Gaussian path model library for intuitive robot motion programming by demonstration

Samuli Soutukorva[1], Markku Suomalainen[1], Martin Kollingbaum[2] and Tapio Heikkilä[1]

Abstract—**This paper presents a system for generating Gaussian path models from teaching data representing the path shape. In addition, methods for using these path models to classify human demonstrations of paths are introduced. By generating a library of multiple Gaussian path models of various shapes, human demonstrations can be used for intuitive robot motion programming. A method for modifying existing Gaussian path models by demonstration through geometric analysis is also presented.**

Index Terms—**Learning and Adaptive Systems, Machine learning, Intelligent and Flexible Manufacturing**

I. INTRODUCTION

Programming by Demonstration (PbD) is one of the commonly proposed solutions for intuitive robot programming, where a user *shows* the desired trajectory, or skill, to a robot [1]. PbD, thus, has the potential to enable domain experts with limited expertise in robotics to teach a robot what actions to perform, purely through demonstrations, allowing a wider use of robotics across multiple industries. In general, a multitude of demonstrations is required to provide the amount of data for robotic systems to properly learn a demonstrated trajectory. Domain experts, however, may prefer robotic systems where one demonstration is sufficient for the robot to deploy a suitable skill that properly imitates the demonstrated trajectory.

In this paper, an approach for programming robot paths with a single demonstration is presented that utilises pre-trained models of prototypical movements a robot may perform (Gaussian path models) and methods of adapting and parameterising such models according to a single demonstration by a domain expert, in order to generate the required robot skill. Gaussian path models are general-purpose prototypical movements (such as, for example, half-circle and other path segments, see Fig. 3) and are generated themselves in a pre-training step, where the training data is derived from either demonstrations or synthetically produced. A library of these path models can then be used in a programming-by-demonstration effort, where a single demonstration provided by a domain expert is used to retrieve the best-fit path model from such a library and parameterize this model with respect to the demonstration.

The contribution of this paper is a set of algorithms for building such a library of Gaussian path models, as well as for matching demonstrations to path models and adapting them for a specific programming purpose. It is also shown how path models can be efficiently modified by a domain expert to create

[1] VTT Technical Research Centre of Finland Ltd, Oulu, Finland `firstname.lastname@vtt.fi`
[2] `mjkollingbaum@hotmail.com`

the required path data for programming a robot, without the need to train a new path model.

II. RELATED WORK AND BACKGROUND

There is a wide variety of ways to extract suitable data from human demonstrations to generate a condensed representation that attempts to capture the demonstrated path, or trajectory. Splines are widely used in robotics in general, as well as in PbD [2]. Lately, different kinds of Gaussian-based models have proven popular; a good overview with examples is provided in [3]. Of the presented models, for example, Hidden Markov Models (HMM) [4] and Gaussian mixture model-Gaussian Mixture Regression (GMM-GMR) have been used for motion captured data [5]. Even Dynamic Movement Primitives (DMP) have been created from GMM [6]. Further Gaussian models are, for example, Infinite Gaussian Mixture Models (IGMM) [7]. Several proposals exist to generate libraries such as in this paper. In [8], a dynamic movement primitive (DMP) library is used to segment trajectories consisting of sequences of movement primitives. Methods for segmenting human demonstrations into movement primitives and compiled to movement primitive libraries have been proposed [9], [10]. Position and force-torque data from human demonstrations have been utilized to recognize predefined skills from a skill library [11], [12]. A common theme to all the mentioned Gaussian-based methods is that a suitable number of keypoints characterizing a path is required to preserve the shape of the movement. However, details regarding the identification of keypoints is often neglected in these papers. For time-variant demonstrations, methods such as Non-Maximum Suppression algorithm (NMS) has been proposed [13]. In contrast, the proposed method uses a Gaussian Hidden Markov Model (GHMM) based approach, and presents practical methods for managing the number of keypoints. Moreover, we present a way for on-the-fly modifications, a method proposed before, however, with different representations and without focusing on the keypoints [2], [14].

III. METHODS

In the approach presented in this paper, a Gaussian path model is created from multiple teaching data sets of path data. These teaching data sets are first brought into a canonicalized form (III-A) and then reduced to an approximation of the original demonstration (III-B) as sequences of path keypoints. With this data, a Gaussian model of the path is created from the means and covariances of the path keypoints (III-C). Such a Gaussian model can then be used to classify demonstrations

(III-D) and utilized in intuitive robot programming. This approach shares similarities with the construction of GHMM, however, it is limited to linear sequences only.

A. Canonicalization

In order for a path model to become flexibly utilizable for arbitrary path recognition and robot programming, the path model must be agnostic with regards to scale, position, and the orientation of the path data. This is achieved through storing the path model in a canonicalized form of the path, where the only feature conserved from the teaching data is the shape of the path (Algorithm 1). The canonicalization of the path heavily relies on Principal Component Analysis (PCA). The rotation matrix for re-orienting the path is defined by the eigenvectors of the sample sets of sample data points (Algorithm 1, line 2). To ensure a right-hand coordinate system defined by the eigenvectors, the rotation matrix is checked and adjusted by changing the sign of the third eigenvector (i.e., the Z-axis), if needed. First, the path point data is centralized around the mean of the data set (line 1 in Algorithm 1). Second, the path points are scaled (= normalized) by the standard deviations of the data set as the square roots of the eigenvalues of the data (line 10 in Algorithm 1). The normalized scale is then defined as the inverse of the square root of the dot product of the square roots of the eigenvalues. This equals to the sum of the vectors defined by the eigenvalues. After the centralization and scaling, the data is rotated with the rotation matrix.

Algorithm 1 Canonicalization of 3-D path

    Input: n data sets of 3-D points
    Output: Canonicalized 3-D points
1: Centralize data
2: Rotation matrix $\leftarrow$ PCA(centralized data).eigenvectors
3: **if** left handed coordinate frame **then**
4:     z-axis = -z-axis   ▷ Left-handed → right-handed
5: **end if**
6: **for** each set in data sets **do**
7:     Centralize the set
8:     Calculate eigenvalues of the set
9:     $\sigma_x, \sigma_y, \sigma_z = \sqrt{\text{eigenvalues}}$
10:     scale $= \frac{1}{\sqrt{[\sigma_x, \sigma_y, \sigma_z] \cdot [\sigma_x, \sigma_y, \sigma_z]}}$
11:     Scale the centralized set
12:     Rotate the set (line 2)
13: **end for**
14: Return canonicalized 3-D points

B. Decimation algorithms

As the teaching data for a path can consist of a large amount of points, the path data is decimated to find the set of keypoints as the least amount of data that is still representative of the demonstrated path. This approximation allows the path recognition to be done on a limited number of keypoints while conserving the shape of the path. The approximation of the canonicalized path is generated by two polyline decimation algorithms. The Ramer–Douglas–Peucker algorithm (RDP)

[15] [16] removes points by finding the point furthest from a generated line segment and comparing that distance to a given tolerance parameter. If the tolerance is exceeded, the point is kept and the process is repeated recursively. The Visvalingam–Whyatt algorithm (VW) [17] removes points by generating triangles formed by adjacent points, finding the smallest triangle and comparing its area to a given tolerance parameter. If the tolerance is exceeded, the point is kept and the process is repeated.

These decimation algorithms are used in sequence before generating a Gaussian model of a path from multiple sets of teaching data: RDP is used to find an appropriate level of decimation (i.e., the number of keypoints) for each generated path model. VW is then utilized to decimate the teaching data sets so that each of them contains an identical number of keypoints. This simplifies the generation of the Gaussian path model as the teaching data sets have the same number of keypoints, and also enables correct matching in path recognition.
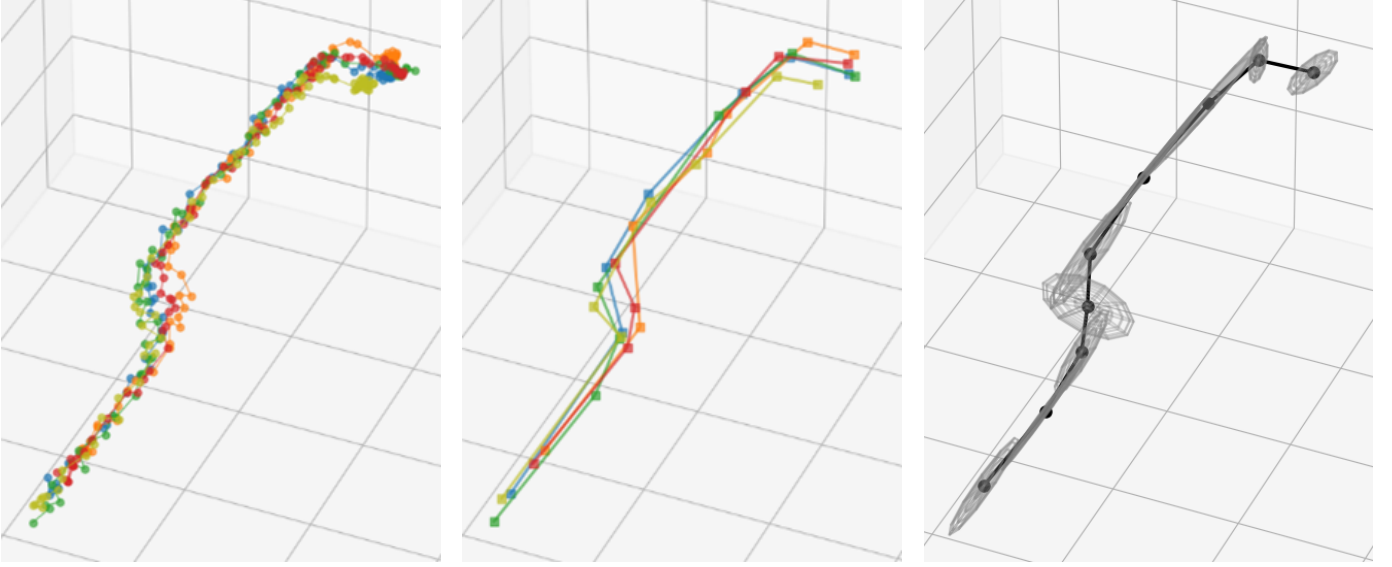
C. Generating Gaussian path models

Fig. 1 presents an example of creating a Gaussian path model from collected teaching data. From multiple sets of teaching data representing the same path (Fig. 1a), a Gaussian model of the path is generated (Algorithm 2). First, the teaching data is canonicalized (Algorithm 1) and filtered with a Gaussian filter (Algorithm 2, line 2). Keypoints for the path model (Fig. 1b) are then retrieved from the teaching data with the RDP and VW decimation algorithms (Algorithm 2, line 3). Mean values and covariances of the path keypoints are calculated from the path keypoints (Algorithm 2, lines 5-7). These mean values and covariances are then used to generate a Gaussian model of the path (Fig. 1c).

For the Gaussian model to be representative of the original path and exploitable in path recognition, the teaching data has to meet some basic requirements. There has to be sufficient variability in the teaching data to create a path model that can be used in path recognition. With too little variability in the teaching path data, the paths cannot be classified as path models, as the path recognition relies on the covariances of the detected keypoints.

The teaching data has to be constructed of more than four separate sets of path points in order to produce variability in all three dimensions for each path model keypoint. For 3-D points, the covariance matrix associated with each keypoint has the rank of three if the number of sample sets is four or more. With only three sets of teaching data, each covariance associated with a Gaussian path model's keypoint would have the rank two. This would reduce the covariance matrix to conform to a plane (and with just two sets of teaching data, the covariance matrix would conform to a line).

As the Gaussian path model is approximated from the original path (III-B), a suitable level of decimation has to be defined. The performance of a Gaussian path model in path recognition varies as the level of decimation is changed. For example, with high enough decimation (i.e., very low keypoint count) the scores obtained in path recognition do not get very

(a) Canonicalized teaching data of the path.   (b) Keypoints from teaching data.   (c) Gaussian model of the path.

Fig. 1: The sequence of creating a Gaussian path model from teaching data.

Algorithm 2 Create a Gaussian model from training data

    Input: 3-D path points
    Output: Gaussian 3-D path model
1: Canonicalized 3-D path ← Canonicalize(3-D path points)
                                             ▷ Algorithm 1
2: Apply Gaussian filter to the canonicalized path data
3: Keypoint count ← RDP(path data, epsilon)
4: Path keypoint sets ← VW(path data, keypoint count)
5: **for** keypoint in path keypoint sets **do**
6:    Calculate the means for each keypoint:
    $mean_i \leftarrow Mean(keypoint\ set_1[keypoint_i],$
                  $keypoint\ set_2[keypoint_i],$
                      $...,$
                $keypoint\ set_{n-1}[keypoint_i],$
                $keypoint\ set_n[keypoint_i])$
7:    Calculate the covariance for each keypoint:
    $cov_i \leftarrow Covariance(keypoint\ set_1[keypoint_i],$
                  $keypoint\ set_2[keypoint_i],$
                    $...,$
                $keypoint\ set_{n-1}[keypoint_i],$
                $keypoint\ set_n[keypoint_i])$
8: **end for**
9: Gaussian model of the path ← Mean values of keypoints, covariances of keypoints
10: Return Gaussian model of the path

---

low with incorrect path models (as the path specific score is the sum of keypoint specific scores). This raises the possibility of an incorrect recognition.

D. Path recognition

The generated Gaussian path models can be utilized in recognizing paths demonstrated by a human and further used as paths for robot motion programming. Algorithm 3 outlines the procedure of recognizing a demonstrated path as a stored Gaussian path model. First, the demonstration of the path is canonicalized (Algorithm 1) and the canonicalized path data is then filtered with a Gaussian filter (Algorithm 3, line 2).

The canonicalized and filtered path demonstration is approximated with the VW decimation algorithm, using the candidate model's parameter for the number of keypoints (Algorithm 3, lines 4-5). This ensures that the decimated and canonicalized data of the demonstration has the same number of keypoints as the Gaussian model it is being compared to. The canonicalized path demonstration can now be compared to existing Gaussian path models (i.e., candidate models). In line 6 of Algorithm 3, the score for each keypoint in the demonstration is defined by calculating the loglikelihood ($\mathcal{L}$) for that keypoint (x), the corresponding keypoint of the candidate model (μ), its covariance matrix ($\sum$) and the rank of the covariance matrix (k) (in our case k equals to 3):

$$\mathcal{L} = \log \frac{\exp(-\frac{1}{2}(x-\mu)^T \sum^{-1} (x-\mu))}{\sqrt{(2\pi)^k \det \sum}} \qquad (1)$$

These keypoint-specific scores are summed up to get the overall score for the path (Algorithm 3, line 8). By comparing the scores of multiple Gaussian path models, the best matching path model can be found for the demonstrated path.

E. Exploitation of Gaussian models and path recognition

Once a demonstrated path is recognized through a specific path model, the keypoints of the matching Gaussian path model can be decanonicalized with Algorithm 4. Here, the

Algorithm 3 Path recognition

    Input: Single demonstration of a 3-D path
    Output: Result Gaussian model

1: Canonical demonstration ← Canonicalize(demonstration
   data set)   ▷ Algorithm 1
2: Apply Gaussian filter to the canonicalized data set
3: **for** each candidate model in model library **do**
4:     Keypoint count ← Candidate model's keypoint count
5:     Demonstration keypoints ← VW(demonstration
   data set, keypoint count)
6:     **for** each keypoint in demonstration keypoints **do**
   Keypoint score ← loglikelihood(demonstration keypoint,
   candidate model keypoint mean, candidate model keypoint
   covariance)   ▷ (1)
7:     **end for**
8:     $Model\ score = \sum_{i=1}^{n} keypoint\ score_i$
9: **end for**
10: Best model score → result model
11: Return result Gaussian model

once canonicalized path model is scaled, transformed, and rotated to the configuration set by the demonstration data. This allows the Gaussian path model's keypoint means to be utilized in robot motion programming, fitted to the demonstration's scale, orientation, and position. Here, the demonstration of the path can be thought of as a localization action for the path keypoints that are fetched from the Gaussian path model through the path recognition procedure.

Algorithm 4 Decanonicalization of 3-D path

    Input: Canonicalized 3-D path, reference scale, reference rotation, reference translation
    Output: Decanonicalized 3-D path

1: **for** each point in path **do**
2:     Revert orientation
3:     Revert scale
4:     Revert centralization
5: **end for**
6: Return decanonicalized path

From multiple Gaussian path models, a path model library, consisting of a variety of paths, can be created and utilized in path recognition, and, furthermore, in robot motion programming by demonstration. In addition to different path shapes, variations of the same path can be generated from the same teaching data to bring versatility to the library. For example, the direction of the path in the model can be reversed allowing the demonstration to be in either direction along the path. More critically, also flipped path models have to be created. As the direction of the eigenvectors (Algorithm 1, line 2) can be one of two, and which cannot be predicted, the orientation of the path can also vary in two different variations per eigenvector. For the path recognition (Algorithm 3) to work robustly, flipped variations for the Gaussian path model are generated for the library. Flipped variations have the rotation matrix

of the canonicalized path model, but rotated 180° around a coordinate axis.

F. Correcting Gaussian path models

As through the decimation step, the created Gaussian path models are based on a very limited number of keypoints, they can be adjusted with relative ease by manipulating the keypoint sequence of the Gaussian path model. In a case, where a path has been demonstrated and recognized to be best represented by a particular Gaussian path model (i.e., the Gaussian path model's keypoints are close to those of a demonstration), the path model's keypoints can be revised by demonstration with a geometric analysis of the demonstrated correction keypoints and the Gaussian path model's keypoints (Algorithm 5). This geometric analysis finds the redundant keypoints from the path model and inserts the correction keypoints to replace them, forming a new path keypoint sequence. This allows for the creation of a new path utilizing the Gaussian path model's keypoints and the revised (or correction) keypoints without having to generate a new path model.

Fig. 2 displays the sequence of the path correction. First, a path must be demonstrated and recognized, and a correction to this path must be demonstrated and approximated as keypoints with RDP (Fig. 2a). After this, a geometric analysis is performed based on the first and last correction keypoints and their closest keypoints of the path model (these are indicated with orange arrows in Fig. 2b). A line segment is generated from these closest keypoints to the following model keypoints (cyan dotted line in Fig. 2b) and the correction keypoint is projected onto that line segment (blue line in Fig. 2b). The redundant keypoints (plotted with red crosses in Fig. 2b) are deduced from whether the projection casts onto the line segment. After this, the redundant keypoints are deleted from the path model and the correction keypoints are inserted, producing a new path (dashed blue line in Fig. 2c). Algorithm 5 displays further details of this geometrical analysis.

IV. RESULTS

A. Generation of the path library

A path model library was constructed for path recognition tests consisting of ten path models. Five of the path models were generic geometric shapes (Fig. 3: parabola, rectangle, quarter-circle, spiral, and half-circle) whose path data was synthetic and generated with software. For each of these synthetic paths, the path data consisted of eight to ten teaching sets. Noise was introduced to the data by adding a normally distributed random variable with a mean of 0 to each path point in the data.

The other five path models were taught by human demonstration on a test object using a tracking tool [18] (Fig. 4). The shapes of these paths (Fig. 3: test paths 1-5) were arbitrarily selected by choosing sections from edges of the test object and marking them as paths. The tracking tool was used to demonstrate these paths and teaching data was collected during the demonstration. Four to five teaching sets were tracked for each of these paths. Due to the small number of teaching

(a) Path model keypoints (black) and correction keypoints (red).

(b) Geometric analysis of path correction.
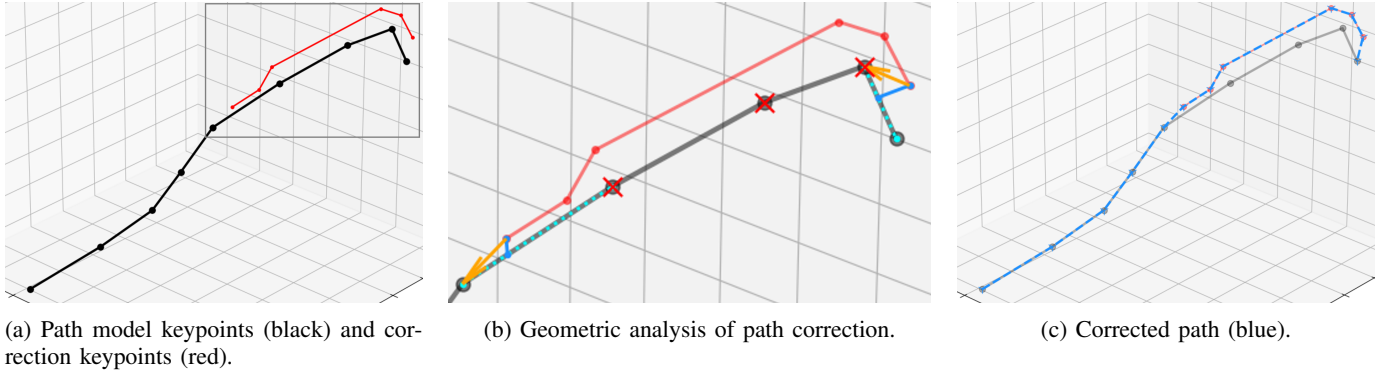
(c) Corrected path (blue).
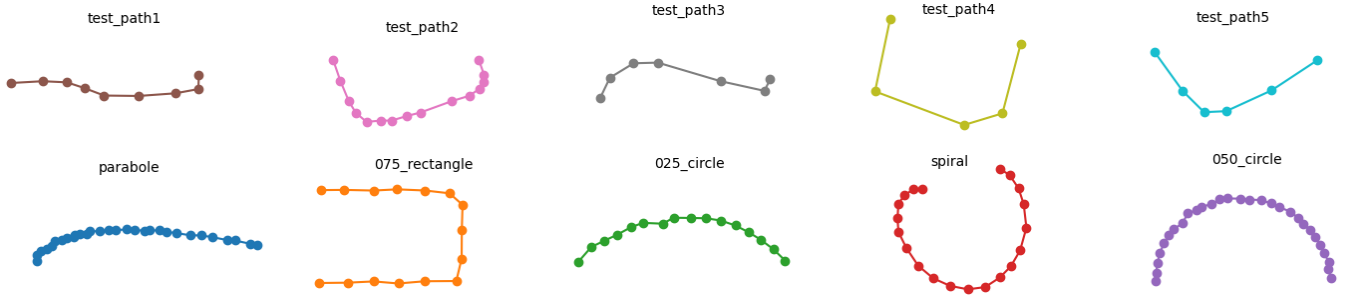
Fig. 2: The sequence of the path correction.

Fig. 3: The keypoints of the Gaussian path models in the path recognition library.

sets, some noise was added to teaching data with a normally distributed random variable. The necessity of having enough variance in the teaching data to create a robust Gaussian model of the path is discussed further in section V-A. Gaussian path

Fig. 4: Test object and tracking tool (NDI Polaris Vega XT) used for path model teaching and path demonstrations.

models were generated from the teaching data with Algorithm 2. A suitable level of decimation for the path shapes was found through iteration: the level of decimation was incremented while tracking the score for a correct recognition and the best score of the incorrect path models. By maximizing the difference between these two scores, the level of decimation for each path was found. In addition to maximizing the difference

of correct and best incorrect scores, it should be considered that a lower level of decimation provides a geometrically more accurate representation of the original path as it is less approximated. This iterative method for finding the level of decimation is discussed further in section V-B.

B. Path demonstrations

Demonstrations corresponding to all of the generated path models in the library consisted of one set of path points. Demonstration data of the five simple geometric shapes was generated with software, much like the teaching data for these paths. Demonstration data of the paths on the test object was collected with human demonstration using the same tracking tool the teaching data was collected with. The test object's pose was different during the collection of path teaching data and during the tracking of the path demonstrations to assess the performance of the path canonicalization. No noise was added to the demonstration data of the paths.

C. Path recognition

The path recognition algorithm (Algorithm 3) was executed with the demonstration data and the taught path models. Fig. 5 presents one of the paths demonstrated on the test object in its canonicalized form (plotted in blue), alongside the Gaussian path model from the library it was classified as (plotted in black). The gray ellipsoid shapes represent the covariances of the Gaussian path model's keypoints.

Fig. 6 displays a comparative analysis of the path recognition algorithm's performance with the selected paths. Each

Algorithm 5 Correct a path model's keypoints

> **Input:** Correction path points, Gaussian path model
> **Output:** Adjusted 3-D path model

1: Correction keypoints ← RDP(Correction path points)
2: Find the closest path model keypoint to the first correction keypoint
3: **if** closest point is the first point of path model **then**
4:     Next keypoint → *first redundant keypoint*
5: **else if** closest point is the last point of path model **then**
6:     Previous keypoint → *first redundant keypoint*
7: **else**
8:     Draw line segment between the closest point and the following path model keypoint
9:     Project first correction keypoint onto line segment
10:     **if** projection is between closest points **then**
11:         Next keypoint → *first redundant keypoint*
12:     **else**
13:         Closest keypoint → *first redundant keypoint*
14:     **end if**
15: **end if**
16: Find the closest path model keypoint to the last correction keypoint
17: **if** closest point is the first point of path model **then**
18:     Next keypoint → *last redundant keypoint*
19: **else if** closest point is the last point of path model **then**
20:     Previous keypoint → *last redundant keypoint*
21: **else**
22:     Draw line segment between the closest point and the following path model keypoint
23:     Project first correction keypoint onto line segment
24:     **if** projection is between closest points **then**
25:         Closest keypoint → *last redundant keypoint*
26:     **else**
27:         Previous keypoint → *last redundant keypoint*
28:     **end if**
29: **end if**
30: *Redundant keypoints* ← range(*first redundant keypoint*:*last redundant keypoint*)
31: Replace the redundant keypoints with the correction keypoints in path model's keypoints
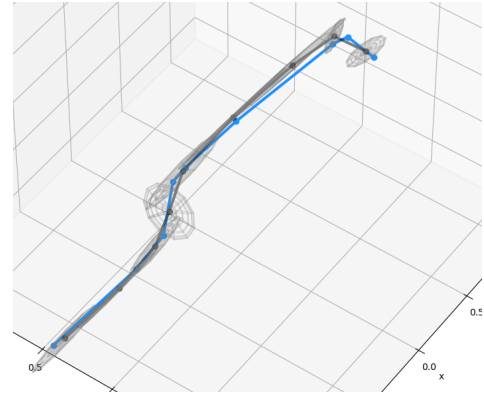32: **return** corrected keypoints

Fig. 5: A demonstrated path (plotted in blue) recognized as a Gaussian path model from the path recognition library (plotted in black).

corrected with the path correction algorithm (Algorithm 5). The path model keypoints, the path correction keypoints and the corrected path are displayed in Fig. 2 for test path 1, Fig. 8 for test path 2, and in Fig. 9 for test path 3.

V. DISCUSSION

A. Role of the quality and quantity of teaching data

The method for path recognition requires only a small amount teaching data — as few as four demonstrations were sufficient for generating a Gaussian path model that was utilizable in path classification. However, for successful path recognition with the generated models, noise was added to the teaching data due to the small amount of teaching data. More testing is needed, with larger teaching sets per path, in order to see if more teaching data can substitute the need for adding noise to the data. Additionally, a process for investigating and deriving a sufficient level of noise in the teaching data to create a robust Gaussian path model is a matter for further research. For the current tests, a suitable amplitude for the added noise for generating the Gaussian model was found by iteration. However, a systematic method for defining the suitable level of noise in teaching data is necessary for utilizing these methods outside testing and development purposes.
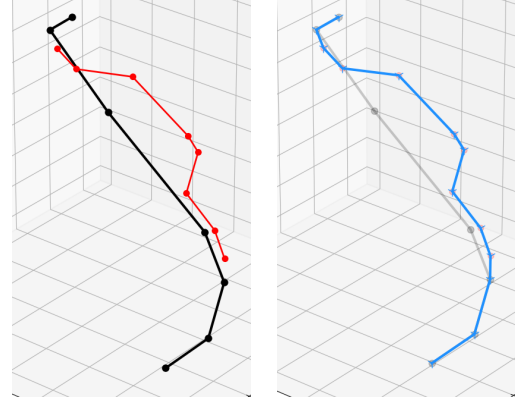
B. Path model decimation

The level of decimation for the path models was found through iteration in the tests by executing the path recognition algorithm while varying the tolerance value of the decimation algorithms and tracking the scores of the correct match and the best-scoring incorrect match (IV-A). The difference of these scores was maximized. Fig. 10 displays examples of the path recognition score's behaviour as the tolerance value of the RDP decimation algorithm is varied. As the tolerance value is increased, the amount of keypoints in the model is reduced, increasing the score for incorrect recognitions. The selected tolerance value for the decimation is highlighted with a dashed line, where the difference between the correct recognition score and the best score for an incorrect recognition is at its

---

column represents a demonstrated path and each row a path model in the path recognition library. A green-yellow-red color coding signifies the score values for each path classification from good to worse. The best score for each demonstrated path is highlighted in bold. The robustness of the presented path recognition algorithm is emphasized by Fig. 6 as no false classifications were made.

D. Path correction

Three paths (test paths 1-3 in Fig. 3) were demonstrated on the test object (Fig. 7) along with corrections to the paths. These paths were classified with the path recognition algorithm (Algorithm 3) and the keypoints of the best-fit model were

		Demonstrated models									
		parabole	075_rectangle	025_circle	spiral	050_circle	test_path1	test_path2	test_path3	test_path4	test_path5
Library models	parabole	**135**	-8515	-1448	-12369	-4157	-294	-4398	-3708	-5837	-4968
	075_rectangle	-9651	**79**	-7945	-69142	-29820	-15668	-19168	-26797	-27902	-21856
	025_circle_rev	-717	-15699	**92**	-2321	-406	-8642	-422	-237	-6312	-223
	spiral	-30486	-16917	-25103	**54**	-16659	-55482	-11303	-5101	-49751	-14602
	050_circle	-76990	-27537	-42393	-7361	**36**	-30044	-325	-1932	-34872	-669
	test_path1_flip1	-205	-11980	-1550	-38523	-10422	**12**	-8212	-9233	-23383	-10138
	test_path2_flip1	-4279	-9526	-3166	-116846	-4815	-9288	**71**	-21902	-61440	-2398
	test_path3_flip1	-2312	-7250	-2736	-15992	-4391	-16759	-11284	**117**	-15357	-12461
	test_path4	-3282	-14777	-3233	-60354	-17721	-1783	-17335	-25189	**22**	-17102
	test_path5_flip1	-1908	-2292	-1569	-1860	-433	-1024	-13	-263	-1444	**14**

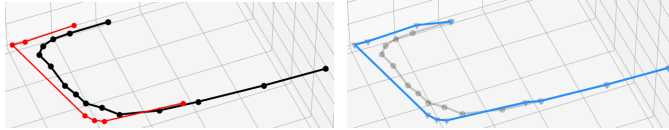Fig. 6: The path recognition algorithm's performance with the selected paths.

Fig. 7: Approximate positions of test path 1 (red), test path 2 (green), and test path 3 (blue) and their respective corrections (dotted lines) on the test object.

(a) Path model keypoints (black) and correction keypoints (red).

(b) Corrected path's keypoints.

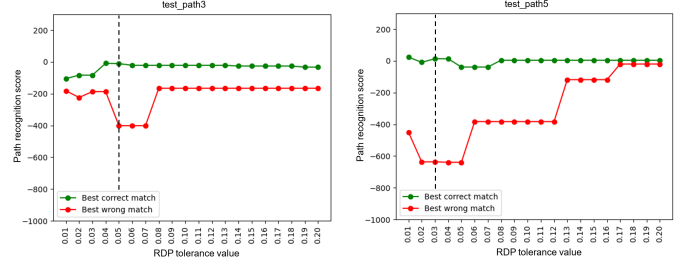Fig. 9: An example of path correction for test path 3.

(a) Path model keypoints (black) and correction keypoints (red).

(b) Corrected path's keypoints.

Fig. 8: An example of path correction for test path 2.

(a) Test path 3.

(b) Test path 5.

Fig. 10: The path recognition scores of the correct recognition (green) and the best incorrect recognition (red) as a function of RDP algorithm's tolerance value. The selected RDP tolerance is highlighted with a dashed line.

largest. It is important to note that this method for finding a suitable level of decimation requires manual work along with the knowledge of a correct match from the path recognition library to the path being demonstrated. Systematic methods for finding the suitable level and decimation for teaching data could be developed. Research on methods that utilized parameters derived from the path's shape (such as path length, angular changes and rate of angular changes) to deduce a tolerance parameter for the decimation algorithms is a possible extension to the work introduced in this paper.

C. Path modification

The presented path modification algorithm has a limitation in that the demonstrated correction has to be in the same coordinate frame as the demonstrated path. However, this method allows for utilizing the path library's models in a flexible manner: A path can be demonstrated by hand, and corrected subsequently through providing a new path without the need for teaching the desired path shape. This can be useful for long or complex paths that require small or one-time modifications. Furthermore, in situations where path shape

changes (such as grinding, where the object geometry changes with each execution of a path) occur over time, this method may be used to compensate for the geometry changes.

D. Future Work

More testing of the path recognition with different paths is needed in order to determine the scope of the required variance in path shapes for generating a path library capable of classifying path demonstrations in most use cases. More investigation into the use of synthetically generated path models for classifying human demonstrations is underway. Preliminary testing on this has already been conducted with the proposed methods. In addition, force values will be introduced in the Gaussian path models to allow the path library to be used for force controlled motion programming by demonstration. Moreover, extending path models with time data and orientations for the path keypoints will be investigated as well. Finally, path segmentation – using path models in such a library as path primitives – will be investigated in the future.

VI. Conclusions

Methods for creating Gaussian path models and using those models to classify paths by demonstration were presented in this paper. The methods were validated through generating a path model library from synthetically generated as well as demonstrated path data and by using this library in the classification of generated and demonstrated paths. Results of the testing showed that the presented methods work as intended for both software generated paths and human demonstrations of paths. Additionally, methods for utilizing the Gaussian path models for robot motion programming and modifying existing Gaussian path models were introduced. The objective of this work is that software generated path data (such as path data from 3-D CAD models) and human demonstrations of paths may be used for programming precise robotic motions intuitively with human demonstrations. The results presented are promising and further studies will be conducted.

Acknowledgment

References

[1] Ravichandar, H., Polydoros, A. S., Chernova, S., and Billard, A. (2020). Recent advances in robot learning from demonstration. Annual review of control, robotics, and autonomous systems, 3(1), 297-330.

[2] J. Aleotti and S. Caselli, "Robust trajectory learning and approximation for robot programming by demonstration," Robotics and Autonomous Systems, vol. 54, no. 5. Elsevier BV, pp. 409–413, May 2006. doi: 10.1016/j.robot.2006.01.003.

[3] Calinon, Sylvain. "A tutorial on task-parameterized movement learning and retrieval." Intelligent service robotics 9 (2016): 1-29.

[4] K. Sugiura, N. Iwahashi, H. Kashioka, and S. Nakamura, "Learning, Generation and Recognition of Motions by Reference-Point-Dependent Probabilistic Models," Advanced Robotics, vol. 25, no. 6–7. Informa UK Limited, pp. 825–848, Jan. 2011. doi: 10.1163/016918611x563328.

[5] J. Sun et al., "Fruit flexible collecting trajectory planning based on manual skill imitation for grape harvesting robot," Computers and Electronics in Agriculture, vol. 225. Elsevier BV, p. 109332, Oct. 2024. doi: 10.1016/j.compag.2024.109332.

[6] Z. Chen and K. Fan, "An online trajectory guidance framework via imitation learning and interactive feedback in robot-assisted surgery," Neural Networks, vol. 185. Elsevier BV, p. 107197, May 2025. doi: 10.1016/j.neunet.2025.107197. and Electronics Engineers (IEEE), pp. 175–185, Feb. 2023. doi: 10.1109/tmech.2022.3196036.

[7] V. Kruger, V. Tikhanoff, L. Natale, and G. Sandini, "Imitation learning of non-linear point-to-point robot motions using dirichlet processes," 2012 IEEE International Conference on Robotics and Automation. IEEE, pp. 2029–2034, May 2012. doi: 10.1109/icra.2012.6224674.

[8] F. Meier, E. Theodorou, F. Stulp, and S. Schaal, "Movement segmentation using a primitive library," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 3407–3412, Sep. 2011. doi: 10.1109/iros.2011.6094676.

[9] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters, "Probabilistic segmentation applied to an assembly task," 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids). IEEE, pp. 533–540, Nov. 2015. doi: 10.1109/humanoids.2015.7363584.

[10] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters, "Learning movement primitive libraries through probabilistic segmentation," The International Journal of Robotics Research, vol. 36, no. 8. SAGE Publications, pp. 879–894, Jul. 2017. doi: 10.1177/0278364917713116.

[11] T. Eiband, J. Liebl, C. Willibald, and D. Lee, "Online task segmentation by merging symbolic and data-driven skill recognition during kinesthetic teaching," Robotics and Autonomous Systems, vol. 162. Elsevier BV, p. 104367, Apr. 2023. doi: 10.1016/j.robot.2023.104367.

[12] T. Eiband and D. Lee, "Identification of Common Force-based Robot Skills from the Human and Robot Perspective," 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids). IEEE, pp. 507–513, Jul. 19, 2021. doi: 10.1109/humanoids47582.2021.9555681.

[13] C. Song, G. Liu, X. Zhang, X. Zang, C. Xu, and J. Zhao, "Robot complex motion learning based on unsupervised trajectory segmentation and movement primitives," ISA Transactions, vol. 97. Elsevier BV, pp. 325–335, Feb. 2020. doi: 10.1016/j.isatra.2019.08.007.

[14] Ginesi, Michele, Nicola Sansonetto, and Paolo Fiorini. "Dmp++: Overcoming some drawbacks of dynamic movement primitives." arXiv preprint arXiv:1908.10608 (2019).

[15] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," Computer Graphics and Image Processing, vol. 1, no. 3. Elsevier BV, pp. 244–256, Nov. 1972. doi: 10.1016/s0146-664x(72)80017-0.

[16] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," Cartographica, vol. 10, no. 2. University of Toronto Press Inc. (UTPress), pp. 112–122, Dec. 01, 1973. doi: 10.3138/fm57-6770-u75u-7727.

[17] M. Visvalingam and J. D. Whyatt, "Line generalisation by repeated elimination of points," The Cartographic Journal, vol. 30, no. 1. Maney Publishing, pp. 46–51, Jun. 1993. doi: 10.1179/caj.1993.30.1.46.

[18] Northern Digital Inc, "Polaris Vega XT," ndigital.com, https://www.ndigital.com/optical-navigation-technology/polaris-vega-xt/ (Accessed Mar. 13, 2025).