Convex Polygon Fitting in Robot-Based Neurorehabilitation

F. Stroppa\textsuperscript{a,*}, C. Loconsole\textsuperscript{b}, A. Frisoli\textsuperscript{a}

\textsuperscript{a}PERCRO Laboratory, TeCIP Institute, Scuola Superiore Sant’Anna, L. Alamanni 13B, 56010 Ghezzano, San Giuliano Terme, Italy
\textsuperscript{b}Department of Electrical and Information Engineering (DEI), Polytechnic University of Bari (Italy), E. Orabona 4, 70126 Bari, Italy

Abstract

Fitting a polygon to a set of points is a task that finds application in many scientific fields. In particular, in robot-based neurorehabilitation, it would be interesting to retrieve the shape that best fits with the path followed by a patient, and evaluate the performance based on the accuracy of the drawing. However, when dealing with a dataset sampled by a drawn trajectory, the methods proposed by the literature may not be exhaustive.

In this work we propose a method to define the distance between a set of points and a polygon, which is used as cost function of a Genetic Algorithm to solve the polygon fitting problem. This method involves a novel space separation metrics to retrieve the correct polygon edge to be compared with each point of the set, featuring linear time complexity.

We compared the proposed approach with the metrics known in the literature, finding that our method performs significantly better in retrieving the original polygon. Finally, we present a robot-based rehabilitation application in which the proposed method is used to evaluate the performance of a group of subjects. The achievements of twenty healthy subjects were compared with three stroke patients. Results emphasize significant differences between the two categories of subjects, proving that the proposed algorithm can quantitatively determine the degree of impairment of a stroke survivor and be used in the future as reference for monitoring and enhancing the efficiency of robot-based therapies.

Keywords: Polygon Fitting, Genetic Algorithms, Geometry Codes, Medical Treatment

1. Introduction

The problem of shape fitting from scattered data is a common geometric problem exploited in several applications. In particular, when the data describes a trajectory, like a set of points traced by a human being, the approaches proposed in the literature might be insufficient.

This problem finds interesting application in the area of robot-based neurorehabilitation\cite{1,2,3}, considering a scenario where a patient with high arm disability is asked to perform a circular or polygonal trajectory following a target with a wearable exoskeleton\cite{4}. Assuming that the device is transparent, the accuracy of the final performance is directly correlated to the subject’s degree of impairment. At the end of the task, the overall performance can be evaluated based on the differences between the reference trajectory and the one traced by the patient. In this case, the patient’s movements are described by a collection of scattered points sampled during the exercise. However, dealing with single points might create some confusion...
and increase the mathematical burden, whereas an actual geometrical shape can easily be compared to the reference proposed by the task. With this motivation, a geometrical shape can be estimated from a recorded dataset, to approximate the patient’s trajectory. Hence, the problem requires a fitting procedure considering all the points as part of a unique path. Besides the neurorehabilitation, such a method can also be used in several other application fields, like image processing, pattern recognition, computational metrology, or robot guidance.

The problem of shape fitting from scattered data has been widely faced in the literature. The most common investigated shapes are circular or elliptical [5, 6, 7] and the proposed methods involve quadratic form [8], least square fitting [9], Bayesian estimator [10], and Hough transformation [11]. In shape classification or approximation of unorganized 2D and 3D point datasets, RANSAC is one of the most efficient automatic algorithms [12]. However, this procedure does not take outliers into account during the model estimation, considering them as noise or disturbance. Discarding the outliers while fitting the best geometrical shape from a human-drawn trajectory means not considering part of the path, compromising then the final evaluation. For industrial applications or object recognition in images, also rectangular shapes have been addressed [13, 14], but these methods solved the problem by retrieving the bounding box of the dataset, which is not suitable for the trajectory recognition. More interesting approaches for rectangle fitting involve minimum width annulus problem [15] or largest empty rectangle problem [16], which present linearithmic time complexity. To the best of our knowledge, the most performing algorithm for polygon fitting is proposed by Sinnreich [17], featuring least-squares method to retrieve the best shape from a set of points in the plane. However, this method provides a curve rather than a polygon, and due to the trigonometric nature of the output, it is not easily replicable for other complex polygons.

Another possible approach is to treat the shape fitting as an optimization problem, searching for the geometric outline that minimizes the distance between its edges and all the points within the given set. The problem of polygonal approximation has been widely addressed in the literature by solving optimization problems, and Genetic Algorithms (GAs) are one of the most diffused solvers for this class of problems [18, 19, 20, 21]. However, to the best of our knowledge, the problems faced in the literature are mostly focused on shape interpolation rather than shape fitting, whereby the proposed methods are not suitable for fitting trajectories.

To formulate the shape fitting as an optimization problem, it is required to define a geometric metrics as the fitness function leading to the best performing solution. A similar problem has been addressed by Liparulo et al. [22], where they propose a point-to-polygon distance estimation to evaluate membership functions in fuzzy clustering algorithms. Their study proposes three different approaches: Gaussian method, Cone-shaped kernel functions, and a solution exploiting geometrical properties. The first two methods perform better in terms of time, but are less accurate than the geometrical approach. In particular, this last strategy defines the distance between a point and a polygon as the one between the point and the closer edge (or vertex). However, since their study is focused on shape recognition rather than shape fitting, the points inside the polygon outline are considered as part of the shape, making this method not suitable for the problem of fitting a polygon outline.

The evaluation of a consistent shape-point distance must consider the subdivision of the space in which the geometrical elements are located, to select the most suitable polygon edge to be compared with each single point. A valid approach is given by Voronoi diagrams [23], a partitioning of the metric space determined by the distance among a given set of discrete elements. Assuming that the metric space is a plane, the Voronoi diagram for a given set of points is the partition that associates each point to a region of the plane, called Voronoi Cell, so each cell contains all the points that are closer to its defining point than to any other point in the set. Voronoi diagrams can also be extended to partitioning of the metric space among edges rather than points [24, 25, 26]. Indeed, some studies use Voronoi diagrams as metrics for distance evaluation
In the light of these considerations, Voronoi diagrams provide a suitable method to select the closer edge from a given point. This metrics computes the distance selecting the closer edge from the point among all the edges of a given polygon, which is also the point that lies in the same Voronoi cell containing the chosen edge. Extending the metrics to a set of points rather than a single one, this method presents a complexity equal to \(O(n \cdot m)\), where \(n\) is the number of points and \(m\) is the number of edges. However, the “Voronoi metrics” may not be the best solution for shape fitting, due to the fact that some local minima are very close to the global one showing completely different polygons as solutions. Since this metrics does not ensure that every edge of the proposed polygon will be associated to at least one point in the dataset, this may deceive a heuristic research algorithm, which will generate a non-optimal solution even providing an efficient fitness value.

In this work, we present an approach to solve the shape-fitting problem implementing a GA, and its application to evaluate the motor skills of arm-impaired patients. We decided to use GAs due to their adaptability to multi-variable problems and their ability of finding good solutions within a reasonable time, which meets the requirements of fast execution for real-time systems, such as rehabilitation applications. In particular: (i) we defined a new mixed metrics to evaluate the shape-point distance, named “Polar metrics”, which combines Voronoi diagrams and a vertex-centroid space subdivision to overcome the limitations of the Voronoi approach; (ii) we implemented this metrics as the GA fitness function to solve the polygon-fitting problem on a generic set of 2D scattered points; and (iii) we applied the genetic application to evaluate performance indices from the shapes drawn by post-stroke patients with a wearable robot. In robot-based rehabilitation field, several studies make use of assistance-as-needed adaptive regulation to promote motor skill learning [30, 31, 32, 33]. Since the aforementioned indices provide both inter and intra-subject quantitative monitoring, they could be exploited in future applications as reference for setting the proper difficulty level of a rehabilitation task. Furthermore, it is important to note that the “Polar metrics” alone could be exploited as a performance index, simply comparing the set of points sampled by the robot and the given trajectory. The need of the GA, and then of the shape fitting procedure, relies in identify the closest shape that the patient has drawn, which provides the following benefits: (i) the direct comparison between two polygons; (ii) the evaluation of the accuracy in drawing the fitted shape, which may not necessary be the same as the reference; and (iii) the graphical visualization of the drawing at the end of the task, which is an effective way to provide visual and valuable feedback to the patient. In particular, visualizing the drawn shape may stimulate patients’ interest and/or challenges them to improve during the following sessions [34, 35], being more immediate than a numerical score that could also have no meaning for a non-medical user.

The rest of the work is structured as follows. Sec. 2 will provide an overview of the algorithm and the space separation metrics. Sec. 3 will describe a set of experiments aiming at validating the method, by (i) demonstrating the results achieved with the Polar metrics with respect to other methods, (ii) showing a benchmark of performance achieved by the GA, and (iii) comparing the genetic procedure with another fitting procedure. In Sec. 4 the proposed GA will be applied in a robot-based neurorehabilitation scenario to gather information about post-stroke patients’ motor skills. Finally, Sec. 5 will conclude the work and illustrate future studies.

2. Genetic Algorithm-Based Shape Fitting Method

This work describes the implementation of a GA to find the best geometrical shape that fits with a given set of scattered points (point cloud) in the plane. Point clouds can be obtained by sampling polygon outlines or recording the movements of a cursor at a certain frequency. Fig. 1 shows the block diagram of the proposed method. Note that a ground truth shape can be given as input and used as phenotype in the initial
GA population: this allows the algorithm to start from a position in the search space that could be relatively close to the optimum solution, assuming that the point cloud resembles the given shape.

Figure 1: Block Scheme of the proposed Genetic Algorithm.

In particular, the algorithm focuses on irregular convex polygons, since regular polygons are sub-cases of irregular ones and curves or ellipses have already widely been addressed in the literature.

2.1. Problem Domain

The phenotypes of the GA population describe geometrical convex polygons. Each chromosome is codified as a binary string, and is composed by the number of bits needed to represent the domain of the problem. All the chromosomes within a certain population must have the same length, but different shapes will have a different representation scheme.

For a general polygon, a single chromosome must encode:

1. the polygon edges, defined by their length;
2. the orientation with respect to the current frame, defined in degrees; and
3. the center point, defined by its $x$ and $y$ coordinates in the current frame.

The number of edges is the only parameter that changes among the different types of polygons: for a rectangle the algorithm must consider its width and height; for an equilateral triangle or pentagon a unique edge is sufficient. It follows that the number of genes changes according to the type of polygon to be represented, featuring three fixed genes (orientation, $x$ and $y$ coordinate of the origin point) and a variable number of genes depending of the number of polygon’s edges.

---

It is worth to point out that the unit of measure between distances depends on how the current frame is defined. In this work, we describe our implementation in cm, but different units or multiples can be used in accordance with the frame.
All the parameters composing the problem are subjected to constraints depending on the spatial composition of the input point cloud. This allows the algorithm to narrow the search space in a neighborhood of the solution, consequently reducing the execution time. Considering a point cloud $P$ composed of $n$ points $p_i := (x_i, y_i)$, the algorithm extracts the following features:

1. the length of the point cloud bounding box diagonal, $\text{diag}_P$;
2. the centroid of the point cloud, $c_P := (x_c, y_c)$; and
3. the center of the point cloud bounding box, $\text{bb}_P := (x_{bb}, y_{bb})$.

The edge domain ranges in $(0, \text{diag}_P]$, whereas the origin point is narrowed in a circle having starting point $p_s (1)$ and a radius $r (2)$. It is obvious that the chances to find the solution within this search space are very high.

$$p_s = (c_P + \text{bb}_P)/2 \quad (1)$$

$$r = \sqrt{(c_P - \text{bb}_P)^2} \quad (2)$$

The orientation depends on the geometry of the polygon rather than the input set, and is limited by $\text{max}_A$, which is the maximum value allowed to prevent the generation of redundant phenotypes. For example, in rectangles an orientation higher than $179^\circ$ would generate phenotypes already encoded in previous representations, with inverted values for the edges. Assuming that each of these 180 values (from 0 to 179) are codified with a unique combination of bits, an 8 bit word is required. Such a word represents up to 256 values, which means that $256 - 180 = 76$ values will be discarded (from 180 to 255). This must be considered as a constraint during GA execution in the case in which one of these 76 exceeding values is generated during the population initialization, crossover, or mutation.

To sum up, a single chromosome is structured with the following parameters:

- the *polygon edges*, codified with the number of bits needed to represent $\text{diag}_P$; one parameter for regular polygons, two or more for irregular ones;
- the *angular orientation* with respect to the current frame, expressed in degrees and codified with the number of bits needed to represent the maximum orientation angle $\text{max}_A$; and
- two parameters representing the *displacement* from the point $p_s$, along both $x$ and $y$ coordinates, codified with the number of bits needed to represent twice the value of the radius $r$.

**Figure 2:** Chromosome structure for a generic phenotype. Note that the $\text{ceil}$ function maps the input value to the smallest following integer.
Fig. 2 shows the chromosome composition for a generic \( m \)-edge convex polygon.

The following evaluations will address each polygon as a collection of vertices \( v \in V, |V| = m \), which implies that a strategy to retrieve \( V \) is required. Such a procedure must be defined by the geometry of the polygon, therefore it cannot be generic: different type of polygons will feature different methods to obtain their vertices from the data codified in the chromosomes. It is important to note that exploiting the aforementioned parameters as descriptors in the chromosomes, rather than the mere vertices, reduces the complexity of the genetic procedure. In particular, by having the vertices as phenotype descriptors, the generation and crossing of genes would produce random points describing random polygons, and the only constraint able to preserve the shape type during all the iterations would be exactly the set of distances between each vertex. The solution described in Fig. 2 already inherently implements the geometry of the problem, and no further check is required.

It follows that the optimization problem can be extended to any kind of convex polygon once defined its geometry and the strategy to retrieve its vertices.

2.2. Fitness Function

We propose a novel algorithm to evaluate the distance between a point and a convex polygon outline, used as the GA fitness function. The procedure aims at being suitable for fitting scattered points to a given convex polygon. This means selecting the “logically” correct edge to compare with each point in the data. The selection of such an edge requires the definition of a comparative metrics.

![Figure 3](image)

**Figure 3:** The figure shows how the two different metrics separate the space and associate different points to different edges. The blue rectangle is the shape to be compared with the set of points, which could be a phenotype proposed as a solution by the GA. In this case, the best fit for the set is a rectangle having size 3 \( \times \) 4, whereas the proposed solution features a size of 5 \( \times \) 10 that is clearly bigger than required. A good metrics should provide a bad fitness value, so that the algorithm could discard it. For the points within the polygon, Voronoi diagram (a) deceives the fitting method due to its space separation metrics, featuring an average distance of 1.49. On the contrary, Polar subdivision (b) distributes points over the entire polygon, featuring an average distance of 1.86. Since the Polar metrics provides a higher value, the rectangle selected as solution may not be the best one, in the context of a minimization problem. The GA will then seek for different solutions in the search space rather than linger on this one.
As previously mentioned, Voronoi diagrams suggest that the distance to consider is between the single point and the edge that lies in the same Voronoi cell, which is also the closest distance between that point and every other polygon edge. However, using this metrics in the context of shape fitting might be problematic, especially while dealing with irregular polygons. For example, with a generic rectangle generated by a random heuristic method (a GA in this work), scattered points within the polygon may deceive the procedure. Fig. 3(a) shows that matching every point with the closest polygon edge might generate a non-optimal solution: the points that should be considered part of the vertical edge are matched with the horizontal edge of the rectangle proposed as solution.

A different approach is the one showed in Fig. 3(b) denominated “Polar metrics”. The strength of this metrics relies on associating every edge with at least one point of the dataset. The diagram is formed by connecting each vertex to their centroid, whereas the space outside the polygon is divided in the same way as the Voronoi diagram. For the space outside the shape, the Voronoi diagram manages correctly the problem with irregular convex polygons, whereas the proposed subdivision would be deceived by a polygon with several vertices concentrated in the same area. This is why the Polar diagram is composed by applying the Voronoi metrics in the space outside the polygon and the centroid-vertex one in the space within the polygon. Considering these assumptions, Polar and Voronoi diagrams coincide for regular polygons, such as equilateral triangles or pentagons, both inside and outside the shape outline.

The following subsections will illustrate in detail the Polar metrics and report the code for its implementation. Note that all the algorithms will show the optimized code; the operations concerning translation and rotation of points will be featured only for the x or y coordinate that will actually be used.

2.2.1. Space Partitioning and Edge Selection

Considering an m-edge convex polygon in the current frame, the algorithm takes as parameters:

- the array $V$ of the shape vertex sorted clockwise, having size $m$;
- the point $p$, whose distance from the polygon must be calculated;

and provides as result:

- the distance $d$ between $p$ and the convex polygon defined by $V$.

The distance $d$ is defined between $p$ and the the chosen polygon edge $e$. According to the metrics previously illustrated, $e$ is selected based on the position of $p$.

For a point $p$ lying outside the polygon, $e$ is chosen using the Voronoi metrics, which simply retrieves the closer edge to $p$, as shown in Algorithm 1.

On the other hand, for a point $p$ within the polygon, $e$ is retrieved exploiting the Polar metrics. The planar space is divided into $m$ cells by connecting all the vertices in $V$ to their centroid $c_V$. Each cell contains only one edge, defined by two consecutive vertices $v_i$ and $v_{i+1}$ ($i \in [1, m]$). At the end of the procedure, $e$ will lie in the same cell containing $p$.

The strategy used to retrieve this cell checks if $p$ lies between the segments $v_i c_V$ and $v_{i+1} c_V$ (Fig. 4(a)). This can be done by a simple coordinate check. By translating (Fig. 4(b)) and rotating (Fig. 4(c)) both the segments and $p$, the procedure discriminates whether the point lies in the cell only by checking its y-coordinate.

\[\text{As will be seen later, the procedure also takes the centroid of } V \text{ as a parameter; this prevents to calculate the same centroid for every point } p \text{ in the input dataset, saving computational time.}\]
Figure 4: Procedure for the “above/below” check. First, the segment connecting the vertex $v$ and the centroid $c_V$ is translated to make $c_V$ coincide with the origin of the current frame. Then, the entire segment is rotated of an angle $-\alpha$ to set all its $y$ coordinates equal to zero. The same translation and rotation are applied on the point $p$, so the evaluation is performed only by checking its $y$-coordinate.
Algorithm 1: Distance with Voronoi

input : The point p to be compared with the polygon, having x and y coordinates
input : The array of vertices V, having m vertices
output: The distance d

1 begin
2   d ← 0;
3   min ← -1;
4   for i ← 1 to m do
5       v1 ← V[i];
6       if i ≠ m then
7           v2 ← V[i + 1];
8       else
9           v2 ← V[1];
10      d ← pointSegmentDistance(p, v1, v2);
11      if min = -1 then
12          min ← d;
13      if min > d then
14          min ← d;
15   d ← min;

Algorithm 2: Distance with Polar

input : The point p to be compared with the polygon, having x and y coordinates
input : The array of vertices V, having m vertices
input : The centroid cV of the polygon vertices, having x and y coordinates
output: The distance d

1 begin
2   d ← 0;
3   for i ← 1 to m do
4       v1 ← V[i];
5       if i ≠ m then
6           v2 ← V[i + 1];
7       else
8           v2 ← V[1];
9           s1 ← checkPointBelowSegment(p, v1, cV);
10          s2 ← checkPointBelowSegment(p, v2, cV);
11          if s1 = true and s2 = false then
12              d ← pointSegmentDistance(p, v1, v2);
Algorithm 3: Check Point Below Segment

- **input**: The point $p$ to be compared with the segment, having $x$ and $y$ coordinates
- **input**: A polygon vertex $v$, having $x$ and $y$ coordinates
- **input**: The centroid $c_V$ of the polygon vertices, having $x$ and $y$ coordinates
- **output**: The result of the comparison: **true** if $p$ is below the segment, **false** if is above

```
begin
  $v' \leftarrow v - c_V$;
  $\alpha \leftarrow \text{atan2}(v'.y, v'.x)$;
  $p' \leftarrow p - c_V$;
  $p''.y \leftarrow p'.x \cdot \sin(-\alpha) + p'.y \cdot \cos(-\alpha)$;
  if $p''.y \geq 0$ then
    result $\leftarrow$ true;
  else
    result $\leftarrow$ false;
end
```

Algorithm 3 shows how to retrieve the distance $d$ by selecting $e$, using the function defined in Algorithm 2 to perform the test. As Fig. 4(c) shows, at the end of the procedure all the points lying on the segment will have the $y$-coordinate equal to 0. Thus, the point $p$ will lie in the cell defined by $v_i c_V$ and $v_{i+1} c_V$ if, and only if, its $y$-coordinate is less and greater than 0 when the first and second segments are aligned with the $x$ axis, respectively. The procedure will find only one cell in which the point $p$ belongs, due to the convexity of the polygon.

**Figure 5**: Evaluation of the point-edge distance for three cases: (i) $p$ is compared to the first endpoint, (ii) is compared to the segment, and (iii) $p$ is compared to the second endpoint.
2.2.2. Distance Point-Edge

At the end of Algorithm[2], the distance $d$ is calculated considering the position of the point $p$ with respect to $e$. With reference to Fig, the evaluation changes whether $p$ lies (i) in the portion of plane defined by the two lines orthogonal to $e$ and passing by its endpoints, or (ii) in one of the two half-planes that do not include $e$. In the first case $d$ will be defined as the distance between $p$ and $e$; in the second one, as the distance between $p$ and the related endpoint.

The calculation is made considerably easier by transforming the pose of the items to make $e$ lying on the horizontal axis of the frame. The procedure is shown in Algorithm[4] exploiting the same idea shown in Sec. 2.2.1 for the horizontal axis of the frame. The procedure is shown in Algorithm 4, exploiting the same idea shown in Sec. 2.2.1 for the “above/below” criteria to rotate both $p$ and $e$.

Note that on line 14, the distance between $p$ and $e$ is already given by the $y$-coordinate of $p$ after it has been translated and rotated, which prevents the burden of calculating the equation of the line where $e$ lies. For the other cases, line 12 and 16 show that $d$ is obtained as a simple euclidean distance.

**Algorithm 4**: Point Segment Distance

```plaintext
input : The point $p$ to be compared with the segment, having $x$ and $y$ coordinates
input : The first endpoint $v_1$ of the segment, having $x$ and $y$ coordinates
input : The second endpoint $v_2$ of the segment, having $x$ and $y$ coordinates
output : The distance $d$ between $p$ and the segment

begin
1. $c ← (v_1 + v_2)/2;
2. v'_1 ← v_1 - c;
3. v'_2 ← v_2 - c;
4. $p' ← p - c;
5. $β ← \text{atan2}(v'_2.y, v'_2.x);
6. $v''_1.x ← v'_1.x \cdot \cos(-β) - v'_1.y \cdot \sin(-β);
7. $v''_1.y ← v'_1.y \cdot \cos(-β) - v'_1.x \cdot \sin(-β);
8. $v''_2.x ← v'_2.x \cdot \cos(-β) - v'_2.y \cdot \sin(-β);
9. $v''_2.y ← v'_2.y \cdot \cos(-β) - v'_2.x \cdot \sin(-β);
10. if $p''.x < v''_1.x$ then
    11. $d ← \sqrt{(p.x - v_1.x)^2 + (p.y - v_1.y)^2};$
else if $(v1''.x ≤ p''.x) ∧ (p''.x ≤ v2''.x)$ then
    12. $d ← |p''.y|;$
else
    13. $d ← \sqrt{(p.x - v_2.x)^2 + (p.y - v_2.y)^2};$
end
```

2.2.3. Point Inside The Polygon

Evaluating whether a point lies within the shape or not is the condition that determines which metrics should be used (Voronoi or Polar). This procedure exploits the same idea shown in Sec. 2.2.1 for the “above/below” criteria. Considering a generic convex polygon, a point $p$ lies within the polygon outline if

---

3It is worth to point out that the last vertex $v_n$ is taken with the first one $v_1$ since the shape is defined as a closed figure (statements 5-8 of Algorithm[12] and [5]).
it is located “below” all the polygon edges. With an $m$-edge convex polygon, this procedure is iterated $m$ times, by rotating each edge and evaluating where $p$ is located with respect of the edge.

The procedure is shown in Algorithm 5. Note that the point lies outside the polygon if the $y$-coordinate of $p$ is negative after it has been translated and rotated (lines 16-17-18). Moreover, the variable $dsp$ defined at line 10 indicates the displacement from the center of the edge $v_m$ to the centroid of the polygon $c_V$ for its translation.

Algorithm 5: Point Inside Polygon

| input : The point $p$ to be compared with the polygon, having $x$ and $y$ coordinates |
| input : The array of vertices $V$, having $m$ vertices |
| input : The centroid $c_V$ of the polygon vertices, having $x$ and $y$ coordinates |
| output: The result of the comparison: true if $p$ is inside the polygon outline, false if is outside |

1 begin
2 result ← true;
3 for $i ← 1$ to $m$ do
4     $v_1 ← V[i]$;
5     if $i ≠ m$ then
6         $v_2 ← V[i + 1]$;
7     else
8         $v_2 ← V[1]$;
9     $v_m ← (v_1 + v_2)/2$;
10    $dsp ← c_V - v_m$;
11    $v'_1 ← v_1 + dsp$;
12    $v'_2 ← v_2 + dsp$;
13    $p' ← p + dsp$;
14    $γ ← \arctan2(v'_2.y, v'_2.x) + π$;
15    $p''.y ← p'.x · \sin(γ) + p'.y · \cos(γ)$;
16    if $p''.y < 0$ then
17        result ← false;
18        break;
19 end

2.2.4. Overall Procedure

To sum up, Algorithm 6 indicates how to execute the overall fitness function, which is composed by the sub-methods previously illustrated. It is worth to point out once again that with Polar metrics we indicate the overall procedure, which exploits a mixed strategy composed of:

- a metrics defined by the Voronoi diagram for the points lying outside the polygon outline; and
- a metrics defined by the centroid-vertex separation, defined in this work with the term “Polar”, for the points lying within the polygon outline.

The array $V$ refers to a polygon phenotype generated by the GA at a certain iteration.
Algorithm 6: Fitness Function

```
input : The array of points P to be compared with the polygon, having n points
input : The array of vertices V, having m vertices
input : The centroid c_V of the polygon vertices, having x and y coordinates
output: The fitness value fit, calculated as the mean value of the distances between P and V

begin
  fit ← 0;
  for i ← 1 to n do
    p ← P[i];
    d ← 0;
    inside ← pointInsidePolygon(p, V, c_V);
    if inside = true then
      d ← distanceWithPolar(p, V, c_V);
    else
      d ← distanceWithVoronoï(p, V);
    fit ← fit + d;
  fit ← fit/n;
end
```

2.2.5. Complexity Estimation

The algorithm for the distance estimation with the Polar metrics features a linear complexity. The analysis can be performed considering a point cloud P having n points and a polygon composed of m edges.

The Voronoi metrics requires to check the distances occurring between all the n points and all the m edges, resulting in a time complexity O(n · m).

On the other hand, the Polar metrics requires more iterations. For each point p ∈ P, every edge is analyzed to evaluate whether p is located inside or outside the polygon, to select the proper space-separation method (Sec. 2.2.3), resulting in n · m iterations. Both Voronoi and centroid-vertex separation associate each of the n points to one of the m spatial cells, which requires more n · m iterations. Moreover, further m iterations are required to evaluate the centroid of the polygon; these last iterations are performed before to start the evaluation on P, so they are independent from n. It follows that, the number of iterations needed is m · (1 + 2 · n), which corresponds to a time complexity O(n · m).

2.3. Genetic Algorithm Details, Execution and Convergence

The GA starts the execution initializing its chromosome population with random values. For each iteration, the crossover and mutation operators are processed taking into account the constraints on the domains detailed in Sec. 2.2. Then, the algorithm analyzes each polygon in the population and evaluates the fitness as the mean distance between all the points in P and all the chromosomes, as described in Algorithm 6. Starting from the information encoded in the chromosomes, the vertices of each figure are calculated and given as input to the fitness function described in Sec. 2.2. Finally, the algorithm assigns a higher probability of reproduction to the phenotypes with lower fitness value, allowing the population to evolve toward more performing solutions.

More in detail, the algorithm implements two different strategy for chromosome selection, that could be chosen based on their performance:
• **Roulette Selection**, where the selection probability of each chromosome $\text{prob}$ is proportional to the fitness value \[3\]; and

$$
\text{prob}_i = \frac{\max(FIT) / \text{fit}_i}{\sum_i \max(FIT) / \text{fit}_i}
$$

(3)

• **Tournament Selection**, where the selection probability of each chromosome $\text{prob}$ is given as shown in \[4\], and assigned to the individuals from the best performing one to the worse, starting with a probability $\text{prob}_{\text{start}}$.

$$
\text{prob}_i = \text{prob}_{\text{start}} \cdot (1 - \text{prob}_{\text{start}})(1 - i)
$$

(4)

Note that in \[3\] and \[4\], the probability refers to the $i$-th individual; whereas the term $FIT$ in \[3\] indicates the array of fitness values for each phenotype $fit \in FIT$.

The crossover operation could be single point or double, meaning that the mating process divides the couple of chromosomes in two parts or three, respectively. As for the mutation operation, only the simple procedure based on a given probability has been implemented.

It is obvious that the GA performance can be strongly dependent on the right choice of the aforementioned parameters: Sec. 3.2 will illustrate an experiment showing the performance achieved by different combinations of the operators.

The entire process is repeated for each iteration (epoch) until the GA converges or reaches the maximum number of iterations allowed. At the end of the execution, the algorithm provides the chromosome having the best fitness value (the lower) among the population. The GA runs for a maximum number of epochs defined by a threshold number $\text{maxIt}$ given as parameter. To speed up the execution, the algorithm features an auto-convergence procedure, which checks whether the best chromosome fitness value varies during epochs. If the best value in the $i$-th iteration does not change for the next $k$ iterations, the GA stops its execution and provides this chromosome as the best solution. The term $k$ is a threshold given as parameter. However, if the fitness value keeps changing, the algorithm ends its execution after $\text{maxIt}$ iterations.

### 3. Validation

This section presents three experiments aiming at validating the proposed procedure as well as showing its accuracy compared to others known in literature. In particular: Sec. 3.1 will compare the Polar metrics with other distance methods to validate the fitness function exploited in the GA; Sec. 3.2 will present an experiment to measure the GA accuracy; whereas Sec. 3.3 will compare the results obtained with the proposed genetic procedure with the results produced by one of the newest and most performing least-squares fitting method.

#### 3.1. Fitness Function Validation

The experiment mainly aims at comparing the performances of the Polar metrics and the Voronoi metrics in terms of their accuracy. Since the difference between the two methods occurs when the polygons do not feature a regular size, the analysis focuses only on rectangles. Furthermore, the test also takes into account another different metrics, proposed by Liparulo et al. \[22\] for shape recognition purposes. As already mentioned in Sec. 1 this metrics associates a null distance to the points inside the polygon, which are
considered part of the shape, and coincides with the Voronoi and Polar metrics for the points outside. For this reason, this kind of approach is considered not suitable for fitting trajectories, and the results of the comparison are expected to be less accurate than the Polar (and the Voronoi) metrics.

We used an analytical nonlinear programming solver, implemented by the MATLAB \texttt{fmincon} function. This aims at proving the effectiveness of the metrics independently from the adopted GA, for a general-purpose use. All the three metrics have been used by the \texttt{fmincon} on the same input data, which has been affected by variable level of Gaussian Noise. Then, their outputs have been compared with a “ground truth” solution.

\textbf{Algorithm 7: Fitness Function Validation}
\begin{verbatim}
input : The array of points \( P \) sampled from the ground truth rectangle
input : The array of vertices \( V_{GT} \) of the ground truth rectangle
input : The maximum error \( E_{max} \) affecting \( P \)
output: The matrix of differences \( D_P \) (21 \times 50) between the ground truth rectangle and the one retrieved by Polar metrics
output: The matrix of differences \( D_V \) (21 \times 50) between the ground truth rectangle and the one retrieved by Voronoi metrics
output: The matrix of differences \( D_L \) (21 \times 50) between the ground truth rectangle and the one retrieved by Liparulo metrics

1 begin
2 \( \lambda \leftarrow 0; \)
3 \( D_p[21][50] \leftarrow \emptyset; \)
4 \( D_v[21][50] \leftarrow \emptyset; \)
5 \( D_l[21][50] \leftarrow \emptyset; \)
6 while \( \lambda \leq 1 \) do
7 \( i \leftarrow 1; \)
8 for \( j \leftarrow 1 \) to 50 do
9   \( P_n \leftarrow P + \lambda \cdot \text{randn(size}(P)) \cdot E_{max}; \)
10  \( V_p \leftarrow \text{fmincon(polar}(V_{GT}, P_n)); \)
11  \( V_v \leftarrow \text{fmincon(voronoi}(V_{GT}, P_n)); \)
12  \( V_L \leftarrow \text{fmincon(liparulo}(V_{GT}, P_n)); \)
13 for \( k \leftarrow 1 \) to 4 do
14   \( D_P[i][j] \leftarrow D_P[i][j] + \sqrt{(V_P[k] - V_{GT}[k])^2}; \)
15   \( D_V[i][j] \leftarrow D_V[i][j] + \sqrt{(V_V[k] - V_{GT}[k])^2}; \)
16   \( D_L[i][j] \leftarrow D_L[i][j] + \sqrt{(V_L[k] - V_{GT}[k])^2}; \)
17  \( D_P[i][j] \leftarrow D_P[i][j]/4; \)
18  \( D_V[i][j] \leftarrow D_V[i][j]/4; \)
19  \( D_L[i][j] \leftarrow D_L[i][j]/4; \)
20  \( \lambda \leftarrow \lambda + 0.05; \)
21  \( i \leftarrow i + 1; \)
\end{verbatim}
3.1.1. Experiment Description

The experiment starts with a point cloud $P$ generated by sampling a given rectangle $R_{GT}$, defined by an angle $\theta$ for the orientation (0°) and the two edges height $h$ and width $w$ (10 × 30 cm). The center of the rectangle has been imposed as (0, 0), removing two negligible variables from the optimization problem. $R_{GT}$ is used as “ground truth” solution since all the points in $P$ have distance 0 from it.

During the experiment, $P$ has been affected by noise generated with a Gaussian Distribution, having than zero mean and unit variance. In Sec. 4 real examples featuring post-stroke patients will show clearly that the errors made by human beings are heteroscedastic; despite this, the use of a Gaussian point cloud model refers to a general-purpose use of the method, which is not necessarily specific as the proposed rehabilitation scenario. As the base case of continuous probability distributions, a normal distribution ensures to obtain the optimal solution from a precise fitting algorithm, which is the purpose of this study.

The experiment is detailed in Algorithm 7. $P_n$ is the point cloud affected by noise, evaluated in statement 8, where:

- $\text{randn}$ is a MATLAB function that provides Normally distributed random numbers knowing the number of points in $P$;
- $E_{\text{max}}$ is the maximum error value affecting $P$ (set to 1/10 of the rectangle diagonal); and
- $\lambda$ is a number that ranges from 0 to 1, with a step of 0.05, to affect $P$ with increasing values of noise from 0 to $E_{\text{max}}$, for a total of 21 iterations.

The term $E_{\text{max}}$ is purposely limited by the size of the diagonal to avoid $P_n$ to be too different from $R_{GT}$, otherwise the procedure would not generate reliable solutions for high values of $\lambda$.

For each value of $\lambda$, the solver is executed 50 times. At the end of the experiment, a number of $21 \cdot 50 = 1050$ rectangles per metrics will be generated. Fig. 6 shows the three different rectangles $R_{GT}$ with the point clouds $P_n$ for three different $\lambda$ values (0, 0.5, and 1). For each rectangle, its difference with $R_{GT}$ is evaluated as the mean value of the euclidean distances of their vertices (statements 13-19).

3.1.2. Results and Discussion

The final results are presented as average and standard deviation for each value of $\lambda$ (hence over 50 samples). The values are named $d_P$, $d_V$, and $d_L$ for Polar, Voronoi, and Liparulo metrics, respectively. Fig. 7 and Tab. 1 show the trend of the results.
The presented values show exponential trends for $d_P$ and $d_V$, which increase with higher values of the disturbance introduced in $P$. By fitting the results with an exponential curve, the trend followed by the metrics are expressed in (5) for Polar, and in (6) for Voronoi. On the other hand, Liparulo metrics shows a more linear trend which can be approximated as in (7).

\[
f(\lambda) = 0.07 \cdot \exp(3.73 \cdot \lambda) \quad (5)
\]

\[
f(\lambda) = 0.32 \cdot \exp(3.16 \cdot \lambda) \quad (6)
\]

\[
f(\lambda) = 106.3 \cdot \lambda - 2.6 \quad (7)
\]

The approximations (5), (6), and (7) are valid only in the chosen interval $[0, E_{\text{max}}]$, which is limited by the size of $R_{GT}$. In this range, the curve fitted from $d_P$ has been observed to be always smaller then $d_V$ and $d_L$.

In order to determine the statistical difference between the performance of the proposed algorithm against Voronoi and Liparulo metrics, a $3 \times 3$ mixed ANOVA was conducted using the metrics and the noise level as factors (“low” for $\lambda \in [0.00 - 0.35]$, “medium” for $\lambda \in [0.40 - 0.65]$, and “high” for $\lambda \in [0.70 - 1.00]$).

**Figure 7**: Graphic results of the evaluation. The plot focuses especially on the difference between Polar and Voronoi metrics. The method proposed by Liparulo does not fit the requirements, showing how the general shape recognition methods exploited in literature are not suitable with trajectory fitting problems.
between subject factor). The Green-House-Geisser method has been used for correcting the assumption on sphericity condition. Since a significant interaction effect was detected ($F(4, 36) = 42.13, p < 0.01$), three more focused ANOVA tests were conducted by fixing a different level of noise each time. When a significant main effect of focused ANOVA was detected, the pairwise comparisons were conducted using the Bonferroni correction method to overcome the issue of protection from multiple univariate tests, and thus to avoid the occurrence of the type I error. We found that, when the level of noise is “medium” or “high”, the proposed algorithm significantly outperforms the other two methods ($p < 0.01$). In the “low” noise condition the proposed algorithm performs significantly better than the Liparulo metrics ($p < 0.01$), and better than the Voronoi method ($p < 0.05$).

The findings of this study might be interpreted as the proof that the proposed method is more efficient than the Voronoi one, showing higher accuracy in addressing more noisy datasets. On the other hand, the results observed for Liparulo metrics prove that this kind of approach is not suitable for the problem of fitting a polygon outline, while it remains a performing method for shape recognition.

### 3.2. Genetic Algorithm Accuracy

The experiment aims at comparing the accuracy of the GA, measuring the fitness value achieved in different runs with different settings. The evaluation has been performed with different combinations of the GA parameters to identify the best performing settings.

#### 3.2.1. Experiment Description

The algorithm has been tested on 50 different shapes for each type of polygon (rectangles, equilateral triangles, and pentagons), randomly generated within a workspace of $40 \times 40$ cm.

The datasets of points used in the test have been obtained by downsampling the 150 shapes. Each dataset is therefore composed by points having a distance equal to 0 with the corresponding shape.

The algorithm has been executed 10 times for each shape, featuring 500 solutions for each of the 150 polygons, totaling 1500 runs for each setting. For each single instance, the test checked different combinations of genetic parameters within the following discrete values:

- number of chromosome in the population $n_c \in \{20, 40, 60\}$;
- type of selection $s \in \{\text{roulette} = R, \text{tournament} = T\}$;
- type of crossover $c \in \{\text{single} = S, \text{double} = D\}$; and
- mutation probability $p_m \in \{0.2, 0.4\}$.

Both $n_c$ and $p_m$ have been limited to finite discrete values to reduce the overall execution time.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$d_F [\text{cm}]$</th>
<th>$d_V [\text{cm}]$</th>
<th>$d_L [\text{cm}]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00 – 0.35</td>
<td>0.15 ± 0.08</td>
<td>0.34 ± 0.26</td>
<td>16.10 ± 11.26</td>
</tr>
<tr>
<td>0.40 – 0.70</td>
<td>0.55 ± 0.23</td>
<td>1.92 ± 0.68</td>
<td>55.03 ± 10.50</td>
</tr>
<tr>
<td>0.75 – 1.00</td>
<td>1.90 ± 0.64</td>
<td>5.22 ± 1.21</td>
<td>91.10 ± 9.37</td>
</tr>
</tbody>
</table>

Table 1: Numeric results of the evaluation. The data has been grouped in three classes of $\lambda$.  

The findings of this study might be interpreted as the proof that the proposed method is more efficient than the Voronoi one, showing higher accuracy in addressing more noisy datasets. On the other hand, the results observed for Liparulo metrics prove that this kind of approach is not suitable for the problem of fitting a polygon outline, while it remains a performing method for shape recognition.
which has been observed being the same for all the polygons: roulette which for an optimal method should be equal to 0. Tab. 2 highlights in blue the best setting for each shape, with higher number of chromosomes, it is possible to achieve better results with a larger population.

### 3.3. Polygon Fitting Validation

The experiment aims at comparing the performances of the GA featuring the Polar metrics and the least-squares method proposed by Sinnreich [17] in terms of their accuracy, in order to validate the proposed procedure as a fitting method. Both the methods propose an algorithm for fitting a polygon to a set of given points in the plane; however, as already mentioned in Sec. 1, the referenced method computes a continuous function approximating a polygon, known as hypotrochoid, which does not feature any vertex.

### 3.3.1. Experiment Description

The experiment aims at comparing the fitting results over three geometrical shapes (a rectangle, an equilateral triangle, and a pentagon) and a set of points defined in the dataset presented in Sinnreich’s work. The fitted polygons are compared to a ground truth shape from which the points are sampled with additional noise (further details are provided in Sinnreich’s work).

Since the evaluation proposed by our work is based on the difference between shape vertices, the comparison between the two methods is not straightforward. In the following experiment, the vertices of the

| $n_c$ | $p_m$ | $s = R$ | $s = R$ | $s = T$ | $s = T$
|------|------|-------|-------|-------|-------|
|      |      | $c = S$ | $c = D$ | $c = S$ | $c = D$
|      |      |       |       |       |       |
| triangle | 20   | 0.0204 ± 0.0046 | 0.0163 ± 0.0052 | 0.0172 ± 0.0024 | 0.0131 ± 0.0002 |
|      | 20   | 0.1408 ± 0.0115 | 0.1444 ± 0.0067 | 0.1759 ± 0.0175 | 0.1831 ± 0.0026 |
|      | 40   | 0.0082 ± 0.0011 | 0.0107 ± 0.0015 | 0.0136 ± 0.0019 | 0.0091 ± 0.0013 |
|      | 40   | 0.0918 ± 0.0028 | 0.0950 ± 0.0106 | 0.1010 ± 0.0115 | 0.1049 ± 0.0092 |
|      | 60   | 0.0065 ± 0.0009 | 0.0059 ± 0.0008 | 0.0088 ± 0.0025 | 0.0084 ± 0.0012 |
|      | 60   | 0.0711 ± 0.0060 | 0.0720 ± 0.0041 | 0.0780 ± 0.0015 | 0.0796 ± 0.0005 |
| pentagon | 20   | 0.0507 ± 0.0065 | 0.0378 ± 0.0045 | 0.0602 ± 0.0080 | 0.0619 ± 0.0082 |
|      | 20   | 0.2698 ± 0.0299 | 0.2861 ± 0.0304 | 0.2862 ± 0.0309 | 0.2973 ± 0.0302 |
|      | 40   | 0.0267 ± 0.0024 | 0.0235 ± 0.0017 | 0.0324 ± 0.0037 | 0.0306 ± 0.0043 |
|      | 40   | 0.2400 ± 0.0265 | 0.2423 ± 0.0264 | 0.2239 ± 0.0238 | 0.2288 ± 0.0269 |
|      | 60   | 0.0170 ± 0.0016 | 0.0145 ± 0.0013 | 0.0177 ± 0.0023 | 0.0180 ± 0.0017 |
|      | 60   | 0.2177 ± 0.0218 | 0.2241 ± 0.0260 | 0.1965 ± 0.0206 | 0.1964 ± 0.0223 |
| rectangle | 20   | 0.0177 ± 0.0011 | 0.0231 ± 0.0025 | 0.0316 ± 0.0011 | 0.0234 ± 0.0006 |
|      | 20   | 0.1485 ± 0.0105 | 0.1447 ± 0.0077 | 0.1753 ± 0.0035 | 0.1741 ± 0.0119 |
|      | 40   | 0.0097 ± 0.0000 | 0.0080 ± 0.0004 | 0.0115 ± 0.0004 | 0.0144 ± 0.0010 |
|      | 40   | 0.1105 ± 0.0074 | 0.1143 ± 0.0029 | 0.1229 ± 0.0024 | 0.1198 ± 0.0087 |
|      | 60   | 0.0080 ± 0.0004 | 0.0074 ± 0.0004 | 0.0113 ± 0.0010 | 0.0097 ± 0.0005 |
|      | 60   | 0.1005 ± 0.0098 | 0.0952 ± 0.0023 | 0.1018 ± 0.0050 | 0.1044 ± 0.0054 |

Table 2: Results of the Genetic Algorithm Accuracy Evaluation (in cm).
polygons computed by Sinnreich’s method have been estimated from the curves by joining the lines approximating the polygon edges.

The GA was executed with the settings featuring the best results for each shape, which are indicated in Sec. 3.2.

3.3.2. Results and Discussion

The final results are presented as average and standard deviation of the distance between the vertices of the ground truth and the fitted polygon for each strategy. The values are named $d_{GA}$ and $d_{S}$ for the genetic procedure and Sinnreich’s method, respectively. Fig. 8, Fig. 9, and Fig. 10 show the graphical difference between the fitted results, whereas Tab. 3 presents the numerical values.

<table>
<thead>
<tr>
<th>Shape</th>
<th>$d_{S}$ [cm]</th>
<th>$d_{GA}$ [cm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rectangle</td>
<td>0.712 ± 0.534</td>
<td>1.312 ± 0.474</td>
</tr>
<tr>
<td>triangle</td>
<td>1.997 ± 0.268</td>
<td>1.809 ± 0.651</td>
</tr>
<tr>
<td>pentagon</td>
<td>1.675 ± 1.332</td>
<td>1.670 ± 0.544</td>
</tr>
</tbody>
</table>

**Table 3:** Numeric results of the evaluation.

The result show that the two methods feature similar accuracy. The proposed genetic approach succeed to perform slightly better than the regression function in the cases of triangle and pentagon, whereas the rectangle deceived the procedure because of the few number of points related to the right-height edge.

The findings of this study might be interpreted as the proof that the proposed genetic procedure is a valid method for fitting polygons, as long as the dataset of points is consistent with the shape to be fitted.

![Figure 8: Graphic results of the evaluation for a rectangle.](image)
Figure 9: Graphic results of the evaluation for a triangle.

Figure 10: Graphic results of the evaluation for a pentagon.
4. Application in Neurorehabilitation

This Section will address a preliminary experimentation in the neurorehabilitation field. The algorithm has been used to gather information about patients’ performance during the execution of a specific task.

In particular, after a therapy session in which a patient is asked to follow a polygonal trajectory with a robotic device, the GA analyzes the drawn trajectory, collects a point cloud by sampling the robot movements, and extracts the polygon that fits with the execution. The differences between the reference and the drawn trajectory will serve as an objective numeric index to describe patients’ status.

Results achieved by healthy subjects will be compared with the ones obtained by post-stroke survivors. The purpose is to assess whether the proposed solution can identify a patient and quantify his/her disability level.

4.1. System Setup

The proposed neurorehabilitation system consists in a simple pursuing task requiring inter-joint coordination of elbow and shoulder. A previous work of the authors [36] describes a similar and more complete setup, which exploits the genetic procedure detailed in Sec. 2. The system scheme is reported in Fig. 11.

The setup is composed by:

1. the Trackhold [37], a passive robotic device for motor rehabilitation of the upper limb, shown in Fig. 12 and
2. a virtual environment presenting the rehabilitation task, displayed on a 60” monitor placed in front of the subjects.

The patient is asked to follow a target that moves clockwise along a predefined polygonal trajectory. The GA is exploited to retrieve the patient’s trajectory from the collection of points sampled by the robot, knowing the geometrical shape given as the path to follow. The settings indicated in Sec. 3.2 are used to run the algorithm.
Figure 13: Shape fitting results on a healthy subject and patient 1 for different polygonal trajectories.
Figure 14: Shape fitting results on patient 2 and 3.
4.2. Experiment Description

The subjects were asked to undergo the task on three different shaped trajectories, performing five laps for each figure. The virtual polygons used in the experiment were:

- a rectangle having size of $40 \times 20$ cm;
- an equilateral triangle having edges of $21$ cm; and
- a pentagon having edges of $14$ cm.

All the polygons lied on a virtual plane, which was oriented as the monitor facing the subjects. The polygons had an orientation angle equal to zero, so they were orthogonally presented to the subjects’ field of view. The polygon center was placed to be on the same sagittal plane of the patient’s arm.

Moreover, the constant linear velocity of the target to be followed was $0.05$ m/s. This slow value has been selected to allow the subjects performing the task with more comfort.

4.3. Participants

The system has been tested on 20 healthy subjects (18 males and 2 females; $29 \pm 4$ yrs old), and on 3 ischaemic post-stroke patients (all males, $62 \pm 12$ yrs old). All the subjects were asked to perform the experiment in the same environment and with the same settings.

Fig. 13(a) shows the graphical shapes drawn by one of the healthy subjects, whereas Fig. 13(b), 14(a), and 14(b) report the execution of the three patients. All the figures report the reference trajectory (dotted line) and the polygon fitted by the algorithm (blue line).

4.4. Evaluation

The evaluation of patients’ performance is addressed considering:

- the convex polygon used as a trajectory to be followed, having $m$ vertices $v_T \in V_T$;
- the convex polygon generated by the GA as the best phenotype, having $m$ vertices $v_{GA} \in V_{GA}$; and
- the movements of the subject, sampled and collected in $n$ points $p \in P$.

These parameters are analyzed for the estimation of the patient’s performance indices, a set of numerical scores indicating the accuracy of the execution. Such an index requires the analysis of the accuracy in replicating the given polygon and the smoothness of the trajectory drawn by the patient. In particular, the procedure extracts:

- $k_1$, which is the mean difference between the vertices $V_{GA}$ of the fitted polygon and the vertices $V_T$ of the original trajectory (8):

$$k_1 = \frac{\sum_{i=1}^{m} \sqrt{(v_{GA}, i - v_{T}, i)^2}}{m}$$

(8)
• $k_2$, which is the fitness value of the best phenotype retrieved by the GA, and represents the mean value of the error occurring between $P$ and the fitted shape defined by $V_{GA}$ (9); and

$$k_2 = \frac{\sum_{i=1}^{n} \text{fitness}(p_i, V_{GA})}{n}$$

(9)

• $k_3$, which is the standard deviation of the fitness value previously calculated (10).

$$k_3 = \sqrt{\frac{\sum_{i=1}^{n} (p_i - k_2)^2}{n}}$$

(10)

The first index $k_1$ represents the differences between the two polygons. Hence, it shows how closely the patient has performed the task, taking into account the sizes of the polygons and their orientation in the space. However, this index is not sufficient to provide a comprehensive description of the performance. It is possible to achieve low values of $k_1$ in the case in which the patient performs a non-smooth trajectory along one of the polygon edges. In fact, a set of points sampled from a sine wave-shaped trajectory can be associated to the straight line passing by its medium value, which provides a fitted shape that deceives $k_1$. To overcome this issue, we introduced the last two indices $k_2$ and $k_3$, which are related to the patient’s accuracy with respect to the fitted polygon.

It is worth to point out that the values of these indices get lower as the patient’s abilities improve.

4.5. Results and Discussion

The indices $k_1$, $k_2$, and $k_3$ are presented as averages over a number of items (that is the number of vertices for $k_1$ and the number of points for $k_2$ and $k_3$), so they are implicitly normalized such that it is possible to compare the results between different subjects and different polygons as well. We found that observing the actual error is more significant than considering, for instance, a value normalized on the size of the shape: executing the task on a very small shape could be too demanding in terms of precision also for a healthy subject, and even the smallest error would be considered as lack of performance. Based on the algorithm procedure and the size of the reference polygons, the indices are measured in cm.

The average healthy subjects’ indices are used as the reference of correct task execution. Tab. 4 reports the mean values of the three performance indices for the healthy subjects’ executions, whereas the results of the patients are reported in Tab. 5. Finally, Tab. 6 shows the ratios between patients and healthy subjects’ performances.

As expected, all the healthy subjects performed the tasks without markedly deviating from the values shown in Tab. 4. On the other hand, the indices of each patient feature higher values. The mean values reported in Tab. 5 are calculated over all the patients for each shape type, but it might be interesting to observe how the patients perform the task differently from each other. In particular, the standard deviations

<table>
<thead>
<tr>
<th>Healthy Subjects [cm]</th>
<th>$k_1$</th>
<th>$k_2$</th>
<th>$k_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rectangle</td>
<td>0.384 ± 0.333</td>
<td>0.312 ± 0.078</td>
<td>0.252 ± 0.062</td>
</tr>
<tr>
<td>triangle</td>
<td>0.350 ± 0.270</td>
<td>0.253 ± 0.042</td>
<td>0.202 ± 0.037</td>
</tr>
<tr>
<td>pentagon</td>
<td>0.415 ± 0.302</td>
<td>0.298 ± 0.075</td>
<td>0.225 ± 0.048</td>
</tr>
</tbody>
</table>

Table 4: Average performance over 20 healthy subjects.
Impaired Subjects [cm]

<table>
<thead>
<tr>
<th></th>
<th>rectangle</th>
<th>triangle</th>
<th>pentagon</th>
</tr>
</thead>
<tbody>
<tr>
<td>patient 1</td>
<td>0.116</td>
<td>0.610</td>
<td>0.871</td>
</tr>
<tr>
<td></td>
<td>0.439</td>
<td>0.461</td>
<td>0.567</td>
</tr>
<tr>
<td></td>
<td>0.367</td>
<td>0.396</td>
<td>0.397</td>
</tr>
<tr>
<td>patient 2</td>
<td>1.216</td>
<td>0.684</td>
<td>0.345</td>
</tr>
<tr>
<td></td>
<td>0.944</td>
<td>0.854</td>
<td>0.996</td>
</tr>
<tr>
<td></td>
<td>0.769</td>
<td>0.648</td>
<td>0.792</td>
</tr>
<tr>
<td>patient 3</td>
<td>1.615</td>
<td>1.876</td>
<td>1.789</td>
</tr>
<tr>
<td></td>
<td>0.787</td>
<td>0.720</td>
<td>0.948</td>
</tr>
<tr>
<td></td>
<td>0.701</td>
<td>0.584</td>
<td>0.800</td>
</tr>
<tr>
<td>avg</td>
<td>0.982 ± 0.634</td>
<td>0.723 ± 0.021</td>
<td>0.612 ± 0.176</td>
</tr>
<tr>
<td></td>
<td>1.056 ± 0.580</td>
<td>0.678 ± 0.163</td>
<td>0.543 ± 0.163</td>
</tr>
<tr>
<td></td>
<td>1.002 ± 0.597</td>
<td>0.837 ± 0.192</td>
<td>0.663 ± 0.188</td>
</tr>
</tbody>
</table>

Table 5: Performance of patients.

<table>
<thead>
<tr>
<th></th>
<th>rectangle</th>
<th>triangle</th>
<th>pentagon</th>
</tr>
</thead>
<tbody>
<tr>
<td>patient 1</td>
<td>0.302</td>
<td>1.741</td>
<td>2.097</td>
</tr>
<tr>
<td></td>
<td>1.405</td>
<td>1.819</td>
<td>1.905</td>
</tr>
<tr>
<td></td>
<td>1.456</td>
<td>1.964</td>
<td>1.763</td>
</tr>
<tr>
<td>patient 2</td>
<td>3.164</td>
<td>1.952</td>
<td>0.831</td>
</tr>
<tr>
<td></td>
<td>3.021</td>
<td>3.371</td>
<td>3.344</td>
</tr>
<tr>
<td></td>
<td>3.053</td>
<td>3.210</td>
<td>3.519</td>
</tr>
<tr>
<td>patient 3</td>
<td>4.201</td>
<td>5.354</td>
<td>4.310</td>
</tr>
<tr>
<td></td>
<td>2.519</td>
<td>2.843</td>
<td>3.182</td>
</tr>
<tr>
<td></td>
<td>2.781</td>
<td>2.894</td>
<td>3.552</td>
</tr>
</tbody>
</table>

Table 6: Ratio between patients and healthy subjects. The results are dimensionless.

are on average almost three times higher than the ones performed by the healthy subjects, showing how less homogeneous the patients are.

The ratios reported in Tab. 5 show that it is possible to discriminate the performance of a subject with arm impairment from a healthy one: the patients performed the task approximately three times worse than the others. Furthermore, it is also possible to observe a difference between the patients themselves. Regarding triangles, patient 3 was slightly more smooth on his movements than patient 2 (on \( k_2 \) and \( k_3 \)), but this one was more accurate on drawing the reference trajectory (on \( k_1 \)). It follows that patient 3 was not able to move his arm in the whole application workspace, suggesting that his degree of motor impairment was higher than patient 2. On the other hand, the first patient performed the tasks with higher accuracy, showing that his degree of impairment is smaller than the other patients. It is also interesting to note how the indices compensate each other: the execution over the rectangle for patient 1 presents a very low value of \( k_1 \) (even lower than the healthy subjects), but the indices \( k_2 \) and \( k_3 \) suggest that the performance was not smooth enough to be considered as a healthy subject - as also graphically shown in Fig. 13(b). This observation remarks what defined in Sec. 4.4 regarding \( k_1 \).

The experiment showed how these indices can provide both intra and inter-patient indication of the...
accuracy performed, which reflects the motor skills of the subject. Moreover, the ratios reported in Tab. 6 might also be significant to introduce a threshold value and separate real patients from healthy people, considering a ratio equal or less than 1 as indication for good health. However, in order to provide a stronger threshold, more clinical studies might be required.

<table>
<thead>
<tr>
<th></th>
<th>healthy (avg)</th>
<th>patient 1</th>
<th>patient 2</th>
<th>patient 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>rectangle</td>
<td>122.93</td>
<td>194.17</td>
<td>188.30</td>
<td>132.37</td>
</tr>
<tr>
<td>triangle</td>
<td>77.79</td>
<td>81.20</td>
<td>98.12</td>
<td>98.20</td>
</tr>
<tr>
<td>pentagon</td>
<td>77.40</td>
<td>136.40</td>
<td>87.00</td>
<td>110.60</td>
</tr>
</tbody>
</table>

Table 7: Experiment execution time

Finally, it is important to remark that the evaluation strategy proposed in Sec. 4.4 does not consider an important detail: the time of the execution. As shown in Tab. 7, the healthy subjects performed the task faster than the patients although the target speed was set to a slow fixed value (0.05 m/s). The aim of the proposed experimentation is to show how the indices introduced herein, which are based on basic geometrical analysis, provide clear differences between healthy and impaired subjects.

5. Conclusion and Future Works

In this paper, we proposed a method for convex-polygon fitting from a set of 2D points describing a trajectory. We defined a new metrics to evaluate the distance between a generic polygon and a set of points in the plane, featuring linear time complexity. The strategy has been implemented as the fitness function of a Genetic Algorithm, handling the polygon fitting as an optimization problem.

The proposed application is exploited in a robot-based neurorehabilitation scenario, used as an evaluation method to provide quality indices about post-stroke patients’ performance. During tests, the algorithm was able to quantify the impairment degree of the subjects, identifying different cases of disability in arm functionality. However, although the conducted experiments have proven the significance of the indices, these numerical scores may not be self explanatory to doctors; nevertheless, their final purpose is to be used as parameters for future active therapy exercises, rather than a new objective clinical scale. Furthermore, even though the fitting algorithm has been proposed only for polygonal-trajectory pursuing tasks, the same procedure can be extended to different reaching/pursuing exercises: tasks involving geometrical shapes that model complex, curve, or linear paths can benefit from this kind of approach.

Future works will include the exploitation of the proposed method in the same therapy scenario featuring an active robot. Since the movements of the patients’ impaired limb are often limited, they might require the guidance of a device providing “assistance-as-needed”. The correct amount of the assistance to provide will be defined from the indices retrieved by the algorithm, aiming at leading the patient reaching the target and, at the same time, enhancing his/her efforts. In such a system, the task difficulty will be suitably tailored to the patient in accordance with his/her motor functionality, maximizing the efficiency of the therapy.

Acknowledgment

This work has been partially supported by RONDA project, funded by Regione Toscana, Italy, within the FAS ‘Salute’ 2014 program, code 4042.16092014.066000065. The authors would also like to thank Dr. Michele Barsotti for the support in the statistical analysis reported in Sec. 3.1.2.
References


**Fabio Stroppa**

Fabio Stroppa (Eng.) received the B.S. and M.S. degrees in Computer Science Engineering from Politecnico di Bari, Bari, Italy, in 2011 and 2013, respectively. He is currently a PhD student in Perceptual Robotics at PERCRO Laboratory, TeCIP Institute, Scuola Superiore Sant'Anna, Pisa, Italy. His publications and his main research interests deal with computer vision for control purposes of robotic devices, robotic-based neurorehabilitation, bioinformatics, virtual reality and artificial intelligence.

**Claudio Loconsole**

Claudio Loconsole (Eng., Ph.D.) received his B.S. and M.S. in Computer Science Engineering respectively from Polytechnic University of Bari, Italy, in 2007 and Polytechnic University of Turin, Italy, in 2009. In 2012, he received the Ph.D. in Innovative Technologies from Scuola Superiore Sant’Anna, Pisa, Italy. Currently, he is an Assistant Professor at Department of Electrical and Information Engineering (DEI), Polytechnic University of Bari (Italy) and Professional Affiliate at Perceptual Robotics (PERCRO) laboratory, TeCIP Institute, Scuola Superiore Sant’Anna. He has been awarded with several international acknowledgments and prizes and is author of several research papers in international journals and conference proceedings. His research interests include human-machine and human-robot interaction, assistive technologies, computer vision, motion planning and neurorehabilitation.

**Antonio Frisoli**

Antonio Frisoli (Eng., Ph.D.) is Associate Professor of Mechanical Engineering at Scuola Superiore Sant’Anna, where he is currently head of the Human-Robot Interaction area at PERCRO laboratory - TeCIP and former chair of the IEEE Technical Committee on Haptics. He received his Ph.D. (2002) with honors in Industrial and Information Engineering from Scuola Superiore Sant’Anna, Italy and his M.Sc. (1998) in Mechanical Engineering, from the University of Pisa-Italy. Antonio Frisoli’s research interests are in the field of design and control of haptic devices and robotic systems, rehabilitation robotics, advanced human computer interfaces and kinematics.
ARTICLE HIGHLIGHTS

Convex Polygon Fitting in Robot-Based Neurorehabilitation

- A procedure for convex polygon fitting is described as an optimization problem, minimizing the distance between the polygons proposed as solutions and the set of 2D points given as input.
- A Genetic Algorithm is implemented to solve the optimization problem.
- A novel metrics to evaluate the distance between a point and a polygon outline is proposed, and used as the fitting function of the Genetic Algorithm.
- An experiment on the algorithm accuracy is described, to assess the efficiency of the method.
- The algorithm is applied in a neuro-rehabilitation therapy, to extract performance indices from the shape drawn by post-stroke patients during motor tasks.