorithms' performance. Each measure is run 10 times and execution time average is found. Three modifications of the proposed algorithm are measured – Hausdorff distance only, Hausdorff distance with transformation and Hausdorff distance with cached transformation parameters. Algorithm should work in less than 30 ms, i.e., faster than "Kinect' produces frames.

To evaluate precision different real human silhouettes are used. Best possible transformation is found using brute force algorithm, Hausdorff distance is calculated using the optimal transformation, then Hausdorff distance is found using the transformation calculated using proposed algorithm and the error is evaluated.

V. RESEARCH RESULTS

Execution times of all algorithms and all frame sizes are shown in figure 1. Same silhouettes were used, but scaled. Silhouettes were around a third of the frame apart.

Unit value in diagrams represents algorithm with cached transformation performance when frame size is equal to "Kinect's" frame size. This time is 26 ms for distant silhouettes and 21 ms for close silhouettes on "Intel i7-3770K" processor (3.5 GHz). Maximum acceptable value on those scales are 1.15 for distant silhouettes and 1.43 for close silhouettes. These values are hardware-dependent and presented for approximate evaluation. The processor used is a little more powerful than "Kinect's" system requirements (i7 series processor, 3.1 GHz clock frequency).

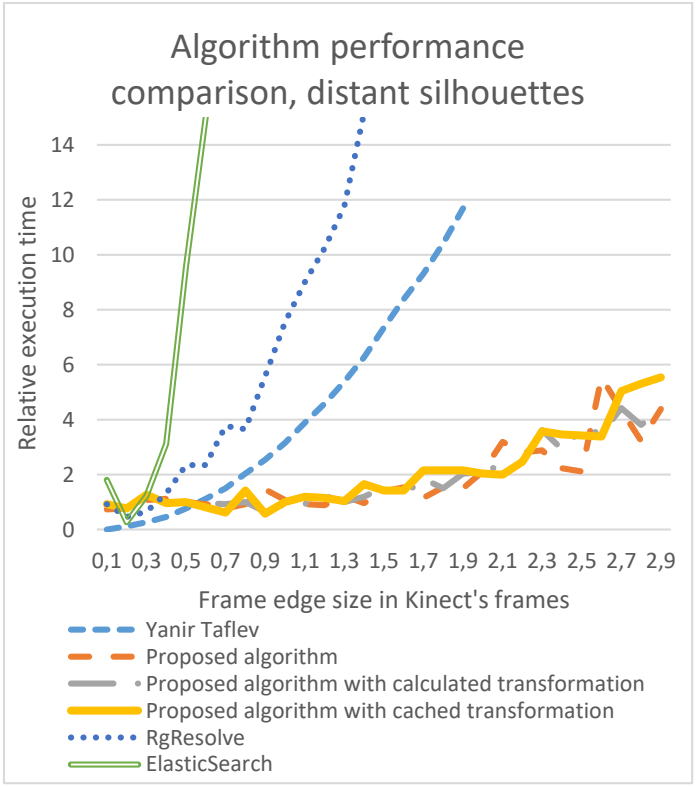

Figure 1. Algorithm performance comparison (distant silhouettes)

The diagram reveals that "Elastic Search" algorithm is the slowest, "RgResolve" – better, Y. Taflev's algorithm would be applicable up to frame size of 0.6 "Kinect 2" frame size. This is close to "Kinect 1" frame size. Performance of proposed algorithm varies and no clear dependency is visible, but execution times slowly increase with increasing frame size. It could be caused by parallel nature of the algorithm. Threads are spawned by the runtime and operating system and determination of their performance could be difficult to predict. Nevertheless, frame sizes of around 1.3 "Kinect's" frame size and smaller are processed faster than in 1.15 relative units and 1.7 and less – in 2 relative units.

Figure 2 shows the result of similar measurements, but the distance between the silhouettes was minimized. Other algorithms had little to no impact because of this change, but proposed algorithm performed better under those conditions. 1.5 relative units is reached at around 1.4-1.5 "Kinect's" frame size, 2.5 – at around 1.8-2 times larger frames than "Kinect's". At round 3 "Kinect's" frame sizes performance increase is around 1.5 times compared to distant silhouette test.

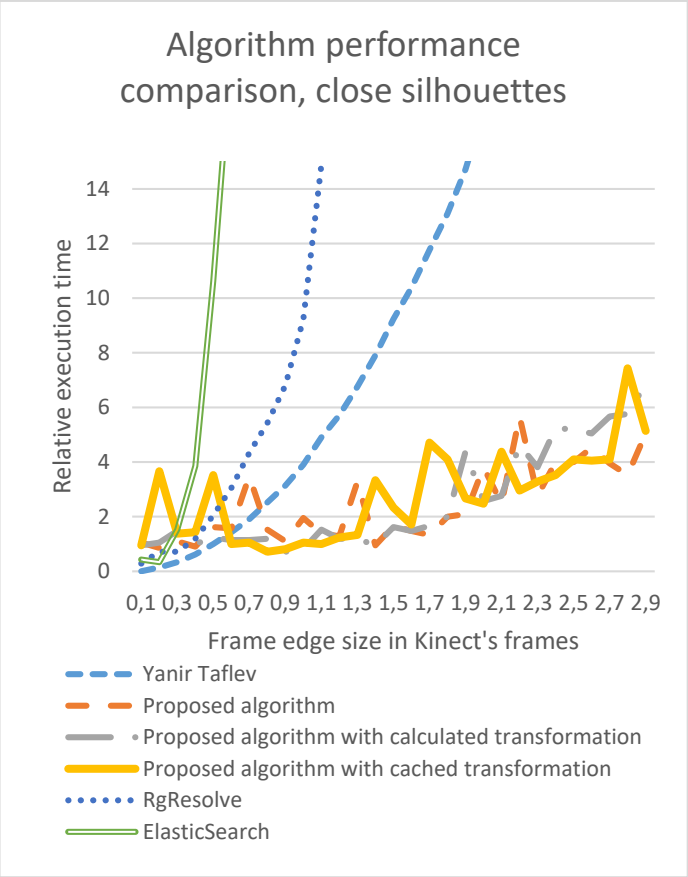

Figure 2. Algorithm performance comparison (close silhouettes)

Algorithm precision was tested using different human silhouettes. Average error was 3.8%, minimum error – 0% (exact result), maximum error – 15.5%, median error – 2.7%. Data set size – 21 silhouettes.

VI. CONCLUSIONS

The proposed algorithm performs faster than existing open source alternatives to calculate Hausdorff distance, but high variation of execution speed is observed because of parallelism involved in the calculations. Despite that, the algorithm in the worst case works faster than in 30 ms with "Kinect 2" recommended hardware, therefore it is applicable for human motion tracking using "Kinect" sensors. Proposed algorithm is precise if applied for human motion tracking and proposed heuristic is worth applying, because it adds little to execution times and precision loss is low.

Proposed algorithm works faster with silhouettes in the same part of frame than distant silhouettes. If this algorithm is applied for human motion tracking and recommended transformations are applied, this will always be the case and comparison will always be fast.

Algorithm is well suited for human position tracking when the human does not change his position relative to sensor, only his pose changes, or the human moves together with the template silhouette. In this case proposed heuristics work well and near-precise value of minimum Hausdorff distance with Chebyshev metric is found in short time.

REFERENCES

[1]"Kinect API Overview," [Online]. Available: https://msdn.microsoft.com/en-us/library/dn782033.aspx. [Accessed 10 03 2017].

[2] K. Ryselis, T. Petkus, "Nestandartinių žmogaus kūno pozicijų atpažinimo tikslumo naudojant „Kinect 2.0" jutiklius tyrimas," Kaunas, 2015.

[3] D. P. Huttenlocher, G. A. Klanderman, W. J. Rucklidge, "Comparing Images using the Hausdorff Distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* no. 15, pp. 850-863, 1993.

[4] Y. Taflev, "Using the Hausdorff distance algorithm to point out differences between two drawings," 27 09 2009. [Online]. Available: https://www.codeproject.com/articles/42669/using-the-hausdorff-distance-algorithm-to-point-ou. [Accessed 10 03 2017].

[5] V. Solutions, "elasticsearch-client/DiscreteHausdorffDistance," [Online]. Available: https://github.com/jprante/elasticsearch-client/blob/master/elasticsearch-client-jts-jdk5/src/main/java/com/vividsolutions/jts/algorithm/distance/DiscreteHausdorffDistance.java. [Accessed 10 03 2017].

[6] "Hausdorff Distance," Princeton University, [Online]. Available: http://www.princeton.edu/~rkatzwer/rgsolve/doc/edu/princeton/polygon/HausdorffDistance.html. [Accessed 12 03 2017].