FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Automated change detection on spatiotemporal events

Pedro Emanuel Sousa Pinto

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Alexandre Miguel Barbosa Valle de Carvalho

Second Supervisor: Prof. Marco Oliveira

July 25, 2023

Automated change detection on spatiotemporal events

Pedro Emanuel Sousa Pinto

Mestrado em Engenharia Informática e Computação

July 25, 2023

Resumo

Atualmente, graças à evolução da tecnologia, há uma grande quantidade de dados espácio-temporais disponível sobre o movimento de objetos como carros ou objetos que mudam de forma ao longo do tempo como as fronteiras de fogos florestais ou de um tumor, mas o desenvolvimento de software capaz de extrair automaticamente informação sobre eventos espácio-temporais não seguiu esta evolução ao mesmo ritmo. Para fazer progredir o estado da arte nesta área, foi proposta uma framework conceptual e desenvolvida no contexto de uma dissertação de mestrado. Na base desta framework encontra-se um algoritmo de Point Set Registration (CPD) usado para detetar mudança, mas testes realizados no passado mostraram que a escolha do algoritmo para esta tarefa tem um grande impacto na qualidade dos resultados desta framework.

Tendo em conta que não foi feito um trabalho exaustivo de comparação antes da seleção do algoritmo de PSR utilizado, esta família de algoritmos foi agora estudada em detalhe, incluindo novas técnicas baseadas em Deep-Learning, de forma a escolher e testar as técnicas que demonstram um maior potencial para melhorar os resultados da framework. Seguindo a metodologia de testes definida para a framework, os diferentes algoritmos foram comparados entre si e com o algoritmo já implementado na framework. Implementado na framework, cada algoritmo foi avaliado numa série de testes de forma a concluir qual é a melhor técnica a ser usada neste contexto, ou se, de forma a maximizar os resultados da framework, é necessário usar diferentes técnicas em tarefas com diferentes características. Para isto, foi desenhado e executado um protocolo de testes com base em testes e métricas já presentes na framework.

Os resultados mostraram que 2 dos algoritmos (GMMREG e KCREG) apresentam vantagens em relação ao CPD em alguns casos específicos, apesar de no geral atingirem resultados ligeiramente inferiores. O algoritmo baseado em Deep Learning escolhido ficou aquém das espectativas.

Keywords: Point Set Registration

Abstract

Nowadays, thanks to the evolution of technology, there is a huge amount of spatiotemporal data available about moving objects like cars or shape shifting objects like forest fire boundaries or tumor boundaries, but the development of software capable of automatically extracting information about spatiotemporal events did not follow this evolution at the same pace. In order to progress the state-of-the-art in this area, a conceptual framework was proposed and developed in the context of a masters dissertation. At the core of this framework a Point Set Registration algorithm (CPD) is used to detect change, but past performed tests showed that the chosen algorithm for this task greatly impacts the overall performance and quality of results provided by the framework.

Given the fact that no exhaustive comparative work was performed to select the adopted PSR algorithm, the Point Set Registration family of algorithms will be studied in detail, including some of the new Deep Learning based techniques, in order to test the algorithms and choose the techniques that have the greatest potential towards result quality and performance. By following the testing methodology defined for the framework, the different algorithms will be compared and also with respect to the algorithm already used by in the framework. Connected to the framework, each alternative technique will run a set of tests in order to conclude which algorithm is the best overall to be used in this context, or if in order to maximize the framework's performance. To do so, a testing protocol was design and executed, based on tests and metrics already present in the framework.

Results show that 2 of the algorithms (GMMREG and KCREG) present advantages in particular use cases when compared to the CPD algorithm, although their results are in general inferior. The Deep Lerning based algorithm under performed for the expectations.

Keywords: Point Set Registration

Acknowledgements

To my parents, for always believing in me, for supporting me in everything I do and for doing their best to guide me through life. Without you, all my accomplishments in life would have not been possible.

To Tomeci, for being the main reason I finished my masters in 5 years, the carrier of my piano heavy dreams but, most importantly, for being an amazing friend that I will always keep by my side.

To Grande Pedro, for always being next to me during this 5 year adventure. We worked together, we suffered together, we have been pushed to our psychological limit together. But we have also laughed and celebrated together. And in the end, we both made it, and we will stay together.

To JP, for all the great memories and being the best bad influence I could have asked for. I cannot think of a better person to play that role in my life.

To Xènia and Sana, for being present both in my highs and lows, for giving me the strength to improve myself and move forward in life, and for being the friends I will always have, far from my sight but close to my heart.

To Pedro, Pacheco and Afonso for never allowing me to have a dry mouth for the last 3 years, and for taking me on many adventures that I will forever remember.

To Edgar, for the company throughout the endless hours I spent working on this project.

To Rita and Inês, for making the effort to remain my friends through the years and being an amazing company in our many (future) travels around the world.

To Professor Alexandre Valle de Carvalho, my supervisor, for guiding me through the whole journey that was this work.

Finally I would like to thank and acknowledge that this thesis was also performed in the context of a sccolarship that was financed by the Portuguese Foundation for Science and Technology I.P. (PIDDAC) and by the European Regional Development Fund (ERDF), through Operational Agenda for Competitiveness Factors Programme (POFC), project Spatiotemporal Data Models and Algorithms for Earth and Environmental Sciences, reference MIT-EXPL-ACC-0057-2021.

Pedro Pinto

“There are some defeats more triumphant than victories”

Michel de Montaigne

Contents

1 Introduction **1**
1.1 Context and Motivation 1
1.2 Problem 2
1.3 Hypothesis 2
1.4 Objectives 2
1.5 Methodology 3
1.6 Structure 3

2 Point Set Registration **4**
2.1 Definition 4
2.2 PSR General Pipeline 5
2.3 Classification of PSR Techniques 6
2.4 Experiments, Datasets and Metrics 10
2.5 Survey of PSR Techniques 13
2.5.1 Schrodinger Distance Transform Method 13
2.5.2 Gravitational Approach for PSR 15
2.5.3 NRGA: Gravitational approach for non-rigid point set registration 16
2.5.4 Robust ICP Registration using Biunique Correspondence 16
2.5.5 Globally Optimal - Iterative Closest Point 17
2.5.6 Non-rigid Iterative Closest Point 18
2.5.7 Iterative Closest Point - Maximum Correntropy Criterion 19
2.5.8 Extended Coherent Point Drift 20
2.5.9 Bayesian Coherent Point Drift ++ 21
2.5.10 Anisotropic Generalized Bayesian Coherent Point Drift 22
2.5.11 FlowNet3D 22
2.5.12 HPLFlowNet 24
2.5.13 Point Net Encoding for Point Cloud Alignment 25
2.5.14 Non-Rigid Point Set Registration Network 26
2.5.15 PointNetLK 27
2.5.16 Groupwise Registration using T-Mixture Model 29
2.6 Summary 30

3 Solution **31**
3.1 Problem 31
3.2 Techniques to be tested 33
3.3 Implementation details 34
3.3.1 General Procedure 35
3.4 Summary 38

4 Testing Protocol **39**
4.1 Testing pipeline . . . 39
4.2 Metrics . . . 39
4.2.1 Metric M1 . . . 40
4.2.2 Metric M2 . . . 42
4.3 Test cases . . . 44
4.4 Procedure . . . 46
4.5 Summary . . . 47

5 Results and Analysis **49**
5.1 Experiments and Results . . . 49
5.2 PR-NET analysis . . . 50
5.3 Threshold Adjustments . . . 52
5.4 Different Number of Vertices . . . 53
5.5 Summary . . . 55

6 Conclusions **57**
6.1 Conclusions . . . 57
6.2 Future Work . . . 60

References **61**

List of Figures

2.1 Visual representation of simple PSR problem, extracted from [31]. 4
2.2 Threats to the accuracy of PSR algorithms, extracted from [51]. 5
2.3 Pipeline of the ICP algorithm, extracted from [28]. 6
2.4 Formulating PSR as a maximum likelihood problem, extracted from [27]. 7
2.5 Example of PSR using a gravitational approach to align two surfaces, from [1]. . 7
2.6 Examples of rigid and non-rigid transformations. 8
2.7 Some examples of the ModelNet dataset. 10
2.8 Example of the FlyingThings3D dataset. 10
2.9 Example of the Stanford 3D scanning repository dataset. 11
2.10 Some examples of the CE-Shape-1 dataset. 11
2.11 Some examples of the BU-3DFE dataset. 11
2.12 Example of the KITTI dataset. 11
2.13 Formula to calculate RMSE. 12
2.14 Visual representation of the first metric created by [6]. 13
2.15 Example of two sequences of transformations that the second metric created by [6] compares. 13
2.16 Visualization of the results of the second test performed by the authors of GAPSR, extracted from [14]. 15
2.17 Visual representation of the results obtained by the techniques tested, extracted from [1]. 16
2.18 Difference in the correspondence calculation step between (a) ICP and (b) BC-ICP, extracted from [49]. 17
2.19 Iterative steps taken by Go-ICP in a PSR task, extracted from [45]. 18
2.20 Example of a registration test performed by NICPs, extracted from [24]. 18
2.21 Visual representation of (a) the mean square distance and (b) correntropy, extracted from [7]. 19
2.22 Results of ECPD and CPD in the registration of 3D scans of an arm, extracted from [14]. 20
2.23 Approach followed by BCPD++ to accelerate registration, extracted from [18]. . 21
2.24 Results of the second experiment performed in [48]. 22
2.25 FlowNet3D architecture, extracted from [25]. 23
2.26 FlowNet3D scene flow predictions for four KITTI scans, extracted from [25]. . . 24
2.27 Visual representation of the architecture of HPLFlowNet, extracted from [16]. . . 24
2.28 Visual representation of Iterative Alignment performed by i-PCRNet, extracted from[37]. 26
2.29 Visual representation of the architecture of PR-Net, extracted from [41]. 27
2.30 Visual representation of the architecture of PointNetLK, extracted from [3]. . . . 28

2.31 Example results of test cases performed to compare PointNetLK with ICP, extracted from [3]. 28
2.32 Comparison between Student's t distribution and Gaussian distribution in the presence of noise, extracted from [34]. 29
2.33 Results of TMM (third column) and mrTMM (fourth column) in two group registration problems, extracted from [34]. 30

3.1 Visual representation of the framework's pipeline for extracting information about spatiotemporal events. 31
3.2 Visual representation of the framework's architecture, extracted from [6]. 34
3.3 Visual representation of the vector between the centroids of two point sets. 36
3.4 Visual representation of the calculation of the angle between the vectors that represent a pair of corresponding sides of two polygons. 36
3.5 Polygon represented by the point set used to create training and testing datasets. . . 37
3.6 Comparison between different levels of deformation. 38

4.1 Example of input sent by the core service and respective output given by the algorithm's servicem extracted from [6]. 40
4.2 Representation of **A** Δ **B** (in orange). 40
4.3 Convex hull of a polygon (in blue). 41
4.4 Visual representation to help understand the calculation of metric M1. 41
4.5 Polygon shapes not intersecting. 41
4.6 Visual representation of how metric M1 penalizes worse estimates. 42
4.7 Example of the calculation of metric M2. 43
4.8 Visual representation of the sequence of tranformations happening in test 5. . . . 45
4.9 Visual representation of the sequence of tranformations happening in test 7. . . . 46
4.10 Visual representation of the sequence of transformations happening in test 11. . . 47
4.11 Diagram about the procedure for each experiment. 47

5.1 Example of alignment performed by PR-NET. 50
5.2 Demonstration of lack of ability of the PR-NET algorithm to detect immutability. 51
5.3 Point removed from the point set. 53
5.4 Points removed from the point set. 54

List of Tables

2.1 Summary of surveyed techniques. 14

5.1 Results of tests 1 to 6. 49
5.2 Results of tests 7 to 11. 50
5.3 Results of immutability detection by PR-NET (measured by the RMSD). 51
5.4 Results of tests 1 to 6 before and after threshold adjustments. 52
5.5 Results of tests 7 to 11 before and after threshold adjustments. 52
5.6 Results of tests 1 to 6 with the removal of one vertice. 54
5.7 Results of tests 7 to 11 with the removal of one vertice. 54
5.8 Results of tests 1 to 5 with the removal of some vertices. 55
5.9 Results of tests 6 to 11 with the removal of some vertices. 55

Chapter 1

Introduction

Spatiotemporal events regarding a given object consist in a change of state like appearance/ disappearance of the object, change of shape, movement/orientation and other non-physical properties that the object can suffer and happen at a particular time (or time interval) and location [47]. Examples of spatiotemporal events are the changes of trajectory, speed and acceleration in a car´s movement or the changes of shape and location as an iceberg drifts in the ocean and melts over time, a forest fire boundary lines or the boundaries of a tumor. With the advance of technology there are currently a considerable number of devices that keep track and record spatiotemporal data [42]. However, the huge amount of captured spatiotemporal data can only be useful if there are efficient processes that automatically extract knowledge regarding features, behaviors, trends and patterns that can be used for a designated purpose.

1.1 Context and Motivation

On the representation and quantification of change in spatiotemporal phenomena [6] it was concluded that there is still a lack of tools that extract spatiotemporal features from raw spatiotemporal data according to user needs. The authors proposed a conceptual framework with the goal of automatically detecting spatiotemporal features based on raw spatiotemporal data and on user-defined goals.

The conceptual framework focuses on the detection and quantification of spatiotemporal phenomena based on user input information such as which types of events should be detected, as well as what scale actually makes them worthy of attention. Different users can define distinct event types and scales leading to distinct outputs for the same input spatiotemporal dataset.

The raw spatiotemporal input data consists of a sequence of temporal snapshots of a given object. Each snapshot contains a 2D shape descriptor that defines the contours of the given object. The output of the framework is the set of spatiotemporal events that match input event types considered relevant, as well as their quantification and the time intervals for which these events occurred, in the form of object change representations (representation of a change specifying the time interval of its occurrence).

The conceptual framework towards automatic detection and quantification of spatiotemporal phenomena requires at its core a Point Set Registration algorithm (PSR) and it was a conclusion of the authors that results may heavily depend on the selected PSR algorithm.

Being a work limited in time, the authors could not perform tests using several PSR techniques with the conceptual framework prototype and measure how distinct techniques affect the quality of results. Hence, the present work focuses on answering the previous questions concerning the detection and quantification of change in spatiotemporal phenomena about how results vary according of the steps identified to improve the framework in order to contribute to the advancement of the state-of-the-art in this area and help further improving a tool that is already very valuable.

1.2 Problem

In order to detect and quantify the relevant spatiotemporal phenomena, this framework has implemented in its core a Point Set Registration algorithm (more about this in Chapter 2) that compares the shape descriptors of two consecutive timestamps and identifies the transformations that happened between them. During the development of the framework, a PSR algorithm was chosen and implemented and later tests concluded that the accuracy of the framework as a whole (and, consequently, its usefulness) greatly depends on the ability of the PSR algorithm to correctly detect the transformations happening between timestamps. But, due to lack of time, no other algorithm was implemented and tested which left open the question of whether the algorithm being used is indeed the one that can give the framework the best possible results.

Therefore, this dissertation further researches within the described scope by studying the application and effect of Point Set Registration algorithms in more detail, including Deep Learning-based techniques.

1.3 Hypothesis

Based on the described problem, this work studies different PSR techniques in the central role of detection and quantification of spatiotemporal data objects that evolve over time. Furthermore, this work assesses traditional PSR techniques as well as deep-learning PSR techniques.

1.4 Objectives

To answer the questions identified in the previous section this work focuses on the study of alternative PSR techniques at the core of spatiotemporal feature detection and quantification. Therefore the research objectives are:

- identify and study existing PSR techniques;
- draw conclusions about what PSR algorithm presents the best results for this context or how to use more than one technique in order to maximize the accuracy of the framework.

1.5 Methodology

The task at hand is divided into the following steps:

1. extensively study the state-of-the-art of PSR algorithms, concerning both the traditional techniques and the more recent deep learning based techniques;
2. after deciding which are the most promising PSR algorithms, implement or adapt implementations to be used in the framework;
3. design an experimentation protocol: tests, procedures, variables, test execution;
4. perform tests and collect results;
5. analyze results and reach conclusions in light of the written hypothesis.

1.6 Structure

This dissertation is structured as follows. Chapter 2 presents the literature review about PSR algorithms and how to test them. Chapter 3 describes the proposed solution, mentioning some implementation details.

Chapter 2

Point Set Registration

This work focuses on Point Set Registration (PSR) techniques. To that end, it is necessary to study the state-of-the-art literature to identify the main concepts about the field, the characteristics of the existing and more promising techniques, as well as how these techniques are evaluated and compared. Therefore, section 2.1 of this chapter exposes the main concepts about PSR. Section 2.2 exposes the general pipeline of PSR techniques using a particular example. After, section 2.3 describes the different ways PSR techniques can be categorized as well as their usual pipeline. Section 2.4 analyzes the experiments and metrics usually used to test and compare PSR techniques. Section 2.5 presents the PSR techniques surveyed during this study and tries to expose their main characteristics, as well as classifying them according to the concepts described in section 2.3 and commenting on their results recorded by their own authors. Finally, section 2.6 ends the chapter with its summary.

2.1 Definition

Point Set Registration (or point matching) is the process of calculating the transformation that aligns two point sets with each other [28]. These point sets simply represent a set of points in a coordinate system. PSR applications are usually related with computer vision tasks like object recognition through shape matching [4], face recognition [43] or medical image processing [39]. Figure 2.1, extracted from [31], gives a visual representation of a simple PSR problem.

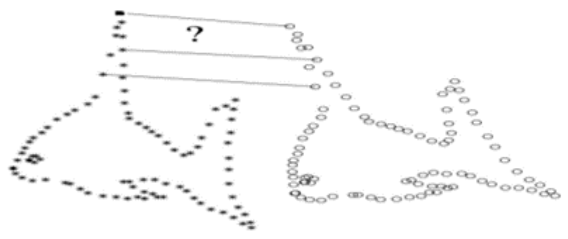


Figure 2.1: Visual representation of simple PSR problem, extracted from [31].

The biggest threats to the accuracy of these techniques, presented in Figure 2.2, are deformations and noise that can be originated by, for example, different viewpoints, poses, partial occlusion or outliers.

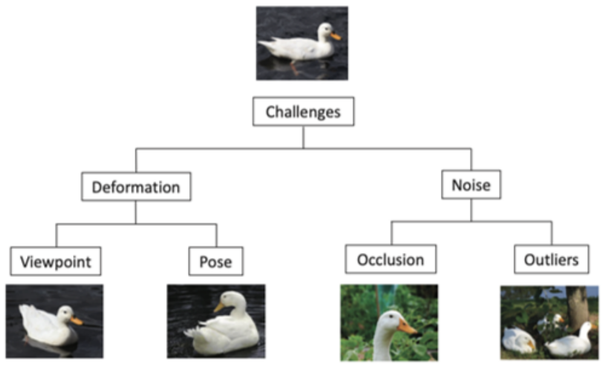


Figure 2.2: Threats to the accuracy of PSR algorithms, extracted from [51].

2.2 PSR General Pipeline

PSR techniques tend to follow a general pipeline which is, in most part, common to all of them. This pipeline is composed of six steps [50]: point selection, point matching, weighting pairs, outlier rejection, error minimization and motion estimation. Maiseli *et al.* [28] explain each one of these steps in the perspective of one of the most well-known PSR algorithms, ICP [36]:

1. Point selection: aims to discard useless data from either point set, and the goal is to boost the computational speed of the algorithm;
2. Point matching: creates correspondences between point sets;
3. Weighting pairs: assigns appropriate weights to matched pairs of the point sets (closer point, larger weights, and distant points, smaller weights);
4. Outlier rejection: eliminates matched point pairs that are likely to affect the error minimization process;
5. Error minimization: minimizes the error (based on a chosen metric) correspondent to the difference of the two point sets;
6. Motion estimation: estimates the transformation that aligns the two point sets.

Figure 2.3 presents the pipeline for the ICP algorithm and, although slightly different from the pipeline presented above, the similarities are clear and the main steps the same.

All techniques surveyed that were proposed prior to 2018 fit one of the categories mentioned above. But, since 2021, new approaches to solve PSR problems have been proposed which are based on a change in the computational paradigm towards deep learning and Artificial Intelligence

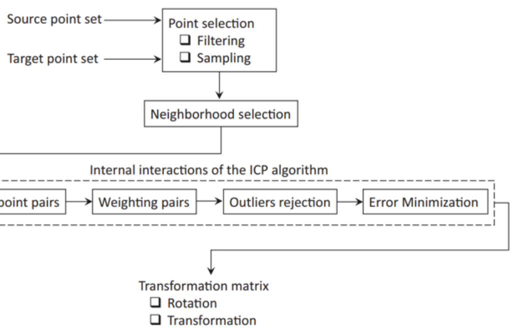


Figure 2.3: Pipeline of the ICP algorithm, extracted from [28].

techniques [5]. To distinguish between these two families, the remainder of this document will refer this last family as Deep Learning Based techniques (addressed in section 2.5) as opposed to the classic computational paradigm referred as Traditional techniques.

2.3 Classification of PSR Techniques

Pairwise/groupwise

PSR techniques are broadly split into two categories by several articles: pairwise and groupwise [51] [17]. Pairwise registration algorithms consider only two point sets and calculate the transformation that better aligns those two sets. According to Zhu *et al.* [51], pairwise registration algorithms can be divided into three categories: distance-based, filter-based and probability-based methods.

- Distance-based methods' major steps are calculating the distance between the two given point sets, finding point correspondences between them and minimizing that difference in the second step based on the correspondences found.
- Filter-based methods make use of a State Space Model (SSM) to solve the registration problem. According to Patterson *et al.* [32], SSM is a time-series model that predicts the future state of a system from its previous states probabilistically. When solving PSR problems with Filter-based techniques like the one proposed by Li *et al.* [23], a state is a set of parameters that represent a transformation. For example, a possible state that represents a transformation that is the result of the combination of a rotation and a translation can follow this structure:

$$x = [x, y, z, tx, ty, tz]$$

- Finally, probability-based models formulate the registration problem as a maximum likelihood estimation problem, mostly using Gaussian Mixture Models. Figure 2.4 helps explain how this concept is applied in a PSR problem, using as example the filter-based PSR technique proposed by Ma *et al.* [27].

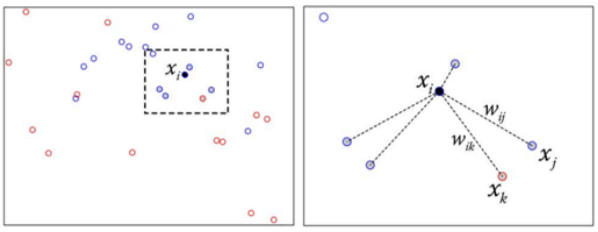


Figure 2.4: Formulating PSR as a maximum likelihood problem, extracted from [27].

In this technique, points from one point set are assigned neighboring points being the closest points belonging to the other point set. Then, weights are assigned to each neighbor based on the likelihood of being the correspondent point and the transformation is finally calculated based on the assigned weights.

While surveying the existent pairwise PSR techniques, we came across a fourth group of techniques, self-entitled physics-based methods [1]. These methods apply physics concepts to solve the registration problem like assigning mass and velocity to the points of the point sets. Figure 2.5 shows a representation of how the technique proposed by Ali *et al.* [1] aligns two surfaces of points. The red surface, which can be described as f(x, y) = cos(x).sin(y) induces a relativistic gravitational force field to attract the blue surface described as f(x, y) = -1.

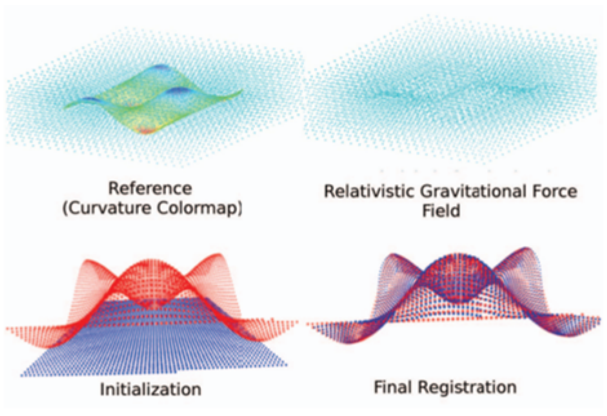


Figure 2.5: Example of PSR using a gravitational approach to align two surfaces, from [1].

Groupwise registration algorithms try to register several point sets simultaneously. Groupwise registration problems can also be solved by using pairwise registration as many times as needed [8] [20] but this solution has some drawbacks like error propagation throughout the several pairwise registration steps [11]. Groupwise registration techniques, according to Zhu *et al.* [51], can be divided into two subgroups: information theoretic-based [17] methods and probability-based methods. In information theoretic-based methods, the registration of multiple point sets is performed based on information theoretic measures, which are a class of measures used for comparing clusters [40]. In probability-based methods, the point sets to be registered are formulated as probability functions. Furthermore, considering the way methods use the mean point set of a cluster, there are two opposite approaches to be taken within groupwise registration techniques: forward and backward [10]. Both approaches involve starting by calculating the mean point set of the cluster. In the forward approach, the different point sets are assumed to be noisy observations of the mean point set, while in the backward approach, the mean point set is assumed to be a noisy observation of all the point sets. Then, it is only necessary to estimate the transformation between the mean point set and each of the point sets in the cluster.

Parametric/non-parametric

PSR techniques can also be described as parametric or non-parametric. According to Romero *et al.* [35], parametric techniques are the ones that try to find a small set of parameters to describe the transformation that minimizes an error function associated with the alignment of two point sets. On non-parametric techniques, the transformation cannot be represented as a set of parameters, implying the existence of deformation vectors for each point.

Rigid/non-rigid transformations

Another difference between PSR techniques is the kind of transformations they estimate. Some techniques estimate only rigid transformations, others estimate only non-rigid transformations and there are some other techniques that can estimate both. Rigid transformations perverse the distance between points, like translations and rotations. Non-rigid transformations are all other transformations that do not perverse the distance between points, with scaling being a very simple example of this type of transformation. Figure 2.6 gives two simple examples of rigid and non-rigid transformations.

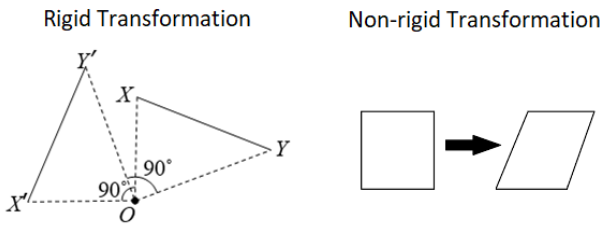


Figure 2.6: Examples of rigid and non-rigid transformations.

Deep Learning Based Techniques

Deep learning based techniques apply the new computational paradigm of deep learning to registration problems. According to Zhang *et al.* [50], Deep Learning Based techniques can be split into two categories, whether they are correspondences-based or correspondences-free.

The key pipeline of most correspondences-free methods involves two steps:

1. identifying the global features (feature descriptors) of the two point sets (this identification must be sensitive to the pose because different poses have different features)
2. calculating the parameters of the transformation corresponding to the difference of the two sets of global features.

The performance of these methods is perfect when the point clouds are the same with different poses but, since the performance greatly depends on the feature extractor network, the major criticism of such correspondence-free approaches is their generalization capability for real life scenarios.

Correspondences-based methods consist of four major modules (similar to the traditional methods): feature extraction, matching, outlier rejection and motion estimation. Most of these deep learning based methods mix traditional modules with deep learning modules. In each one of these modules, deep learning techniques can be applied in different ways:

- In feature extraction, unlike correspondence-free methods, which extract the global features per-point set, the correspondences-based methods extract the per-point or per-patch (subset of a point set) embeddings of the two input point clouds to generate a mapping and estimate a rigid transformation.
- In the matching phase, the common principle for traditional methods is to search the most similar points yi in Y to xi in X to form corresponding pairs. There are deep learning techniques that apply this principle, however, owing to the sparsity and partiality, two point clouds do not always have a point-to-point correspondence. To overcome this, the virtual point method is proposed. This method is used to match points in the matching stage, creating virtual points to correspond to real points.
- For outlier rejection, in traditional methods, RANSAC [12] is the most widely used robust fitting algorithm, which utilizes the maximum consistency as a supervised signal to filter out the outlying matching pairs. Deep learning methods here can be supervised (usually through classification to distinguish inliers from outliers) or unsupervised (usually using a consensus maximization strategy to maximize the number of inliers).
- The last module, motion estimation, can be done either through regression, Singular Value Decomposition (SVD) or weighted SVD that gives a bigger weight to points with more features.

2.4 Experiments, Datasets and Metrics

The authors of PSR techniques tend to perform similar experiments on their techniques and use similar metrics to measure the results of the techniques on those experiments.

Experiments consist in registering several pairs of point sets and calculating the results of the technique based on the chosen metrics. These experiments can be performed in either real datasets (usually datasets with real 3D scans of objects or indoor scenes) or synthetic datasets.

Synthetic datasets allow the authors to have a better control of the registration tasks they expose their techniques to, especially if the authors create the dataset themselves. These datasets also allow the authors to know the ground truth of each test case (the exact transformation the techniques need to identify), which helps to calculate as accurately as possible the metrics that evaluate the performance of the techniques. Real datasets allow authors to better assess how a certain PSR technique can perform in real life scenarios.

There are some datasets which tend to be used more frequently by authors to test their PSR techniques (all the figures from the following listing were extracted from ResearchGate[1]:

- ModelNet [44] is a synthetic dataset which contains numerous 3D CAD (computer-aided designed) models of various different categories like airplanes, plants or lamps;

Figure 2.7: Some examples of the ModelNet dataset.

- FlyingThings3D [29] is a synthetic dataset used particularly for scene flow estimation and consists of everyday objects flying according to randomized 3D trajectories;

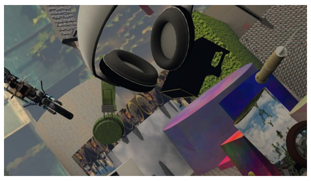


Figure 2.8: Example of the FlyingThings3D dataset.

- The Stanford 3D scanning repository [2] makes countless high-quality real 3D scans of objects available to researchers;

[1]`https://www.researchgate.net/`

[2]`https://graphics.stanford.edu`

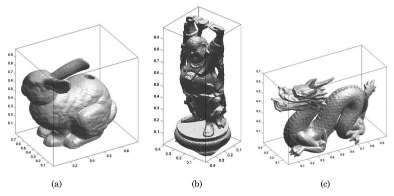


Figure 2.9: Example of the Stanford 3D scanning repository dataset.

- CE-Shape-1 [22] consists of 1400 synthetic shapes group into 70 classes, each containing 20 similar objects like guitars, hats or different animals;

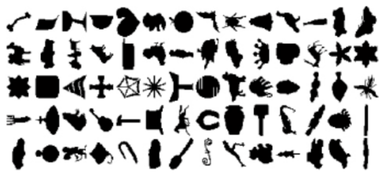


Figure 2.10: Some examples of the CE-Shape-1 dataset.

- BU-3DFE [46] is a 3D real dataset consisting of scans of human faces;

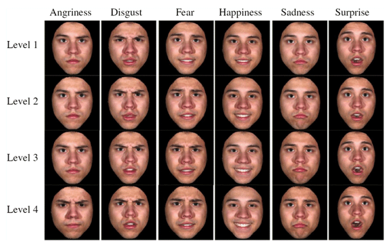


Figure 2.11: Some examples of the BU-3DFE dataset.

- GatorBait100 [3] has 100 2D shapes representing fishes from 30 different classes;
- The KITTI dataset [30] consists of vision tasks built using an autonomous driving platform.

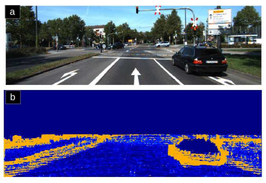


Figure 2.12: Example of the KITTI dataset.

Apart from these datasets, PSR techniques are also tested with different kinds of medical scans like real 3D scans of arms in different poses [2] or real scans of human heads [13].

[3]https://www.cise.ufl.edu/~anand/publications.html

Just like the experiments and the datasets used across different articles tend to be similar, so do the metrics used to evaluate the results of the techniques by different articles.

Probably the most used metric is the root mean squared distance or root mean squared error (RMSD or RMSE) which calculates the difference between the two point sets, meaning that the goal of the techniques is to minimize it. The formula in Figure 2.13 indicates how RMSE is calculated.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Figure 2.13: Formula to calculate RMSE.

There are other metrics like the end point error (EPE or 3D EPE) that can be used with flow estimation accuracy (ACC). While EPE and 3DEPE calculate the difference between two vectors regarding a transformation, ACC calculates the portion of vectors which difference from the ground truth is below a chosen threshold, making it similar to another possible metric which is recall (ratio of true-positive correspondences between points of different point sets after applying the transformation within a pre-specified tolerance).

There are also metrics related to efficiency like the run time or the number of iterations necessary for a technique to converge to a chosen accuracy/error value.

Testing PSR techniques for the detection of spatiotemporal events

Carneiro *et al.*[6], responsible for the conceptual framework this work aims to improve upon, developing their own testing environment which, although using similar experiments and metrics to the ones mentioned above, has its own particularities. They decided to create their own synthetic 2D dataset using the free software Blender. They also created two custom metrics that fit the goal of the framework. The first metric compares the representation of the 2D object obtained when applying the transformation calculated by the framework with the ground truth using the ratio between areas of polygons and convex hulls [15]. Figure 2.14 gives a visual representation of the application of this metric, whose objective is then to evaluate how accurate is the final result given by the framework.

The second metric compares the sequences of transformations identified by the framework with the ground truth sequence of transformations. This sequence of transformations is a list of frame intervals and the transformations that occur during that interval. Figure 2.15 shows an example of the list outputted by the framework and the expected result, which this metric compares.

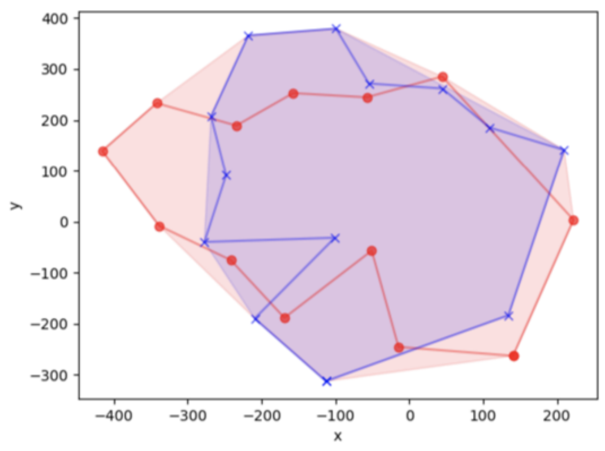
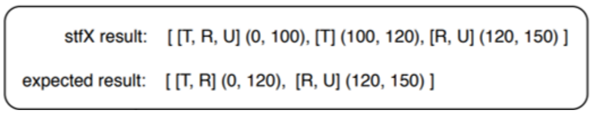


Figure 2.14: Visual representation of the first metric created by [6].



stfX result: [[T, R, U] (0, 100), [T] (100, 120), [R, U] (120, 150)]

expected result: [[T, R] (0, 120), [R, U] (120, 150)]

Figure 2.15: Example of two sequences of transformations that the second metric created by [6] compares.

2.5 Survey of PSR Techniques

Zhu *et al.*[51] performed a survey on PSR techniques where it is presented a table summarizing a considerable number of traditional techniques as well as the classification regarding the criteria presented in section 2.3. During the literature review performed for the current work, we came across several other algorithms which are not present in [51], both traditional and deep learning based. Fitting newly surveyed techniques in the same classifications feels relevant to show that the classification system holds and that it is possible to compare techniques according to similar characteristics.

The following sections address each of the surveyed PSR techniques and present its main characteristics, details and what tests were performed. The following table summarizes the main characteristics of each surveyed technique. Table 2.1 shows that surveyed techniques are mostly beyond 2018, are pairwise and parametric.

2.5.1 Schrodinger Distance Transform Method

SDTM [9] is a **parametric, pairwise, physics-based** PSR technique that supports **rigid as well as non-rigid transformations**. It addresses the problem of point cloud matching as a shape matching problem by transforming each of the given point clouds into a shape representation called the Schrodinger distance transform (SDT) representation. Based on this, point-sets are represented as square-root densities placed at unique locations on a

Table 2.1: Summary of surveyed techniques.

Research Study	**Year**	**Pairwise/ Groupwise**	**Method**	**Rigid/Non-Rigid**	**Parametric/Non-Parametric/**
SDTM	2014	Pairwise	Physics-based	Rigid and Non-rigid	Parametric
Gravitational Approach for PSR	2016	Pairwise	Physics-based	Rigid	Parametric
NRGA: Gravitational approach for non-rigid point set registration	2018	Pairwise	Physics-based	Non-rigid	Parametric
Robust ICP Registration using Bi-unique Correspondence	2011	Pairwise	Distance-based	Rigid	Parametric
Globally Optimal - Iterative Closest Point	2016	Pairwise	Distance-based	Rigid	Parametric
Non-rigid Iterative Closest Point	2018	Pairwise	Distance-based	Non-rigid	Parametric
Iterative Closest Point - Maximum Corretropy Criterion	2019	Pairwise	Distance-based	Non-rigid	Parametric
Extended Coherent Point Drift	2016	Pairwise	Probability-based	Non-rigid	Parametric
BCPD++	2021	Pairwise	Probability-based	Non-rigid	Parametric
Anisotropic Generalized Bayesian Coherent Point Drift	2022	Pairwise	Probability-based	Rigid	Parametric
FlowNet3D	2019	Pairwise	Correspondences-free	Rigid	Non-Parametric
HPLFlowNet	2019	Pairwise	Correspondences-free	Rigid and Non-rigid	Non-parametric
Point Net Encoding for Point Cloud Alignment	2019	Pairwise	Correspondences-free	Rigid	Parametric
PR-Net	2019	Pairwise	Correspondences-based	Non-rigid	Parametric
PointNetLK	2019	Pairwise	Correspondences-based	Rigid	Parametric
TMM	2017	Groupwise	Probability-based	Rigid	Parametric

unit Hilbert sphere: a generalization of the Euclidean space with inner product that does not need to be restricted to a finite number of dimensions.

To gauge the performance of the algorithms, the authors of SDTM use recall ("defined as the ratio of true-positive correspondences calculated in a dataset within a pre-specified tolerance"). SDTM is compared with CPD and GMMReg. The first test was about rigid registration in the form of rotations. For these tests, the algorithms performed registrations on shapes from GatorBait100 dataset and, unlike SDTM and GMMReg, CPD was not able to find the correct correspondence in all test cases. For non-rigid transformation, tested on a 2D fish shape dataset subject to various degrees of deformation, noise, outliers, and occlusion, a fourth algorithm was also tested: RPM-LNS. Results show that SDTM was clearly superior to the other algorithms overall, with RPM-LNS being the second best and CPD arguably the worst. Two more tests were performed on real 3D scans datasets of a toy house and an indoor scene and once again SDTM seems to greatly outperform the other algorithms, with RPM-LNS being the worst of the four algorithms this time.

2.5.2 Gravitational Approach for PSR

The GAPSR technique, proposed by Golyanik *et al.* [14], is a **parametric, pairwise, physics-based, towards rigid transformations** PSR technique. Formulates point set registration as a modified N-body problem with additional constraints. A template point set moves under gravitational forces induced by a reference point set. Every point from both a reference and a template point set is treated as a particle with its own position, velocity and acceleration. Pose updates are completed by numerically solving the differential equations of Newtonian mechanics. Particle movements are expressed by differential equations of Newtonian mechanics.

The authors of GA use mean distance and root-mean-square distance to evaluate the results. GA was compared with ICP and CPD. The first tests done with a synthetically created 3D scan of a bunny shows that GA performs better than ICP but slightly worse than CPD. A second test, which can be visualized with the help of Figure 2.16, was performed consisting of the registration of a real 3D human body scan and in this case, GA achieved very good results, while CPD gets trapped in local minima and ICP fails to get even close to the desired solution.

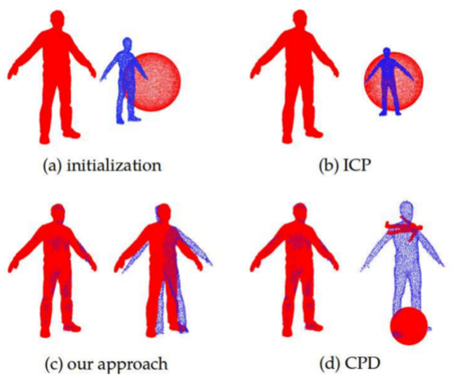


Figure 2.16: Visualization of the results of the second test performed by the authors of GAPSR, extracted from [14].

Two more tests were performed with 3D point sets with the addition of a lot of noise like outliers, and in these two last tests GA proved to be superior to CPD. The authors then conclude that GA out-performs ICP overall and can also out-perform CPD in the presence of very noisy data.

2.5.3 NRGA: Gravitational approach for non-rigid point set registration

NGRA is a **parametric, pairwise, physics-based, towards non-rigid transformations** PSR technique based on simulation of gravitational interactions between points, proposed by Ali *et al.* [1]. It sees the points as particles with masses interacting inside an isolated non-empty space according to the universal law of gravitation. The optimal alignment is referred to as the state of minimum gravitational potential energy between the point sets. It aligns the gravitational fields produced by both point sets.

The error in each experiment of NRGA is given by the root-mean-squared error calculation. Tests were made using the fish and the line (2D models) and the bunny (3D model). For each model, five different cases were created with the addition of different types of noise or perturbations. The NRGA was compared with the NR-ICP, TPS-RPM, CPD and GMMReg. Figure 2.17 shows the results obtained for each of the techniques tested with missing data from one of the point sets.

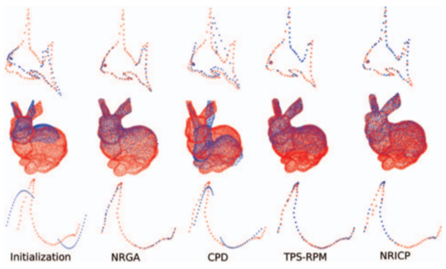


Figure 2.17: Visual representation of the results obtained by the techniques tested, extracted from [1].

For these tests, NRGA seems to be the second best algorithm overall, behind CPD but, one a last test where real 3D scans of human faces were used, NRGA seems to outperform all other algorithms (with CPD being second and NR-ICP being last), which might suggest that the advantages of the NRGA become more clear when registering more complex point sets.

2.5.4 Robust ICP Registration using Biunique Correspondence

The BC-ICP technique is **parametric, pairwise, distance-based and towards rigid transformations**. This technique tries to improve the original ICP [28]. Bi-unique correspondence is introduced to enhance the performance of ICP by searching multiple closest points, and forces every point to be part of one single correspondence. Figure 2.18 shows the difference between ICP and BC-ICP in the step of finding point correspondences.

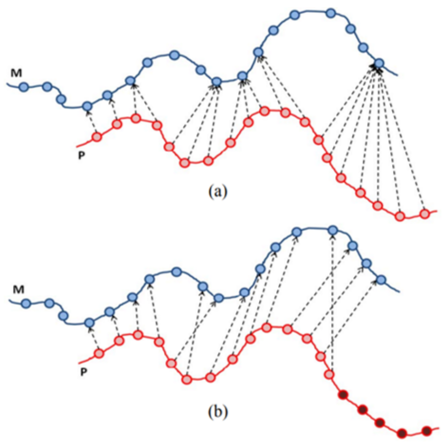


Figure 2.18: Difference in the correspondence calculation step between (a) ICP and (b) BC-ICP, extracted from [49].

It defines a new kind of outlier, called No-Correspondence (NC) Outlier, which is the point that is not assigned to a bi-unique correspondence. Discarding NC Outliers greatly improves the robustness in the case that large rotation error and non-overlapping areas exist.

The BC-ICP was compared against the original ICP and two other variants, PICP and FICP. The root mean square error (RMSE) was used to calculate accuracy. The first test consisted in the registration of two point sets formed by the 3D scan of a statue. BC-ICP was the only technique that could reach the desired solution, while the other three all seem to get trapped in local minima (with PICP being the second best and the original ICP the third best). The second experiment was done using the scan of an indoor environment and the results were very similar. The authors then concluded that the BC-ICP outperforms the original ICP both in accuracy and runtime.

2.5.5 Globally Optimal - Iterative Closest Point

Go-ICP, based on the original ICP [28], is proposed by Yang *et al.* [45] and is a **parametric, pairwise, distance-based and towards rigid transformations** PSR technique. It is also based on a branch-and-bound scheme.

The Branch-and-Bound algorithm goes as follows: before enumerating the candidate solutions of a branch (which, in the context of PSR, is a possible transformation), the branch is checked against upper and lower estimated bounds on the optimal solution, and is discarded if it cannot produce a better solution than the best one found so far by the algorithm. BnB guarantees global optimality regardless of the initialization.

Go-ICP uses iterations of this branch-and-bound scheme intercalated with iterations of ICP to find the transformation that minimizes an error function.

Go-ICP technique was tested in the registration on 3D scans of objects and indoor scenes. Tests used the root mean square error to measure the accuracy of the algorithm. Figure XX shows the iterative steps taken by Go-ICP in one of the experiments.

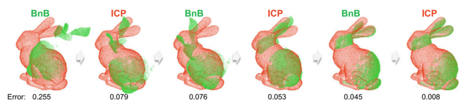


Figure 2.19: Iterative steps taken by Go-ICP in a PSR task, extracted from [45].

The authors conclude that when real time performance is not critical or when a good initialization is not reliably available, the algorithm obtains very good results reaching nearly perfect solutions. However , the authors did not make any direct comparisons with other existent algorithms.

2.5.6 Non-rigid Iterative Closest Point

Proposed by Liang *et al.* [24], and based on the ICP [28] technique, NICPIPCs is a **parametric, pairwise, distance-based and towards non-rigid transformations** PSR technique. In this technique, the distance between point sets (that needs to be minimized) is determined based on multiple two-way correspondences rather than single one-way correspondences.

According to the authors, there are some limitations about this method:

- the computational time is longer than other traditional methods;
- the current version of multiple correspondence search algorithms cannot distinguish similar patterns. For example, the method may deform two horse legs into one or swap them.

The authors of NICPs measure the accuracy of the methods by calculating the distance between the template point set and the target point set. A 3D real dataset of human bone scans is used. Figure 2.20 is an example of one of the test cases.

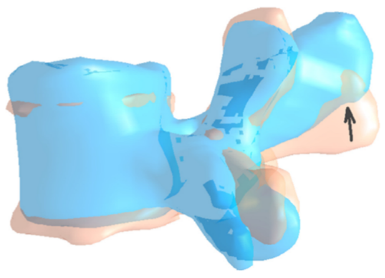


Figure 2.20: Example of a registration test performed by NICPs, extracted from [24].

The algorithm was compared with the original ICP and three other algorithms (Andresen2001, Pauly2005 and Jian2011). The results of the first set of tests (vertebrae scans) show that ICP performs better than the other three algorithms but apparently NICPs is able to achieve even better results than the ICP in all of the 14 tests made. The second set of tests (femur scans) shows that Jian2011 stands out negatively and NICPs continues to be the best algorithm considering the average error of all the 23 tests, but this time is only able to be the best algorithm in 6 of those tests. The authors conclude that even though their method seems to achieve the best results among the algorithms chosen for comparison, it loses to those algorithms in runtime. NICPs' good results also depend a lot on the right selection of some parameters based on the point sets characteristics.

2.5.7 Iterative Closest Point - Maximum Correntropy Criterion

ICP-MCC is a **parametric, pairwise, distance-based and towards non-rigid transformations** PSR technique proposed by Chen *et al.* [7], and is yet another ICP-based technique [28]. It substitutes the traditional least squares measure with a maximum correntropy (another generalized similarity measure) criterion that avoids the influence of outliers, to build a new registration model. Figure 2.21 helps explain why correntropy is less affected by outliers than the mean square distance.

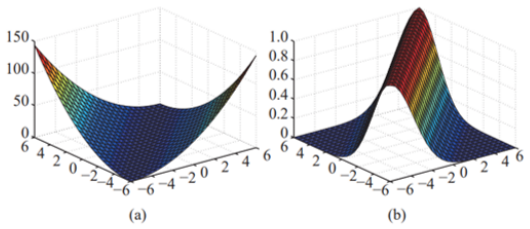


Figure 2.21: Visual representation of (a) the mean square distance and (b) correntropy, extracted from [7].

The mean square distance is not optimal if the error distribution has outliers or is non-symmetric. Because of the inability of the mean square distance in dealing with outliers (the samples that are far away from the mean value of the error distribution have a bad effect), correntropy is introduced to build an alternative objective function (in correntropy, samples far away from the mean value of the error distribution barely have any effect).

At each iteration, the algorithm sets up the index mapping of two point sets according to the known transformation, and then computes the closed-form solution of the new transformation according to the known index mapping.

The authors of ICP-MCC use data from two repositories (CE-Shape-1 and Stanford 3D Scanning Repository) to assess the quality of the results achieved by their algorithm.

For each point set of the dataset, they transform it randomly and store that random transformation as the ground truth. Then, they make their algorithm calculate the transformation and calculate the error as the absolute difference between the result and the ground truth. ICP-MCC is compared with ICP and CPD in these tests. After running some tests both with 2D and 3D point sets, the authors conclude that ICP-MCC is able to achieve better accuracy than ICP with only an almost neglectable increase in runtime, while CPD seems to be the worst of the three algorithms tested.

2.5.8 Extended Coherent Point Drift

ECPD, proposed by Golyanik *et al.* [14], is a **parametric, pairwise, probability-based, towards non-rigid transformations** PSR technique based on the original CPD [31]. It adopts a coarse-to-fine registration strategy which accelerates the registration process linearly by splitting the registration problem into two sub-problems. This is done by, given two point sets X and Y, after calculating the correspondences between them, a Z point set is created by subsampling Y (but only the points that do not have correspondences, the points that have correspondences in Y are guaranteed to be part of the subsample). After registering Z with X, Y is registered with Z to ensure the number of points of the result remains unchanged.

ECPD is compared with CPD, ICP and TPS-RPM on a synthetic 2D “Fish” dataset. In this experience the authors prove that in the presence of structured outliers, ICP and TPS-RPM show poor results, CPD is an improvement compared to those two algorithms but ECPD reveals the best results of all of them. In a second experiment, ECPD is compared with CPD on a real 3D dataset of scans of arms in different poses. The results, visually presented in Figure 2.22, show that ECPD performs better than CPD and its results are less affected by the parameters chosen.

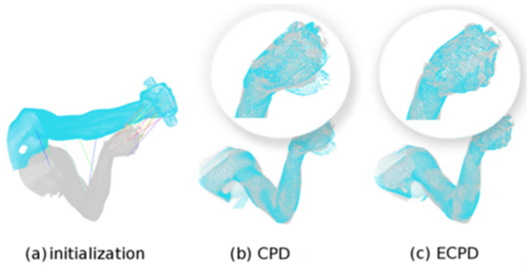


Figure 2.22: Results of ECPD and CPD in the registration of 3D scans of an arm, extracted from [14].

A third experiment was conducted, once again comparing ECPD with CPD on a 3D real dataset with human faces scans and, once again, the authors show the higher accuracy and resistance to outliers presented by the ECPD.

2.5.9 Bayesian Coherent Point Drift ++

Proposed by Hirose *et al.* [18], BCPD++ is a **parametric, pairwise, probability-based, towards non-rigid transformations** PSR technique. Also BCPD-based [19], accelerates non-rigid registration by dividing it into three steps, which Figure 2.23 tries to represent visually:

1. downsampling of point sets;
2. non-rigid registration of downsampled point sets;
3. interpolation of shape deformation vectors corresponding to points removed during downsampling (blank circles in Figure 2.23).

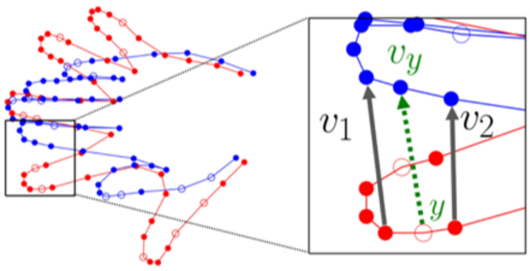


Figure 2.23: Approach followed by BCPD++ to accelerate registration, extracted from [18].

To register downsampled point sets, it uses a registration algorithm based on a prior distribution, called motion coherence prior, which represents the uncertainty of, in this case, a transformation, also enforcing motion coherence between the points.

The experiments were made on a synthetic dataset of 3D scanned shapes (downloaded from the Stanford 3D scanning repository (https://graphics.stanford.edu)). For each shape used, a deformed version of it was created to then make the algorithm solve the registration problem between the original shape and the deformed one. The root mean square error (RMSE) was used to calculate accuracy. The first experiment was done with large data (point sets with a large amount of points) and BCPD++ was compared with BCPD. BCPD achieved slightly better results (0.3 per cent better at best) but BCPD++ was much faster to reach those results, with a runtime up to 10 times shorter than BCPD for point sets with one million points. In the performance evaluation with small data (point sets with at most 1200 points), BCPD++ was compared with BCPD, CPD, GMM-REG and TPS-RPM. Results show that BCPD++ seems to be only the third best algorithm overall in this scenario, with BCPD being first and TPS-RPM a close second, while GMM-REG seems to achieve the worst results. This means that the main advantage of the BCPD++ algorithm is in its time efficiency, which becomes more noticeable in the registration of massive point sets. Two more experiments were made to compare BCPD++ and BCPD in datasets containing 3D scans of human bodies and, once again, it was concluded that BCPD++ is much faster at solving the registration problem, also achieving very identical results to the ones achieved by the BCPD algorithm.

2.5.10 Anisotropic Generalized Bayesian Coherent Point Drift

AGBCPD is a **parametric, pairwise, probability-based, towards rigid transformations** PSR technique based on the Bayesian Coherent Point Drift (BCPD) technique [19]. Proposed by Zhang *et al.* [48], generalizes BCPD under the Variational Bayesian Inference (VBI) framework [19]. It considers both the positional and the normal vectors of points to find point correspondences more accurately and become more robust to noise and outliers.

Two variants of the AGBCPD were compared with ICP, CDP and the original BCPD on two 3D real datasets with scans of human bones. Two types of tests were performed, one being the alignment of two complete point sets and the other the alignment of a partial point set with a complete point set.

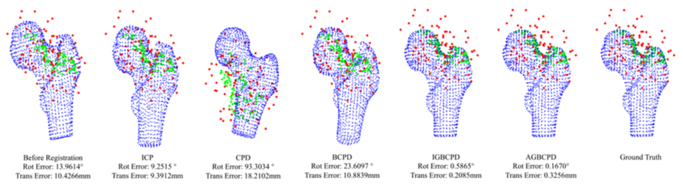


Figure 2.24: Results of the second experiment performed in [48].

According to the authors, the results showed that both variants of the AGBCPD achieved better results than the other three algorithms by a good margin. Among the other three algorithms, ICP and CPD seemed to be at the same level, with BCPD performing slightly worse than those two methods. The authors then conclude that, even though their method is slightly slower than the BCPD per iteration, it has a better convergence speed, robustness and registration accuracy than the other algorithms tested against it.

2.5.11 FlowNet3D

The FlowNet3D, proposed by Liu *et al.* [25] is the first deep learning based algorithm mentioned in this section. It is a **non-parametric, pairwise, correspondences-free and towards rigid transformations** PSR technique. It is composed of three key modules:

- point feature learning: is a PointNet++ [33] architecture that learns hierarchical features;
- point mixture: is a flow embedding layer It takes a pair of point clouds and for each point in point set X searches within a radius for a correspondent point in point set

Y. If it does not find one, it uses a neural layer to aggregate flow votes from all the neighbor points. The computed flow embeddings are further mixed through a few more set conv layers to obtain spatial smoothness. This also helps resolve ambiguous cases (for example, points on the surface of a translating table) that require large receptive fields for flow estimation;

- flow refinement: is a set upconv layer and does the up sampling step by receiving the source points and a set of target locations to propagate the point features to.

The architecture of the FlowNet3D consists of four set conv layers, one flow embedding layer, four set upconv layers and one linear flow regression layer to output the prediction. This architecture is summarized by Figure 2.25. The FlowNet3D also makes it possible to include more point features such as color and Lidar intensity.

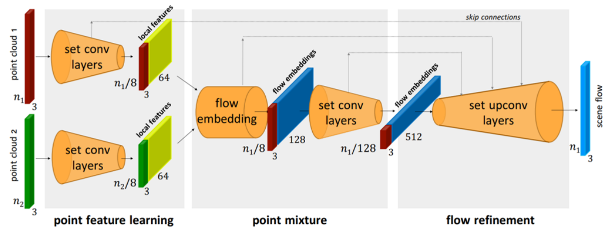


Figure 2.25: FlowNet3D architecture, extracted from [25].

The authors of FlowNet3D use 3D end point error (EPE) and flow estimation accuracy (ACC) to measure the performance of the algorithms tested. Two tests were made. FlowNet3D was compared with another deep learning based technique (FlowNet-C), with three baseline deep models (EM, LM and DM) and with the traditional method ICP.

The first experiment was made on a synthetic FlyingThings3D dataset which consists of stereo and RGB-D images rendered from scenes with multiple randomly moving objects sampled from ShapeNet. Results of this first experiment show that FlowNet-G stands out negatively, ICP is able to achieve better results than 2 out of the 3 baseline models (EM and LM) but FlowNet3D is the best method achieving considerably better results than all the other methods.

For the second experiment, the KITTI scene flow (real data) dataset was used, and Figure 2.26 shows acne flow predictions by FlowNet3D on four KITTI scans.

It is designed for evaluations of RGB stereo based methods. In this experiment FlowNet3D was compared with PRSM and ICP once again. The results once again show that FlowNet3D can outperform ICP, with PRSM achieving the worst results of the three. It is also shown that with some fine tuning FlowNet3D can achieve even better results. The authors conclude then that FlowNet3D showed its competitive or better results to various baselines and prior arts on both challenging synthetic dataset and real Lidar point clouds.

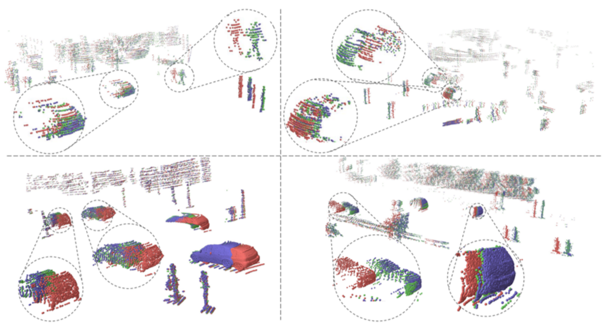


Figure 2.26: FlowNet3D scene flow predictions for four KITTI scans, extracted from [25].

2.5.12 HPLFlowNet

Proposed by Gu *et al.* [16], HPLFlowNet is a **parametric, pairwise, correspondences-free and towards rigid and non-rigid transformations** PSR technique. Inspired by Bilateral Convolutional Layers (BCL) and the permutohedral lattice, it proposes three new layer designs: DownBCL, UpBCL, and CorrBCL. First creates a hourglass-like structure starting by downsampling the two point patches using several DownBCL layers, the latest of which intercalated with some CorrBCL layers to fuse information from both point clouds and ends by up sampling the resultant patch in order to output the scene flow estimation. Figure 2.27 helps visualize this technique's architecture.

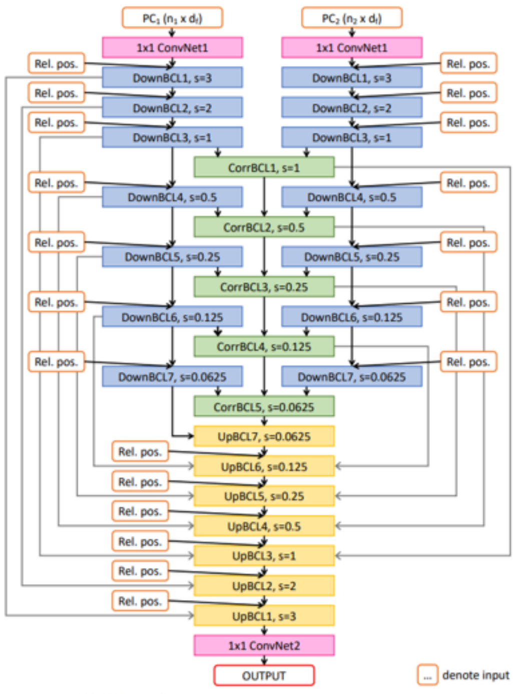


Figure 2.27: Visual representation of the architecture of HPLFlowNet, extracted from [16].

The authors of HPLFlowNet use several metrics to evaluate the results but the main metric used is the EPE3D. The HPLFlowNet is directly compared with ICP, FlowNet3D, SPLATFlowNet and the original BCL. The first test is performed on the FlyingThings3D (large-scale synthetic) dataset, where the model was trained. Results show that HPLFlowNet is better than any other technique in all 6 metrics used. FlowNet3D seems to be the second best algorithm by a small margin against both SPLATFlowNet and BCL. It should be mentioned that since the code for FlowNet3D is not available, these results were achieved by an implementation of the FlowNet3D made by the authors of this article. In the second test, to study the model's generalization ability, the KITTI Scene Flow (3D real data) dataset was used, without any prior fine-tuning given to HPLFlowNet and the results were exactly the same as the ones obtained in the first test. The last tests also prove that the model is able to obtain these results without sacrificing performance, on the contrary, actually being faster than FlowNet3D on both datasets used and for point sets of different sizes.

2.5.13 Point Net Encoding for Point Cloud Alignment

The Point Net Encoding for Point Cloud Alignment (i-PCRNet) framework proposed by Sarode *et al.* [37] is a **parametric, pairwise, correspondences-free and towards rigid transformations** PSR technique that can produce approaches that are shape specific or general to unseen shapes, depending on the prior information about the shape of the object formed by the point clouds. It does not propose a single registration technique but instead a framework that generates a number of registration approaches, including some already known methods.

The performance of the various approaches produced by the framework depends on factors such as the prior information about the shape of the object, noise in the measurements and computation time. The approach uses PointNet [33] to encode the shape information of a template and a source point cloud as feature vectors, and estimates the pose that aligns these two features using data driven techniques. The pose estimation from the features are carried out either using a number of fully connected (FC) layers or using classical alignment techniques such as Lucas-Kanade (LK) algorithm (results in good generalizability, but is not robust to noise). The FC layers are robust to noise, but are not generalizable to shapes unseen during training. The framework also uses Iterative Alignment which iteratively improves the calculated transformation. Figure 2.28 visually represents this Iterative Alignment process.

For the tests of i-PCRNet, the authors used models from ModelNet40 dataset and compared i-PCRNet, PointNetLK and ICP. The main metric used was the AUC which expresses the success of registration.

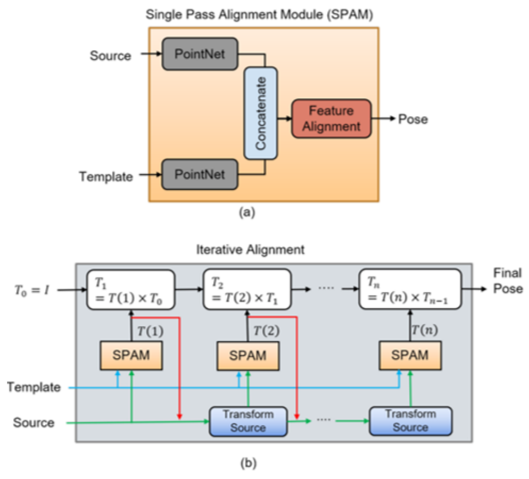


Figure 2.28: Visual representation of Iterative Alignment performed by i-PCRNet, extracted from[37].

In the first test, with objects from categories not seen during training and without the presence of noise, results showed that PointNetLK was the best method and i-PCRNet was the worst but, on the second test, with objects from the same category as the objects from the training phase, i-PCRNet registered a massive improvement reaching the same accuracy as PointNetLK in this test, and both being better than ICP.

Tests performed on partial data show that i-PCRNet gets worse than ICP the more points are missing from the point set, especially if not trained with partial data.

Tests performed with noisy data show that this is probably the biggest advantage of i-PCRNet, as it is the one that achieves the better results, unlike PointNetLK which is by far the worst of the three.

Even if trained without noisy data, i-PCRNet is only slightly worse than ICP. In the last test, i-PCRNet is compared with PointNetLK, ICP, DCP and Go-ICP in both accuracy and runtime. In terms of pure accuracy, Go-ICP gets the better results, with i-PCRNet being a close second. ICP, DCP and PointNetLK come after in this order but all far from the first two and even far from each other. But, when it comes to runtime, while almost all methods have similar values, Go-ICP presents a runtime 3 orders of magnitude greater than all others, which means that considering an accuracy/runtime trade-off, i-PCRNet seems to be the best method among the ones tested.

2.5.14 Non-Rigid Point Set Registration Network

Proposed by Wang *et al.* [41], PR-Net is a **parametric, pairwise, correspondences-based and towards non-rigid transformations** PSR technique. The architecture of this technique, which can be visualized with the help of Figure 2.29, can be divided in three steps:

- learn the shape descriptors of the point sets which are stored in a grid format;
- calculate the shape correlation between the two grids;
- learn the parameters of the desired transformation based on the shape correlation grid.

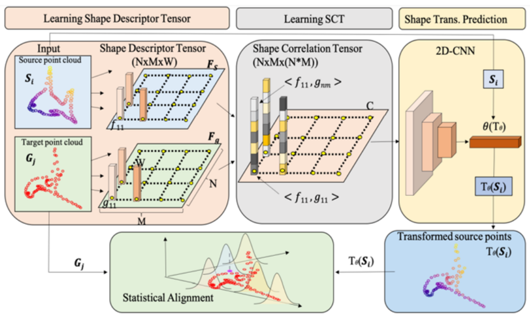

Learning Shape Descriptor Tensor
Learning SCT
Shape Trans. Prediction
Input
Source point cloud
Shape Descriptor Tensor (NxMxW)
Target point cloud
Shape Correlation Tensor (NxMx(N*M))
2D-CNN
Transformed source points
Statistical Alignment

Figure 2.29: Visual representation of the architecture of PR-Net, extracted from [41].

PR-Net is compared with CPD in a 2D synthetic fish dataset according to accuracy (measured by the distance between predicted result and ground-truth) and runtime. When it comes to accuracy, PR-Net presents only slightly a slightly worse value of average accuracy than CPD but with a smaller standard deviation which suggests it can provide a more stable registration. The biggest difference between the two techniques is in the runtime, since PR-Net was able to go through both the training and testing datasets in just 13 minutes while CPD took 24 hours in total, because the CPD treats every new pair of point sets independently and has to repeatedly register them from the start.

The authors of PR-Net performed more tests to evaluate the performance of the technique in the presence of different kinds of noise and deformations. PR-Net can still maintain a very high accuracy in the presence of noise and it can still achieve very good results when faced with deformation levels up to 1,0. Above this value, there is a sharp decrease of the accuracy of PR-Net, but these values already represent very heavy deformations.

2.5.15 PointNetLK

PointNetLK, proposed by Aoki *et al.* [3], modifies the original and traditional Lukas-Kanade method [26] and is unrolled as a recurrent neural network from which PointNet [33] is then integrated. It is a **parametric, pairwise, correspondences-based and towards rigid transformations** PSR technique. Point cloud inputs source and template received by this technique are passed through a shared MLP (Multi-Layer Perceptron), and a symmetric pooling function, to compute the global feature vectors. The optimal transformation parameters are found, which are used to incrementally update the pose of the source point cloud and then its global feature vector is recomputed. Figure 2.30 represents the architecture we just described.

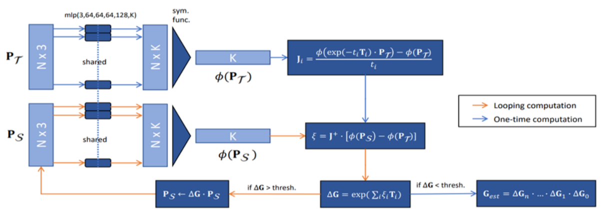

mlp(3,64,64,64,128,K)
sym. func.
$\mathbf{P}_{\mathcal{T}}$
N x 3
shared
N x K
K
$\phi(\mathbf{P}_{\mathcal{T}})$
$\mathbf{J}_i = \frac{\phi(\exp(-t_i \mathbf{T}_i) \cdot \mathbf{P}_{\mathcal{T}}) - \phi(\mathbf{P}_{\mathcal{T}})}{t_i}$
Looping computation
One-time computation
$\mathbf{P}_{\mathcal{S}}$
N x 3
shared
N x K
K
$\phi(\mathbf{P}_{\mathcal{S}})$
$\xi = \mathbf{J}^+ \cdot [\phi(\mathbf{P}_{\mathcal{S}}) - \phi(\mathbf{P}_{\mathcal{T}})]$
$\mathbf{P}_{\mathcal{S}} \leftarrow \Delta\mathbf{G} \cdot \mathbf{P}_{\mathcal{S}}$
if $\Delta\mathbf{G}$ > thresh.
$\Delta\mathbf{G} = \exp(\sum_i \xi_i \mathbf{T}_i)$
if $\Delta\mathbf{G}$ < thresh.
$\mathbf{G}_{est} = \Delta\mathbf{G}_n \cdot \ldots \cdot \Delta\mathbf{G}_1 \cdot \Delta\mathbf{G}_0$

Figure 2.30: Visual representation of the architecture of PointNetLK, extracted from [3].

During training, a loss function is used which is based on the difference in the estimated rigid transform and the ground truth transform. The transformation that turns PS into PT (template) is calculated iteratively: the nth iteration calculates the transformation that turns the results of the (n-1)th iteration into PT. The final transformation is calculated by simply adding the transformations resulting from every iteration.

The tests for PointNetLK are performed on a ModelNet40 dataset containing models for 40 object categories.

In the first test, PointNetLK is trained with 20 object categories and tested in the same 20 object categories. Comparing it directly with ICP shows that rotation errors from the PointNetLK are only slightly smaller than ICP but translation errors are considerably smaller. To assess the generalization capability of PointNetLK, in the second test it is trained with 20 object categories and tested on the remaining 20 object categories and the increase in both translation and rotation errors is almost neglectable. Figure 2.31 shows the results of some of these tests, where the orange points represent ICP's prediction and the blue points represent PointNetLK's prediction.

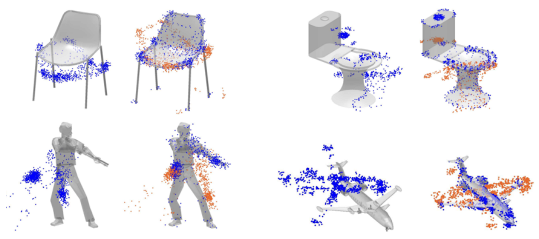


Figure 2.31: Example results of test cases performed to compare PointNetLK with ICP, extracted from [3].

It is also tested on the Stanford bunny dataset, once again being better and faster than ICP. The third test serves to conclude that average pooling is the best way for PointNetLK to account for noise in data. The fourth test shows that, when registering only partial point sets, PointNetLK, when trained with full point sets, is only slightly better than ICP but, if trained with partial point sets, its accuracy improves significantly.

In the last test, the authors show thatPointNetLK can align even different objects as long as part of the same category (e.g. two different airplanes) minimizing the error by making both objects face the same direction and, once again, PointNetLK seems to achieve better results than ICP on most cases.

2.5.16 Groupwise Registration using T-Mixture Model

TMM is a **parametric, groupwise, probability-based and towards rigid transformations** PSR technique proposed by Ravikumar *et al.* [34]. The technique is based on the Student's t-distribution which is much more robust to noise when compared to the Gaussian distribution, as shown in Figure 2.32, and proposes two versions based on a T-Mixture Model and a multi-resolution extension of the T-Mixture Model.

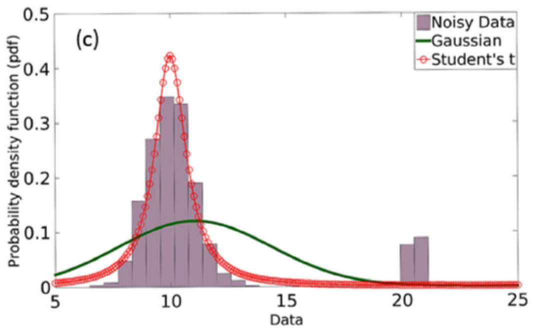


Figure 2.32: Comparison between Student's t distribution and Gaussian distribution in the presence of noise, extracted from [34].

This technique also uses a Forward-Scheme because the point sets are treated as transformed observations sampled from the model. The pipeline of this technique can summarily be described in two steps:

- calculating the mean point set based on all point sets of the group;
- align each point set with the mean point set.

The authors of this technique compared it with several others (namely CPD, SpSSM, GMM and JRMPC) in several tests including 2D and 3D point sets and different levels of noise. Figure 2.33 shows two examples of the tests performed and the results achieved by TMM and mrTMM.

Results showed that mrTMM is a small improvement to TMM in all tests, but both versions of this technique outperform the remaining tested techniques, particularly in the presence of a significant amount of outliers, while also being able to tolerate missing data.

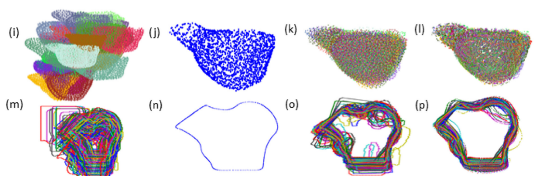


Figure 2.33: Results of TMM (third column) and mrTMM (fourth column) in two group registration problems, extracted from [34].

2.6 Summary

In this chapter, we focused on the analysis of the literature regarding Point Set Registration algorithms.

In section 2.1, we exposed the main concepts about PSR techniques, mainly the definition of PSR and the main challenges faced by these techniques. Section 2.2 described the usual pipeline of PSR techniques, using the example of ICP [28]. Section 2.3 explained the categories PSR techniques can be fitted into based on their characteristics. In particular we explored the difference between pairwise/groupwise techniques, parametric/non-parametric techniques, rigid/non-rigid transformations and correspondences-based/correspondences-free techniques. Section 2.4 mentioned the type of experiences used to measure the results of PSR techniques, as well as the most common datasets and metrics used on those experiences. Lastly, Section 2.5 surveyed the PSR techniques found during this state-of-the-art review, also categorizing them based on the concepts of section 2.2, identifying their main characteristics and commenting on the results presented by each technique's authors.

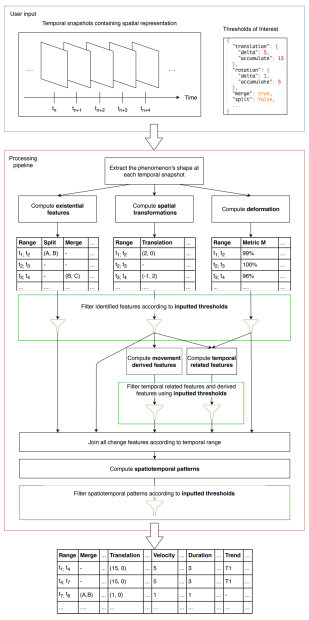
Chapter 3

Solution

This chapter starts by addressing the problem already presented in Chapter 1. Section 3.2 presents the techniques chosen to be implemented and tested, as well as why they were chosen. Section 3.3 exposes the implementation details of our solution for each algorithm and section 3.4 summarizes this chapter.

3.1 Problem

As already mentioned in Chapter 1, a conceptual framework [6] was previously created towards a solution for automatic detection of spatiotemporal events. Figure 3.1 shows a visual representation of the conceptual framework's pipeline for extracting information about spatiotemporal events which are relevant according to the user interest.


User input
Temporal snapshots containing spatial representation
...
...
Time
t_n
t_{n+1}
t_{n+2}
t_{n+3}
t_{n+4}
Thresholds of Interest
Processing pipeline
Extract the phenomenon's shape at each temporal snapshot
Compute **existential features**
Compute **spatial transformations**
Compute **deformation**
Filter identified features according to **inputted thresholds**
Compute **movement derived features**
Compute **temporal related features**
Filter temporal related features and derived features using **inputted thresholds**
Join all change features according to temporal range
Compute **spatiotemporal patterns**
Filter spatiotemporal patterns according to **inputted thresholds**

```
{
  "translation": {
    "delta": 5,
    "accumulate": 15
  },
  "rotation": {
    "delta": 1,
    "accumulate": 5
  },
  "merge": true,
  "split": false,
  ...
}
```

Range	**Split**	**Merge**	...
$t_1; t_2$	(A, B)	-	...
$t_2; t_3$	-	-	...
$t_3; t_4$	-	(B, C)	...
...	....	...	...

Range	**Translation**	...
$t_1; t_2$	(2, 0)	...
$t_2; t_3$	-	...
$t_3; t_4$	(-1, 2)	...
...	....	...

Range	**Metric M**	...
$t_1; t_2$	99%	...
$t_2; t_3$	100%	...
$t_3; t_4$	96%	...
...	....	...

Range	**Merge**	...	**Translation**	...	**Velocity**	...	**Duration**	...	**Trend**	...
$t_1; t_4$	-	...	(15, 0)	...	5	...	3	...	T1	...
$t_4; t_7$	-	...	(15, 0)	...	5	...	3	...	T1	...
$t_7; t_8$	(A,B)	...	(1, 0)	...	1	...	1	...	-	...
...	....	...	...	...	...	...	...	...	...	...

Figure 3.1: Visual representation of the framework's pipeline for extracting information about spatiotemporal events.

The framework receives as inputs a sequence of snapshots describing the contour boundaries of a spatiotemporal phenomenon and other inputs given by the user in the form of a set of thresholds. It starts by extracting the shape of the phenomenon represented at each temporal snapshot. It then computes the transformations that the shape of the phenomenon suffers throughout the temporal snapshots using a PSR algorithm. This is done by having the PSR algorithm calculate the transformations (translation, rotation, scale or immutability) suffered by the shape of the phenomenon between every pair of consecutive temporal snapshots. By gathering all the transformations calculated, the framework is able to describe the movement of the shape of the phenomenon for the hole sequence of temporal snapshots and filter the events of interest based on the thresholds inputted by the user. In the future, the framework will also be able to compute existential transformations like splits and merges and movement and temporal derived features. Lastly, the framework organizes the information to be presented in an easily readable format.

Events of Interest

The user can define as events of interest the transformations to be detected by the framework as a set of thresholds, thus allowing the quantification of events that the user is interested in. For each transformation, when its value surpasses the defined threshold, an event of interest is considered to be found.

There are four types of thresholds:

- the delta threshold refers to the exceeding of a threshold by a change feature identified from two sequential temporal snapshots. For example, if the user sets a delta threshold for a translation of 10 units and, between two consecutive temporal snapshots, a translation of 15 units occurs, this translation will be considered an event of interest by the framework.
- the absolute accumulated threshold refers to the total amount of change accumulated throughout a sequence of timestamps that triggers the detection of an event of interests. For example, if the user sets an absolute accumulated threshold for a translation of 10 and, between two consecutive temporal snapshots, a translation of 5 units occurs, it will not be considered an event of interest. But, if between the next pair of temporal snapshots, a translation of 8 units occurs, the framework will detect a translation of 13 units as an event of interest.
- the directed accumulated threshold derives from the absolute accumulated threshold, but only takes into account a transformation happening during a sequence of timestamps if it is consistent in its direction. This means that every time a transformation inverts its direction, the accumulation for this threshold gets reset. For

example, if the user sets a directed accumulated threshold for a translation of 10 and, between two consecutive temporal snapshots, a translation of 5 units occurs and, in the next pair of temporal snapshots, a translation of -8 units occurs, no event of interest will be detected because the two translations have opposing directions.

- the noise epsilon is a noise filtering threshold used to define a value under which any amount of a transformation is considered to be noise. For example, if the noise epsilon is set to 0.5 degrees for a change in direction (rotation) and between two given timestamps the framework detects a 0.02 degrees rotation, it will be ignored by all other thresholds.

So far, a single PSR technique was tested in the context of the framework, but Carneiro [6] verified that the PSR technique used may have a great impact on the quality of the results achieved by the framework.

Carneiro performed a series of tests consisting of a dataset (sequence of snapshots representing the contour boundary of a phenomenon), multiple thresholds inputted by a user defining the user is interest in particular types of events, and the expected results to be used as ground-truth. To evaluate the results of the framework on those tests, two metrics (section 2.5)) were used and, according to the results achieved, it was concluded that the framework achieves very satisfying accuracy values but also that it “is highly dependent on the accuracy of the point set registration algorithm chosen to compute the spatial transformations”.

Given the previous context, this study tries to address the following research questions:

RQ1: Why do the results depend on the PSR technique used?

RQ2: How much do the results depend on the PSR technique used?

Considering the existence of some more recent PSR techniques based on Deep-Learning, a third research question emerges:

RQ3: How do the Deep-Learning based PSR techniques compare to the traditional PSR techniques in the context of this framework?

3.2 Techniques to be tested

The authors of the framework [6] stated that the PSR technique to be used, in order to be compatible with the framework, must be pairwise (compares two shapes), parametric (uses fixed number of parameters) and support affine transformations. Affine transformations are the most basic non-rigid transformations and preserve collinearity (all points lying on a line initially still lie on a line after transformation) and ratios of distances (the midpoint of a line segment remains the midpoint after transformation).

As the present study stand on top of the described conceptual frame, and based on the performed state of the art review, two traditional techniques - GMM-REG [21] and KCReg [38] identified in the survey of [51] - and a Deep Learning based technique - PR-Net [41] - were selected to be tested with the proposed framework.

The selection of these PSR techniques goes beyond the fact that each presents good results in the tests performed by their respective authors. They were selected because they differ from each other in the way they approach PSR problems:

- GMM-REG considers the point sets as Gaussian Mixture Models [21] and tries to align them;
- KCReg uses the concept of Kernel Correlation [38], a function of point set entropy which it tries to maximize;
- PR-Net, on the other hand, stands out for adopting the new computational paradigm that is Deep-Learning [41].

3.3 Implementation details

The latest version of the conceptual framework is described by its authors [6] as a server-side application utilizing the microservices architectural approach, in which each microservice represents a module. Figure 3.2 shows a visual representation of the architecture of the framework.

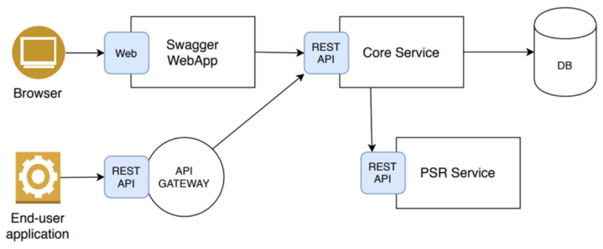


Figure 3.2: Visual representation of the framework's architecture, extracted from [6].

The framework exposes two artifacts to the end-user: a web application and an API. It also has a database and the Core Service which is responsible for data management (e.g. data storage, attainment of features of interest, etc.). The computation of the spatiotemporal change features is done by the PSR service which is the most relevant module for this study.

The already implemented algorithm in the PSR Service is the Coherent-Point-Drift (CPD) [31]. The authors of the conceptual framework used a Python implementation of the algorithm and decided to build the server using Flask[1].

To keep future studies and corresponding prototype developments as compatible as possible with the current version of the conceptual framework, we decided to created three micro-services (one for each new PSR technique to be tested) also using Flask. Therefore, the new PSR service modules fit and extend the original framework in a seamless fashion.

3.3.1 General Procedure

The general procedure adopted was

- to download the source code corresponding to each PSR algorithm from the official source;
- to download and install the necessary packages/software to run the algorithm;
- to study the algorithm execution and run with the author provided test datasets (when provided);
- to study the code to define execution features, requirements, starting functions, outputs and output structure;
- adapt the code in order to return the necessary information about the transformations that lead to the alignment of the point sets calculated by the algorithm.

The next sections provide implementation details for each of the selected techniques.

GMM-REG integration

The function in the GMM-REG code that aligns two point sets receives, as input a parameter, an *.ini* file with several information such as point set data and thresholds. We then create that *.ini* file in runtime so we can use that function in its original format.

This algorithm considers the two input point sets as model and the scene, where the first, is the point set for which we want to align with the second. Before performing the alignment, the algorithm normalizes both point sets and, for each, saves the centroid and the scale factor . Then computes the solution point set which is the "prediction". When denormalizing the third point set, the algorithm uses the centroid and the scale factor of the original scene point set, basically ensuring that the position of the predicted point set and its scale will be correct.

[1]`https://flask.palletsprojects.com/en/1.1.x/`

It was identified that this algorithm does not output any resulting transformation matrix and, in order to evaluate the estimate point set, we calculate the transformations detected by comparing the model point set (the initial one) with the final point set predicted by the algorithm. For the translation, we calculate both centroids and then the vector that goes from one centroid to the other, as presented on Figure 3.3.

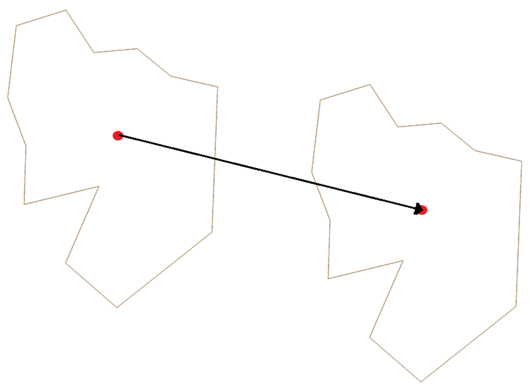


Figure 3.3: Visual representation of the vector between the centroids of two point sets.

This way of calculating translations works well, simultaneously with the translation, only other affine transformations occur. If the translation occurs at the same time as an irregular deformation, small errors might be introduced by this calculation.

For the rotation, we treat each side of the polygon described by the final point set as a vector, as well as the sides of the polygon represented by the initial point set. Then, for each vector of the final point set, we calculate the angle between it and the respective vector of the initial point set and lastly the average of all the angles to obtain the rotation value. Figure 3.4 visually represents the calculation of the rotation angle between one pair of sides of the polygons that represent the point sets.

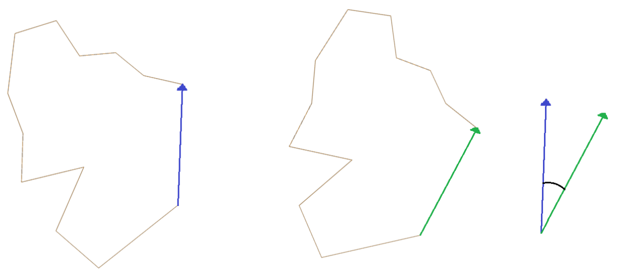


Figure 3.4: Visual representation of the calculation of the angle between the vectors that represent a pair of corresponding sides of two polygons.

Like the calculation of the translation, this method of calculating rotations will introduce errors when happening at the same time as irregular deformations, but, in those situations, the accuracy of the algorithm itself is expected to be lower.

For the scale, we use the same method, calculating the ratio between the sizes of each pair of vectors instead of the angle between them.

Since we do not extract the transformations from the algorithm directly, we might be introducing small errors resultant from the calculations performed. Still, by using the resultant point set predicted by the algorithm, we are using its ability to adapt the shape of the point sets to small deformations that will affect the calculation of the transformations.

PR-Net integration

The PR-Net algorithm calculates a transformation matrix but it comes in a shape of 18x1 which does not allow the directly extraction of the transformations required by the conceptual framework. Furthermore, upon our first attempt at running the algorithm, the obtained results were extremely underwhelming. So, as a first attempt to improve those results, before passing the two point sets to the model, inspired by the solution of the GMMREG algorithm, we allign the center of both point sets with the origin of a two dimensional axis and we also normalize them, storing their real center and the scaling factor to revert this operation after receiving the prediction of the algorithm. This way, the algorithm is not used to calculate translations or scales, but still gives information about rotations and deformations.

The input for the training process of the algorithm is a single point set, so the point set that creates the polygon of Figure 3.5 was used.

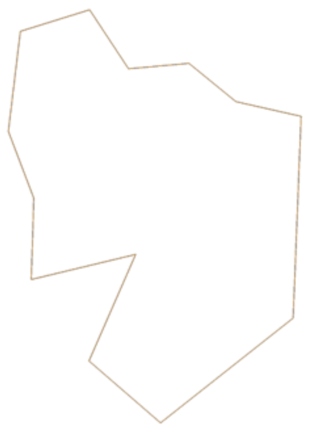


Figure 3.5: Polygon represented by the point set used to create training and testing datasets.

The algorithm uses that point set to create the training and testing sets. This is done as follows:

- starting with a 18x1 transformation matrix that represents immutability, 18 random values are created and added to the values of the transformation matrix. The range of the random values can be controlled through a parameter that defines the deformation level that we want to apply to the point set;

- then, the resulting transformation matrix is applied to the original point set and both the resultant point set and the transformation matrix used to create it are stored.

Figure 3.6 compares two different levels of deformation experimented with. On both sides of the image, the blue dots represent the original point set and the red crosses represent the deformed version of the original point set. The right side of the image uses double the deformation as the left side.

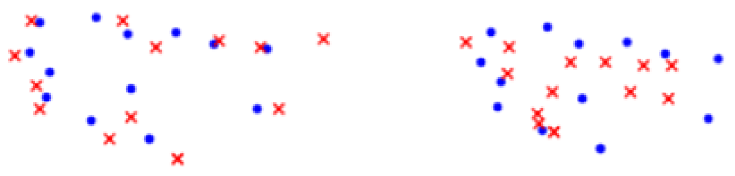


Figure 3.6: Comparison between different levels of deformation.

KCReg integration

The KCReg algorithm is implemented in MATLAB so the python library *matlab.engine* is used in order to allow the python code to use the MATLAB functions that compose the algorithm.

The algorithm is capable of calculating three types of transformations: euclidean, affine and projective. Euclidean refers to rigid transformations and projective shows how the perceived objects change when the view point of the observer changes[2]. Since we are working in the context of affine transformations, that is the type of transformation we focused on.

The KCReg algorithm calculates a 3x2 transformation matrix from which the affine transformations can be extracted, so in this case we can directly extract that matrix and convert it in the affine transformation's parameters.

3.4 Summary

This chapter described the proposed solution that we created to achieve the objectives of this dissertation. It started by characterizing the problem and identifying three research questions in section 3.1. Then, section 3.2 presented the techniques to be implemented, as well as the reasons that led us to choose this techniques and section 3.3 detailed the implementation of each technique.

[2]`https://www.graphicsmill.com/docs/gm5/Transformations.htm`

Chapter 4

Testing Protocol

This chapter documents the testing procedure that led to the extraction of the results later presented and discussed in chapther 5. Section 4.1 describes the testing pipeline. The metrics used to evaluate the results are presented in section 4.2. Section 4.3 details every test and their purpose. Section 4.4 describes the procedure performed to run each test and section 4.5 summarizes the chapter.

4.1 Testing pipeline

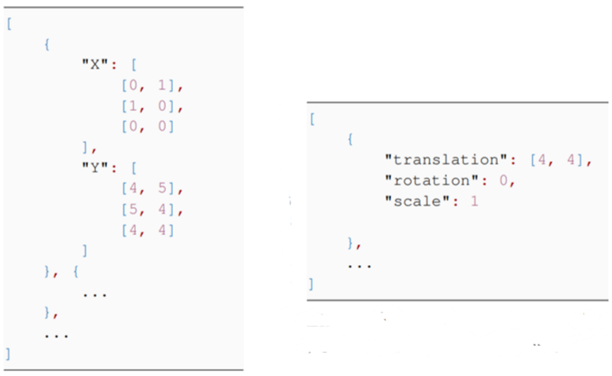
Carneiro *et al.*[6] already suggested and implemented a testing pipeline. Each experiment consists of a dataset (2D snapshots of the contour of a spatiotemporal phenomenon), a set of thresholds (explained in section 3.1) defined by the user, describing its degree of interest, and the expected result which is used as the ground-truth.

The core service of the framework sends a request with the post method to the service of one of the PSR algorithms (the services of all the algorithms work in the same way). As input, the service receives a listing of pairs of point sets. In each pair, "X" represents the source point set and "Y" the target point set. The output is the list of identified transformations, where an element in the list corresponds to the registration of the element in the same index in the input list. Figure 4.1, extracted from [6], shows an example of an input sent by the core service and respective output given by the algorithm's service.

The next sections address the concepts required to conduct testing of the PSR technique implementation with the conceptual framework.

4.2 Metrics

The results provided by the framework are then evaluated according to the two metrics also created for measuring this framework's ability to correctly identify the transformations happening to a point set. These metrics are summarily described in section 2.4 but



```
[
    {
        "X": [
            [0, 1],
            [1, 0],
            [0, 0]
        ],
        "Y": [
            [4, 5],
            [5, 4],
            [4, 4]
        ]
    }, {
        ...
    },
    ...
]
```

```
[
    {
        "translation": [4, 4],
        "rotation": 0,
        "scale": 1

    },
    ...
]
```

Figure 4.1: Example of input sent by the core service and respective output given by the algorithm's servicem extracted from [6].

a more detailed explanation follows.

4.2.1 Metric M1

Metric M1 is based on the symmetric difference between two areas which, given two areas A and B, can be described as:

$$\mathbf{A \Delta B = (A \cup B) - (A \cap B)}$$

This value represents the area of A and B that is not part of the union of A and B. Figure 4.2 shows the area corresponding to A Δ B when comparing two polygons.

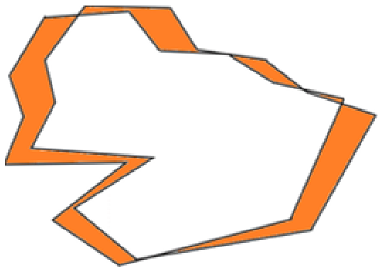


Figure 4.2: Representation of **A Δ B** (in orange).

Metric M1 is different from the symmetric difference between two areas. First it uses the concept of convex hulls. A polygon is convex when all its interior angles are less than 180 degrees and a convex hull is the smallest convex shape that can enclose a polygon. Figure 4.3 shows the convex hull for the polygon used in Figure 4.2.

When comparing two polygons, the metric M1 is calculated as the ratio between the area of the smallest convex hull of the two polygons and the area of the convex hull that encloses both polygons. Using the same example of Figure 4.2, M1 would be the ratio between the area colored yellow and the area colored both in yellow and blue in Figure 4.4.

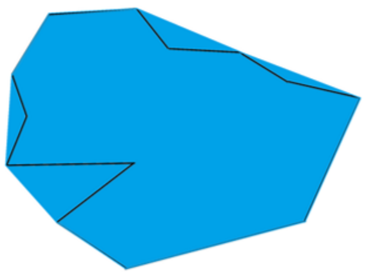


Figure 4.3: Convex hull of a polygon (in blue).

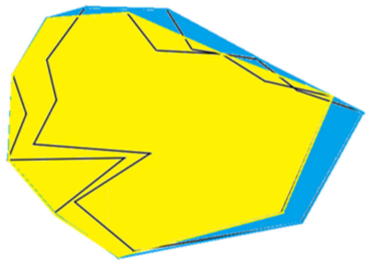


Figure 4.4: Visual representation to help understand the calculation of metric M1.

So both values can be used to calculate the alignment of two polygons, but representing two different things. The symmetric difference represents an error while metric M1 can be better described as a form of accuracy. This means that aligning the two shapes minimizes the symmetric difference between them (the best value is 0) but maximizes the metric M1 (the best value is 1).

There is an advantage in using metric M1 over the symmetric difference in this context. Unlike metric M1, symmetric difference does not take into account the distance between the two shapes when they don't intersect at all. In Figure 4.5, A Δ B and A Δ C have the same value (the maximum value for the symmetric difference which is the sum of the areas of both polygons) even though polygon B is closer to polygon A than polygon C.

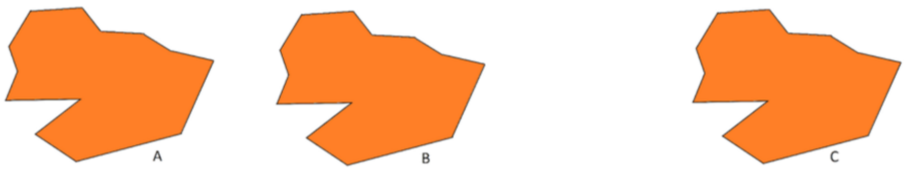


Figure 4.5: Polygon shapes not intersecting.

In the context of this framework this is relevant because if A is the ground-truth and B and C are two estimates, estimate B is better than C because it is closer to A but both have the same symmetric difference. Metric M1 does take into account the distance between the polygons because the further away they are from each other the bigger the area of the convex hull that encloses both polygons will be, lowering the value of M1 that we want to maximize, penalizing this way a worse estimate. Figure 4.6 shows a visual representation of the effect just mentioned.

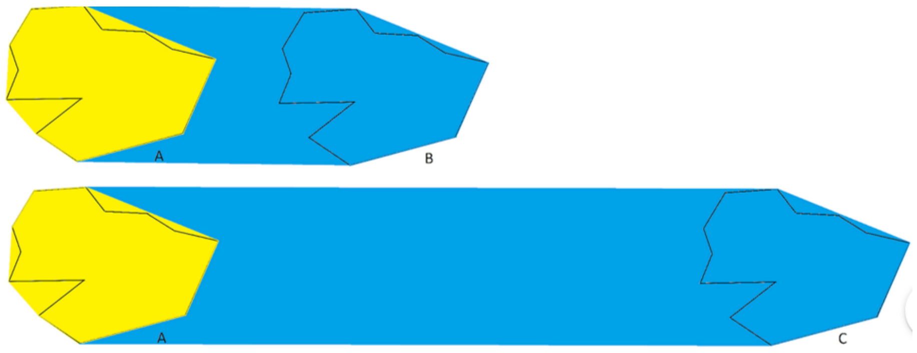


Figure 4.6: Visual representation of how metric M1 penalizes worse estimates.

4.2.2 Metric M2

Metric M2 is inspired by the concept of Longest Common Subsequence (LCSS) common to two sequences. For example, the longest subsequence between the sequences of characters “abcdef” and “abxdef” is “def”.

But, this metric does more than just finding the longest common subsequence between two sequences of transformations. It calculates the distance between the calculated results and the ground truth as the number of insertions or removals of transformations necessary to fully match the two, comparing to the worst case scenario (which is the removal of all transformations from the calculated results and the insertion of all transformations from the ground truth).

According to [6], this metric can me modelled as:

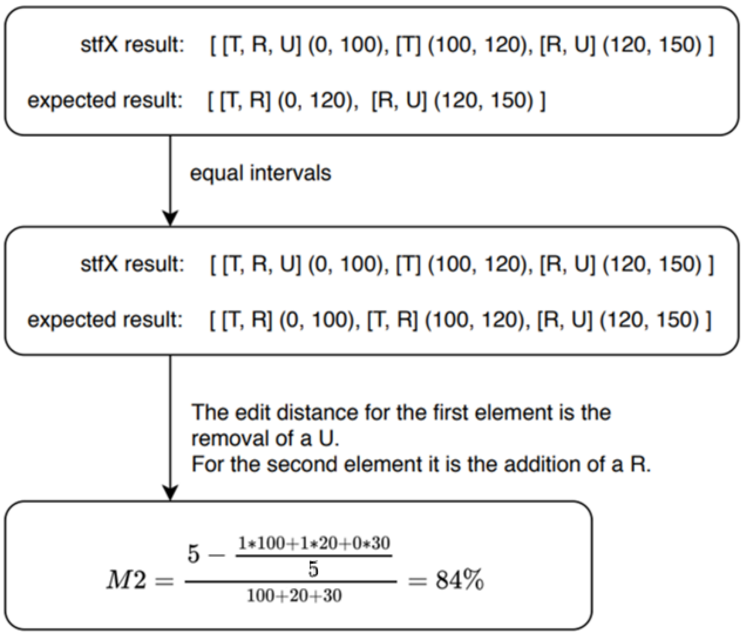
$$M2 = 1 - \frac{\sum_{n=1}^{N} \frac{|(A_n - B_n) \cup (B_n - A_n)| * R_n}{|A_n| + |B_n|}}{\sum_{n=1}^{N} R_n}$$

where A and B are arrays of sets of equal size where

$$\forall A_n \in A : B_n \subseteq T \wedge \forall B_n \in B : B_n \subseteq T,$$

T is the set of possible transformations, and R is an array containing the temporal ranges duration.

Figure 4.7, extracted from [6], shows an example of the calculation of this metric. In this figure, the elements inside the square brackets represent transformations, while the values inside round brackets indicate temporal ranges.


stfX result: [[T, R, U] (0, 100), [T] (100, 120), [R, U] (120, 150)]
expected result: [[T, R] (0, 120), [R, U] (120, 150)]
equal intervals
stfX result: [[T, R, U] (0, 100), [T] (100, 120), [R, U] (120, 150)]
expected result: [[T, R] (0, 100), [T, R] (100, 120), [R, U] (120, 150)]
The edit distance for the first element is the removal of a U.
For the second element it is the addition of a R.

$$M2 = \frac{5 - \frac{1*100+1*20+0*30}{5}}{100+20+30} = 84\%$$

Figure 4.7: Example of the calculation of metric M2.

The first step in the calculation of this metric is to split the framework's result and the expected result into equal intervals when they do not coincide. It the example of figure 4.7, the expected result is divided into two intervals (from 0 to 120 and from 120 to 150) but the result calculated by the framework is divided into three intervals (from 0 to 100, from 100 to 120 and from 120 to 150). It is then necessary to divide the first interval of the expected result into two (from 0 to 100 and from 100 to 120) in order to have the same intervals as the result calculated by the framework.

Then, for each interval, the distance between the calculated result and the expected result is calculated. This distance is the number of insertions and removals of elements from the list of transformations occurring calculated by the framework in order for that list to be equal to the list of transformations of the expected result from the same time interval. In this case, in the first interval (from 0 to 100), the distance is 1 because the removal of the transformation U (uniform scale) is necessary to match the two lists. In the second interval (from 100 to 120) the distance is also 1 because the insertion of the transformation R (rotation) is necessary. In the third and final interval (from 120 to 150), the distance is 0 because the list of transformations calculated by the framework for that interval is equal to the list of transformations of the expected result for the same time interval.

The last step is to apply the formula presented above. For each interval, it multiplies the distance between the lists of transformations by the range of that interval. Then, all those values are summed and then divided by the sum of the number of temporal intervals of each result (in this case, this value is 5 because originally, the result calculated by the framework had 3 intervals and the expected result 2). The resultant value is then subtracted to the sum of the number of temporal intervals of each result (once again, 5) and, finally, the resultant value is divided by the total range of the result (in this case, 150, or 100 + 20 + 30).

4.3 Test cases

Each PSR algorithm is subjected to eleven tests:

- test 1 consists of a translation of 500 units in the x-axis and 200 units in the y-axis, during 60 time units;
- test 2 consists of a final rotation of 10º clockwise, composed of a first rotation of 20º counterclockwise, during 30 time units, followed by a second rotation of 30º counterclockwise, also during 30 time units;
- test 3 consists of a uniform scale with value 2.52, resultant of the accumulation of three distinct uniform scales: a scaling of 1.4 in the first 20 time units, a scaling of 1.2 in the next 20 time units, and a scaling of 1.5 in the last 20 time units;
- test 4 consists of the phenomenon remaining immutable for 60 time units;

These first four tests serve to evaluate the ability of the framework to detect each transformation individually when receiving the calculations from a certain PSR algorithm. In each of these tests the thresholds are set with the exact values of the transformations occurring. For example, for test 2, the absoluteAcc threshold for the rotation is set to 50.

- test 5 consists of a sequence of transformations occurring non-simultaneously. The main goal of this test is to evaluate if the framework can detect transformations when they occur in a sequence, when based on the calculations of a particular PSR algorithm. For this test, the thresholds were set appropriately, using the directedAcc threshold for all transformations. The exact transformations happening are: immutability, from time unit 0 to time unit 30; a translation of 1000 units in the x-axis and minus 400 units in the y-axis, from time unit 30 to time unit 90; a rotation of 50º counterclockwise, from time unit 90 to time unit 120; a rotation of 50º clockwise, from time unit 120 to time unit 150; a uniform scaling of 1.3, from time unit 150 to time unit 180; immutability, from time unit 180 to time unit 240; and a uniform scaling of approximately 0.377, from time unit 240 to time unit 270. Figure 4.8 provides a visual representation of the transformations happening.

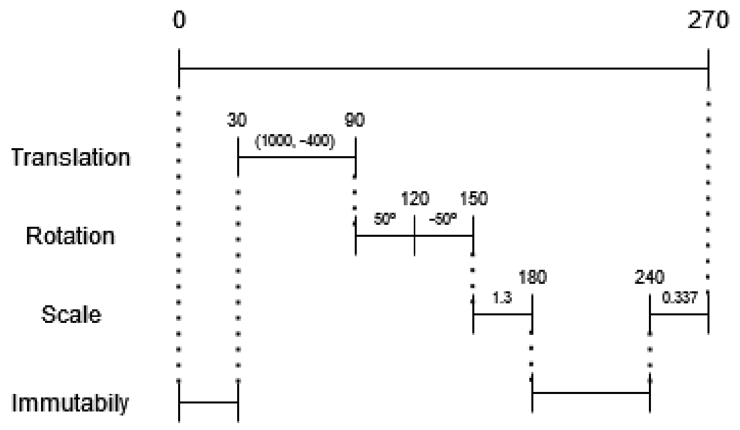


Figure 4.8: Visual representation of the sequence of tranformations happening in test 5.

- test 6 is formed by the exact same sequence of transformations as test five but with different thresholds. In this test, the absoluteAcc thresholds are used and, in the particular case of the rotation, the value of the threshold is set to 50° (instead of 49.5° like in test five). The goal of this test is to evaluate the impact that the choice of thresholds can have in the result of the framework.

Test 7, 8 and 9 serve two purposes. They evaluate if the framework can detect different transformations happening simultaneously and also how using different types of thresholds impacts the results of the framework in this context. A detailed characterization of each one of these tests follows.

- The transformations happening in test 7 are: a translation of minus 200 units in the x-axis and 40 units in the y-axis, a rotation of 45° clockwise and a uniform scaling of scaling factor 2, all during 60 time units. For this test, the thresholds are set for the exact values of each transformation. For example, the threshold related to the rotation is set to 45. Figure 4.9 provides a visual representation of the transformations happening.

- test 8 is formed by the exact same sequence of transformations as test 7 but the thresholds used are slightly less rigid. For example, for the rotation, the threshold is set to 44.5 instead of 45.

- test 9 is also formed by the exact same sequence of transformations as test seven with different thresholds. In this case the thresholds are set to very small values for the framework to be able to detect every single small transformation that might occur. Once again, for example, the threshold for the rotation is set to 1.

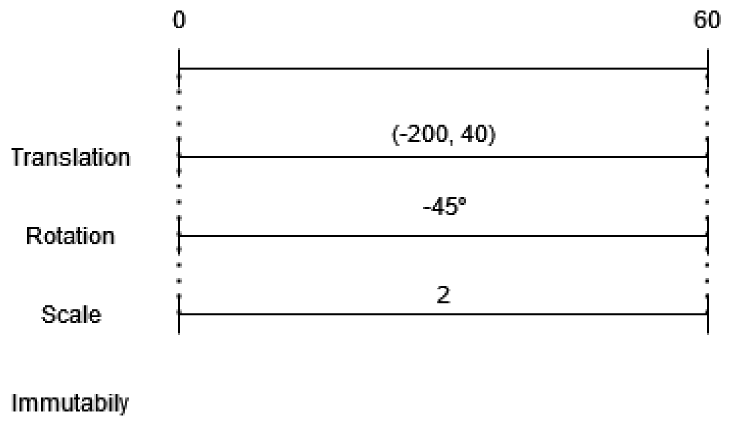


Figure 4.9: Visual representation of the sequence of tranformations happening in test 7.

- test 10 consists of the same sequence of transformations as test 5 and uses the same threshold values. The difference is that this is the only test that does not use noise filtering. The goal of this test is then to evaluate the impact of the removal of the noise filtering in the result of the framework;
- test 11 consists of a sequence of transformations in a very unorganized way. The goal of this final test is to evaluate the ability of the framework to achieve satisfactory results when confronted with very chaotic and complex scenarios. The exact transformations happening in this test are: a time-consuming translation, of 20 units in the x-axis and minus 10 units in the y-axis, from time unit 1 to time unit 190; simultaneous to the described translation, a rotation of 10° clockwise, from time unit 110 to 120, and a rotation of 10° counterclockwise, from time unit 150 to 160; immutability, from time unit 190 to time unit 210; a quick-paced change, from time unit 210 to time unit 225, where a translation of -30 units in the x-axis and 10 units in the y-axis and a uniform scaling of factor 0.4 co-occur; simultaneously, a translation of 5 units in the x-axis and a uniform scaling of factor 3, in the temporal range 225 to 255; and, at last, a rotation of 85° counterclockwise, in the temporal range 255 to 275, with a uniform scaling of factor 0.95 also occurring, from time unit 260 to time unit 265. Figure 4.10 provides a visual representation of the transformations happening.

4.4 Procedure

For each experiment, the following procedure is performed:

- the dataset is loaded into the server;

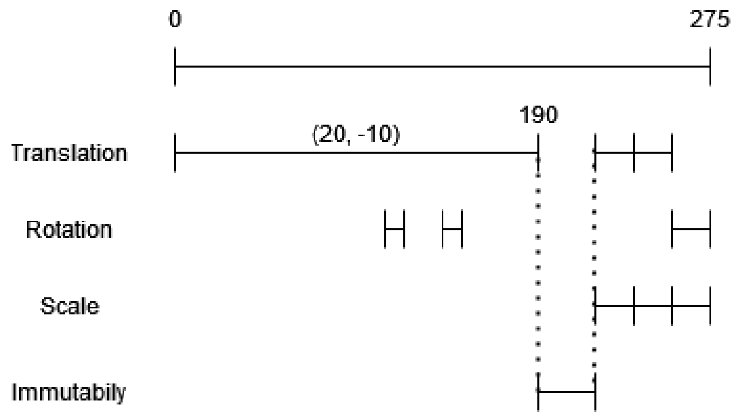


Figure 4.10: Visual representation of the sequence of transformations happening in test 11.

- the Core Service requests the PSR Service to compute the transformations present in the dataset and receives from it the values of the computed transformations;
- according to the thresholds previously defined, the features of interest are obtained;
- to validate the solution, the features of interest returned by the server are compared with the expected result (metric M2) and the spatial disposition of the phenomenon at the end of the ground truth animation is also compared with the one resulting from the transformations calculated by the framework (metric M1).

Figure 4.11 illustrates the previous pipeline.

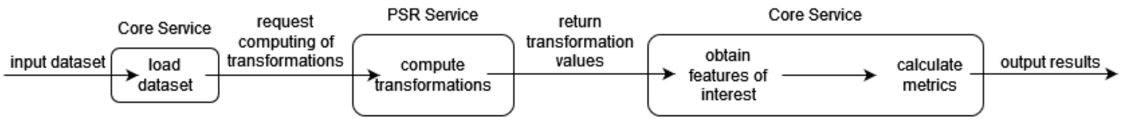


Figure 4.11: Diagram about the procedure for each experiment.

Since the Core module calls and receives the spatiotemporal events that occur in the dataset from the PSR Server, in order to test different PSR techniques, we created a different server for each technique and, by doing so, simply changing which server is used allows us to use the exact same testing pipeline for all the different implemented techniques with minimum effort.

4.5 Summary

This chapter presented all the aspects related to the testing phase of our solution. It started by describing the testing pipeline in section 4.1, where its test consists of a dataset, a set of thresholds defined by a user and the expected result which is used as the ground truth.

Section 4.2 explained the two metrics used to evaluate the results obtained: metric M1 which is used to compare the area occupied by the contour of the phenomenon calculated by the framework with the ground truth, and metric M2, used to evaluate the ability of the framework to identify the transformations happening and the respective time intervals. Section 4.3 exposed the details of each of the eleven test cases used to evaluate each PSR algorithm. It was shown that every test focuses on a different aspect, whether it being a particular transformation or a sequence of simultaneous transformations. Section 4.4 ended the chapter by detailing the procedure executed for each test, which consists in the loading of the dataset into the server, the computation of the transformations present in the dataset by the PSR algorithm, the filtering of the events of interest by the user inputted thresholds and calculation of the two metrics by comparing the result calculated by the framework with the ground truth.

Chapter 5

Results and Analysis

This chapter presents the results from the testing phase, as well as the corresponding result analysis. Section 5.1 reports the results of the 11 tests performed and presents a analysis of the three algorithms. Section 5.2 analysis PR-NET in more detail and documents attempts to improve its results. Section 5.3 explores our the results change with a few adjustments to the thresholds used in the tests. Section 5.4 studies how the algorithms behave when presented with pairs of point sets with different number of points and section 5.5 summarizes the chapter.

5.1 Experiments and Results

The three chosen algorithms were subjected to the eleven tests described in 4.1 and the values of the metrics M1 and M2 were extracted for each test in order to compare their performances among themselves and also with the results achieved by the algorithm that was already part of the framework, CPD. Tables 5.1 and 5.2 show the results achieved. Table 5.1 presents the results on tests 1 to five, which are the tests where no transformations happen simultaneously, and table 5.2 presents the results on tests 6 to 11, which are the tests where transformations happen simultaneously.

Table 5.1: Results of tests 1 to 6.

	1		2		3		4		5		6	
	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
CPD	1	1	1	1	1	1	1	1	1	1	0.476	0.740
GMMREG	1	1	0.916	0	1	1	1	1	0.592	0.889	0.652	0.649
PR-NET	0	0.500	0	0.444	0	0.500	0	0	0.002	0.272	0.002	0.269
KC-REG	1	1	0.916	0	0.157	0	1	1	0.129	0.667	0.339	0.740

These results allow us to make a first evaluation of each algorithm:

Table 5.2: Results of tests 7 to 11.

	7		8		9		10		11	
	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
CPD	0.247	0	0.768	0.685	0.841	0.925	1	0.665	0.558	0.800
GMMREG	0.250	0.142	0.769	0.750	0.667	0.925	0.999	1	0.333	0.800
PR-NET	0	0.932	0	0.872	0	0.990	0.002	0.272	0	0.536
KC-REG	0.247	0	0.769	0.685	0.841	0.925	0.135	0.440	0.002	0.601

- PR-NET greatly under performs to the point that in every single test the value of the metric M1 is either zero or very close to zero.
- KC-REG manages to achieve results on the level of the CPD algorithm on about half of the tests (1, 4, 6, 7, 8 and 9) but also under performs on the remaining tests.
- GMMREG is the only algorithm that manages to achieve results comparable to the CDP algorithm, even besting it in some tests (7, 8 and 10).

The following subsections analyze other aspects of the testing phase.

5.2 PR-NET analysis

Overall, the PR-NET seems to lack the capacity to accurately align the points of two point sets. It makes moderately good predictions for most points but a small number of points always ends up getting a terrible estimate which drastically drags down the accuracy of the hole prediction. Figure 5.1 shows an example of an alignment performed by the PR-NET algorithm where the problems just mentioned are present.

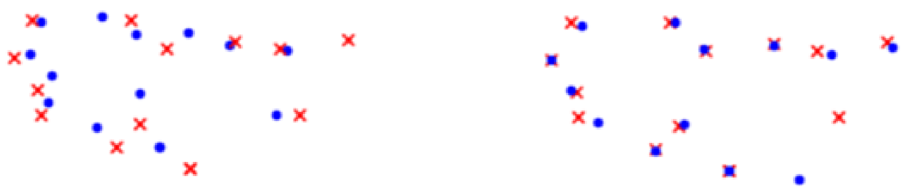


Figure 5.1: Example of alignment performed by PR-NET.

On the left side of the image, the blue dots represent the initial point set and the red crosses represent a second point set. The goal of the algorithm is to align the point set represented by the blue dots with the point set represented by the red crosses. On the right side of the image, the red crosses represent the same point set as the red crosses on the left side of the image, and the blue dots represent the original point set after the algorithm tried to align it with the point set represented by the red crosses.

As we can see on the right side of the image, although with associated error, most points are well aligned. The exception is the point in the low right corner of the image, which is very poorly aligned. Since we take every point into consideration when calculating the transformations that align the two point sets, a big misalignment of one point alone is enough to introduce a great error in the calculations and negatively impact the results of the algorithm in both metrics.

When analysing the behaviour of the algorithm, the first problem noticed was the fact that the PR-NET cannot detect immutability, as shown in figure 5.2.

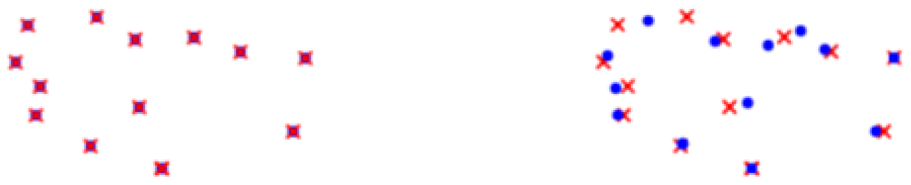


Figure 5.2: Demonstration of lack of ability of the PR-NET algorithm to detect immutability.

It was hypothesized that this happens because the original training and testing datasets used do not contain cases of immutability. This happens because of how the datasets are created. This algorithm represents transformations as a 18x1 matrix and only needs to be given one point set to create the datasets. It creates each element of the datasets by starting with a matrix that represents immutability and then randomly applying noise to that matrix. It then applies the randomly generated matrix to the point set received and saves that matrix and the point set resultant of its application. This makes it so immutability examples are never created because the chances of all the 18 random generated values being 0 are practically nonexistent.

In order to try and improve the results of this algorithm when it comes to detection of immutability, we introduced a chance of a training or testing dataset entry to contain a case of immutability. In order to evaluate the impact that the presence of immutability in the training and testing datasets could have for this algorithm, we created models with 0%, 10% ad 50% cases of immutability in their respective training and testing datasets, as well as 10 tests, each one representing a case of immutability. We then calculated the Root Mean Squared Distance (RMSD) of each model in the 10 tests. The results of this experiment are presented in table 5.3.

Table 5.3: Results of immutability detection by PR-NET (measured by the RMSD).

% of immutability	1	2	3	4	5	6	7	8	9	10	average
50%	0.67	0.98	0.67	0.25	0.55	1.08	0.73	1.05	0.23	0.88	0.71
10%	0.64	1.46	0.64	0.25	0.35	1.17	0.93	1.14	0.22	1.35	0.82
0%	0.91	1.14	0.91	0.36	0.38	1.44	1.12	1.35	0.32	1.06	0.90

The results show that, even though increasing the amount of immutability in the training and testing datasets reduces the RMSD when detecting immutability, this reduction is not very significant and the values of the RMSD in the best case are still too high to be considered a good alignment of the point sets. We can then conclude that the inability of this algorithm to produce accurate alignments in the context of this framework is not due to the absence of immutability cases in its training and testing datasets.

5.3 Threshold Adjustments

The results presented in tables 5.1 and 5.2 are achieved using the same thresholds used by [6] when testing CPD. But, Carneiro *et al.*[6] had already concluded that small adjustments in the values of those thresholds can significantly change the results achieved by the framework. These small adjustments consist in reducing the value of the threshold by a small margin to ensure it detects a certain transformation. For example, when trying to detect a rotation of 50 degrees (test 2), reducing the value of absolute accumulative threshold of the rotation for this test from 50 to 49.5 allows GMMREG and KCREG to detect the rotation. Tables 5.4 and 5.5 introduce the results of GMMREG and KCREG when some very small adjustments like the one just mentioned are performed.

Table 5.4: Results of tests 1 to 6 before and after threshold adjustments.

	adjustment	1		2		3		4		5		6	
		M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
CPD		1	1	1	1	1	1	1	1	1	1	0.476	0.740
GMMREG	before	1	1	0.916	0	1	1	1	1	0.592	0.889	0.652	0.649
GMMREG	after	1	1	1	1	1	1	1	1	1	1	0.652	0.649
KC-REG	before	1	1	0.916	0	0.157	0	1	1	0.129	0.667	0.339	0.740
KC-REG	after	1	1	1	1	1	1	1	1	0.340	0.733	0.339	0.740

Table 5.5: Results of tests 7 to 11 before and after threshold adjustments.

	adjustment	7		8		9		10		11	
		M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
CPD		0.247	0	0.768	0.685	0.841	0.925	1	0.665	0.558	0.800
GMMREG	before	0.250	0.142	0.769	0.750	0.667	0.925	0.999	1	0.333	0.800
GMMREG	after	0.678	0.740	0.769	0.750	0.667	0.925	0.999	1	0.333	0.800
KC-REG	before	0.247	0	0.769	0.685	0.841	0.925	0.135	0.440	0.002	0.601
KC-REG	after	0.692	0.657	0.769	0.685	0.841	0.925	0.34	0.522	0.002	0.601

The results of PR-NET are not present in this tables because small adjustments in the thresholds of the tests do not improve the results of the algorithm.

After the adjustment of the thresholds, both GMMREG and KCREG become much more comparable to CPD in terms of results. From tests 1 to 5, the 3 algorithms obtain maximum value for every metric in every test (with the exception of KCREG in test 5). In tests 6 and 10, KCREG under performs when compared to CPD but GMMREG over performs. In test 7, both GMMREG and KCREG achieve better results than CPD and in tests 8 and 9, KCREG achieves similar results to CPD while GMMREG once again manages to improve. Test 11 becomes the only test where CPD can outperform the remaining algorithms. We can then conclude that KCREG is the most limited algorithm of the three and, on the other hand, GMMREG seems to be better than CPD in most scenarios.

5.4 Different Number of Vertices

The CPD algorithm assumes that the two point sets it is trying to align have the same number of points. But this might not be the case on a real scenario. It is then an advantage if an algorithm is able to align two point sets with different number of points.

We studied the three algorithms implemented (GMMREG, KCREG and PR-NET) on how they react to the alignment of two point sets with different number of points.

PR-NET cannot calculate the alignment of two point sets with different number of points since, like the CPD algorithm, its code assumes that the two point sets it receives share the same number of points.

GMMREG can calculate the alignment of two point sets with different number of points. In order to evaluate the ability of this algorithm to perform in this situation, we repeated the 11 tests performed before, but we removed one point from one of the point sets. Figure 5.3 identifies the point removed.

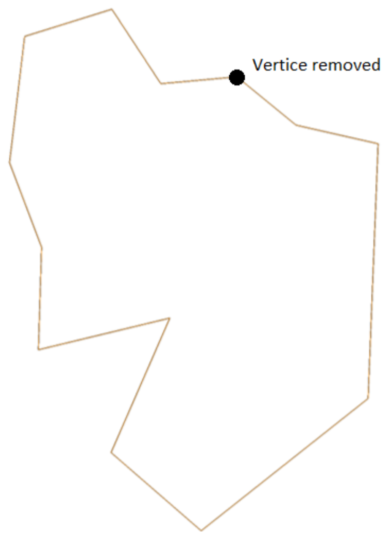


Figure 5.3: Point removed from the point set.

This point was chosen because it has a small effect on the shape of the polygon formed by the points of the point set when removed.

Tables 5.8 and 5.9 compare the results achieved by the GMMREG (with the threshold adjustments previously mentioned) when a point is missing in one of the point sets with the results it achieved when both point sets were complete.

Table 5.6: Results of tests 1 to 6 with the removal of one vertice.

	vertices removed	1		2		3		4		5		6	
		M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
GMMREG	0	1	1	1	1	1	1	1	1	1	1	0.652	0.649
GMMREG	1	0.614	1	0.484	0.467	0.352	0.467	1	0	0.001	0.312	0.001	0.315

Table 5.7: Results of tests 7 to 11 with the removal of one vertice.

	vertices removed	7		8		9		10		11	
		M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
GMMREG	0	0.678	0.740	0.769	0.750	0.667	0.925	0.999	1	0.333	0.800
GMMREG	1	0.197	0.732	0.171	0.662	0.216	0.902	0.002	0.324	0	0.534

These results show that, even though the GMMREG can calculate transformations to align two point sets with different number of points (unlike CPD and PR-NET), its results are very negatively impacted and it becomes unreliable

KCREG is also able to calculate the alignment of two point sets with different number of points, so we made the same experiment mentioned above to test if KCREG is more reliable than GMMREG in this situations. Figure 5.4 shows which points were removed.

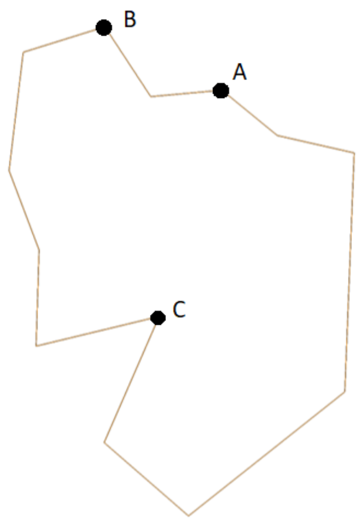


Figure 5.4: Points removed from the point set.

Three experiments were performed: in the first one point A was removed; in the second experiment both points A and B were removed; lastly, on he third experiment, points A, B and C were removed.

Tables 5.8 and 5.9 report the results of this experiment, also achieved with the threshold adjustments previously mentioned.

Table 5.8: Results of tests 1 to 5 with the removal of some vertices.

	vertices removed	1		2		3		4		5		6	
		M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
KC-REG	0	1	1	1	1	1	1	1	1	0.340	0.733	0.339	0.740
KC-REG	1	1	1	1	1	1	1	1	1	0.444	0.752	0.444	0.752
KC-REG	2	1	1	1	1	1	1	1	1	0.392	0.575	0.443	0.696
KC-REG	3	0.982	1	1	1	1	1	1	1	0.397	0.730	0.397	0.619

Table 5.9: Results of tests 6 to 11 with the removal of some vertices.

	vertices removed	7		8		9		10		11	
		M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
KC-REG	0	0.692	0.657	0.769	0.685	0.841	0.925	0.34	0.522	0.002	0.601
KC-REG	1	0.702	0.662	0.769	0.685	0.841	0.925	0.498	0.449	0	0.538
KC-REG	2	0.692	0.657	0.770	0.685	0.841	0.925	0.448	0.439	0	0.593
KC-REG	3	0.691	0.657	0.768	0.685	0.841	0.925	0.393	0.520	0.012	0.594

According to these results, KCREG is reliable at dealing with missing points, as the removal of up to three points in this case barely had any impact in the results of the algorithm.

5.5 Summary

This chapter presented the results extracted from the testing phase and the respective analysis. Section 5.1 compared the algorithms on the 11 tests performed and made the first analysis of those results. It was concluded that PR-NET greatly under performs in every single test, KC-REG achieves similar results to the CPD algorithm on about half the tests but under performs on the remaining tests and GMMREG is the only algorithm that manages to achieve results comparable to the CPD algorithm, but still slightly below it. Section 5.2 analysed PR-NET with more detail, registering the attempts to improve its results, namely the introduction of immutability in the training and testing datasets of the model, which didn't bring considerable improvements to the algorithm's results. Section 5.3 explored how small adjustments to the thresholds used in the 11 tests improve the results of the KC-REG and the GMMREG algorithms. This adjustments saw the KC-REG algorithm being able to achieve results much more comparable with the CPD algorithm, even besting it in some tests. The GMMREG algorithm also improved its results to values at the same level of the CPD algorithm. Finally, section 5.4 evaluated the performance of the algorithms when aligning two point sets with different number of vertices. It was concluded that, unlike CPD, both GMMREG and KC-REG are able two align point sets with different number of points. For GMMREG, the removal of a single point is enough to introduce a considerable amount of error in the results of the algorithm. On the other hand, KC-REG manages to align point sets with up to 3 points removed with zero to none error introduced by this factor.

Chapter 6

Conclusions

This chapter finishes this dissertation by drawing the main conclusion of the work as a hole in section 6.1 and identifying possible future work in section 6.2.

6.1 Conclusions

As spatiotemporal data becomes increasingly more available, the need to have tools able to automatically extract information about spatiotemporal events also increases. Considering the lack of tools designed for such tasks, a conceptual framework with the goal of automatically detecting spatiotemporal features based on raw spatiotemporal data and on user-defined goals was previously proposed. This framework takes 2D shape descriptors that define the contours of a given object and calculates the transformations that contour suffers throughout a sequence of snapshots using a Point Set Registration (PSR) algorithm. It then filters these transformations to extract the events of interest based on user-defined thresholds.

Point Set Registration is a wide family of algorithms which serve the same purpose but through very different methods, with different PSR algorithms being more or less limited then other alternatives. Since only one PSR algorithm was implemented in the original version of this framework, questions arose whether the results of the framework could be improved by using a different PSR algorithm, because the algorithm plays a critical role in detecting transformations as a by-product of the alignment goal.

The main goal of this work was then to review and study PSR algorithms and select the most promising ones to be integrated into the framework solution prototype and tested to better understand the impact different PSR algorithms have in results.

The performed literature review, presented on chapter 2, allowed us to define Point Set Registration and learn its applications. We also learned the general pipeline of a PSR algorithm and the different categories this algorithms can fall into, based on their characteristics. More specifically, these algorithms can be pairwise/groupwise, parametric/non-parametric and rigid/non-rigid, with some of these categories having sub-categories of their own. It was also noticed the existence of more recent deep learning based algorithms. We also explored the types of experiments, datasets and metrics that are more commonly used to test these algorithms. In the end of the chapter, we surveyed several PSR algorithms from all categories to find the ones better suited for our purpose.

Based on that survey, three algorithms were chosen to be implemented and tested in the framework: GMMREG, KCREG and PR-NET.

The algorithms were implemented in the framework prototype by creating a service module for each one of them. This architectural decision was based on the service module already existent in the framework, containing the original PSR algorithm, CPD.

With the implementation done, a testing protocol was created and put into practice, where we took advantage of the 11 tests already present in the framework, as well as the 2 metrics created by the original creator of the framework, to compare the 3 algorithms between themselves and with the CPD algorithm. A testing procedure was designed and repeated for the 3 algorithms, where the results achieved by each algorithm on the 2 metrics were registered for each test.

The results showed that the GMMREG algorithm was the only integrated algorithm able to achieve results comparable to the CPD algorithm, although still slightly below the CPD algorithm in general. The KC-REG algorithm managed to achieve results comparable to the CPD algorithm in only half the tests, and the PR-NET algorithm greatly under performed to the point that in every single test the value of the metric M1 was 0 or close to 0.

After analysing the PR-NET algorithm, it was concluded that this happened because the PR-NET model we created simply could not successfully align two point sets, as it would always introduce great errors in the alignment of some of the points of the point sets, to the point that the model was not even able to align two equal point sets (when immutability occurs). It was hypothesized that this happens because there were no cases of immutability in the training and testing datasets used to create the model but, after introducing immutability in those datasets, the results of the algorithm barely improved, leading us to conclude that the inability of this algorithm to produce accurate alignments is not due to the lack of immutability in its training and testing datasets.

Carneiro *et al.*[6] had already concluded that small adjustments in the values of the thresholds can significantly change the results achieved by the framework. With this is mind, very small adjustments were made to the thresholds used for the 11 tests, and those tests were repeated with the new thresholds for the GMMREG algorithm and for the KCREG algorithm. And, in fact, these adjustments led to the improvement of the results of both algorithms. Although the KCREG still remained below CPD, GMMREG was able to achieve results of the same level as the CPD algorithm.

Lastly, the ability of the algorithms to align point sets with different number of points was tested. CPD is not able to calculate the alignment in this context but both GMMREG and KCREG are. The results of the GMMREG algorithm lose a lot of accuracy when a single point is removed from one of the point sets it is trying to align but, on the other hand, the KCREG algorithm can align point sets with up to 3 points removed (out of 14) with barely any decrease in accuracy.

We can then conclude that, overall, CPD still seems to be the best algorithm to be used by this framework. The only weakness it presents is the inability to align point sets with different number of points, which might prove to be a big limitation when working with real datasets. In this context, KCREG proved to be the best algorithm by a big margin, which means that maybe it can be used to complement or substitute CPD in particular cases when the two point sets have different number of points and the CPD algorithm cannot be used.

In chapter 3, we proposed 3 research questions that we can answer now:

RQ1: Why do the results depend on the PSR technique used?

The results of the framework depend on the PSR algorithm used because, firstly, it is the PSR algorithm that calculates the transformations for the framework to filter based on the user-inputted thresholds. This means that the better the calculation of the transformations the more accurate the events of interest generated by the framework will be.

RQ2: How much do the results depend on the PSR technique used?

As seen in the results presented in chapter 5, for the same test, the framework can achieve both maximum and minimum values for the 2 metrics used to evaluate the results, depending on which algorithm is performing the calculation of the transformations so, it is safe to conclude that the results of the framework hugely depend on the PSR algorithm used.

RQ3: How do the Deep-Learning based PSR techniques compare to the traditional PSR techniques in the context of this framework?

Based on the results presented in chapter 5, we could say that Deep-Learning based PSR algorithms greatly under perform compared to traditional PSR algorithm in the context of this framework. But, we think that the work performed in this dissertation is not enough to safely reach that conclusion. Firstly, only one Deep-Learning based PSR algorithm was tested and we cannot assume that every other Deep-Learning based PSR algorithm would achieve similar results. Also, the training of the models based on this algorithm might not have been optimized for this context, negatively effecting its results.

The code developed for this dissertation is in the following web address: `https://gitlab.inesctec.pt/avc/mit-ees-datalab-2023/-/tree/code/`

6.2 Future Work

In order to continue the study started by this dissertation, there are some paths that can be taken:

- creating new tests with real datasets or more irregular synthetic datasets to better evaluate the ability of the framework to perform with real data;
- integrate at least a second Deep-Learning bases PSR algorithm into the framework to draw better conclusions about the performance of this type of PSR algorithms in the context of this framework.

References

[1] Sk Aziz Ali, Vladislav Golyanik, and Didier Stricker. Nrga: Gravitational approach for non-rigid point set registration. pages 756–765. Institute of Electrical and Electronics Engineers Inc., 10 2018.

[2] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data.

[3] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust efficient point cloud registration using pointnet, 2019.

[4] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 4 2002.

[5] Hamid Reza Boveiri, Raouf Khayami, Reza Javidan, and Alireza Mehdizadeh. Medical image registration using deep neural networks: A comprehensive review. *Computers and Electrical Engineering*, 87, 10 2020.

[6] Edgar Filipe Amorim Gomes Carneiro, Alexandre Miguel Barbosa Valle de Carvalho, and Rui Pedro Amaral Rodrigues. Representation and quantification of change on spatiotemporal phenomena, 2020.

[7] Hongchen Chen, Xie Zhang, Shaoyi Du, Zongze Wu, and Nanning Zheng. A correntropy-based affine iterative closest point algorithm for robust point set registration. *IEEE/CAA Journal of Automatica Sinica*, 6:981–991, 7 2019.

[8] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. pages 145–155, 10 1992.

[9] Yan Deng, Anand Rangarajan, Stephan Eisenschenk, and Baba C. Vemuri. A riemannian framework for matching point clouds represented by the schrodinger distance transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[10] Stanley Durrleman, Xavier Pennec, Alain Trouvé, and Nicholas Ayache. Statistical models of sets of curves and surfaces based on currents. *Medical Image Analysis*, 13:793–808, 10 2009.

[11] Georgios Dimitrios Evangelidis and Radu Horaud. Joint alignment of multiple point sets with batch and incremental expectation-maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:1397–1410, 6 2018.

[12] J D Foley, Martin A Fischler, and Robert C Bolles. Graphics and image processing random sample consensus: A paradigm for model fitting with apphcatlons to image analysis and automated cartography, 1981.

[13] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[14] Vladislav Golyanik, Sk Aziz Ali, and Didier Stricker. Gravitational approach for point set registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[15] Ronald L Graham and F Frances Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4, 1983.

[16] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds, 2019.

[17] Erion Hasanbelliu, Luis Sanchez Giraldo, and Jose C. Principe. Information theoretic shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:2436–2451, 12 2014.

[18] Osamu Hirose. Acceleration of non-rigid point set registration with downsampling and gaussian process regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:2858–2865, 8 2021.

[19] Osamu Hirose. A bayesian formulation of coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:2269–2286, 7 2021.

[20] Daniel F. Huber and Martial Hebert. Fully automatic registration of multiple 3d data sets. volume 21, pages 637–650. Elsevier Ltd, 7 2003.

[21] Bing Jian and Baba C. Vemuri. Robust point set registration using gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1633–1645, 2011.

[22] L.J. Latecki, R. Lakamper, and T. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 1, pages 424–429 vol.1, 2000.

[23] Liang Li, Ming Yang, Chunxiang Wang, and Bing Wang. Rigid point set registration based on cubature kalman filter and its application in intelligent vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 19:1754–1765, 6 2018.

[24] Luming Liang, Mingqiang Wei, Andrzej Szymczak, Anthony Petrella, Haoran Xie, Jing Qin, Jun Wang, and Fu Lee Wang. Nonrigid iterative closest points for registration of 3d biomedical surfaces. *Optics and Lasers in Engineering*, 100:141–154, 1 2018.

[25] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds, 2019.

[26] Bruce D Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, Vancouver, Canada, August 1981.

[27] Jiayi Ma, Huabing Zhou, Ji Zhao, Yuan Gao, Junjun Jiang, and Jinwen Tian. Robust feature matching for remote sensing image registration via locally linear transforming. *IEEE Transactions on Geoscience and Remote Sensing*, 53:6469–6481, 12 2015.

[28] Baraka Maiseli, Yanfeng Gu, and Huijun Gao. Recent developments and trends in point set registration methods, 7 2017.

[29] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[30] M. Menze, C. Heipke, and A. Geiger. Joint 3d estimation of vehicles and scene flow. volume 2, pages 427–434. Copernicus GmbH, 8 2015.

[31] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drifts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:2262–2275, 2010.

[32] Toby A. Patterson, Len Thomas, Chris Wilcox, Otso Ovaskainen, and Jason Matthiopoulos. State-space models of individual animal movement, 2 2008.

[33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[34] Nishant Ravikumar, Ali Gooya, Serkan Çimen, Alejandro F. Frangi, and Zeike A. Taylor. Group-wise similarity registration of point sets using student's t-mixture model for statistical shape models. *Medical Image Analysis*, 44:156–176, 2 2018.

[35] L. Romero and F Calderón. Scene reconstruction pose estimation and tracking. 2007.

[36] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM*, pages 145–152, 2001.

[37] Vinit Sarode, Xueqian Li, Hunter Goforth, Yasuhiro Aoki, Animesh Dhagat, Rangaprasad Arun Srivatsan, Simon Lucey, and Howie Choset. One framework to register them all: Pointnet encoding for point cloud alignment. 12 2019.

[38] Kanade T. Tsin, Y. A correlation-based approach to robust point set registration. 2004.

[39] Nazlı Tümer, Aimee C. Kok, Frans M. Vos, Geert J. Streekstra, Christian Askeland, Gabrielle J.M. Tuijthof, and Amir A. Zadpoor. Three-dimensional registration of freehand-tracked ultrasound to ct images of the talocrural joint. *Sensors (Switzerland)*, 18, 7 2018.

[40] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? page 1073–1080, 6 2009.

[41] Chen J. Li X. Fang Y Wang, L. on-rigid point set registration networks. 2019.

[42] Senzhang Wang, Jiannong Cao, and Philip S. Yu. Deep learning for spatio-temporal data mining: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34:3681–3700, 8 2022.

[43] Renliang Weng, Jiwen Lu, and Yap Peng Tan. Robust point set matching for partial face recognition. *IEEE Transactions on Image Processing*, 25:1163–1176, 3 2016.

[44] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[45] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-icp: A globally optimal solution to 3d icp point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:2241–2254, 11 2016.

[46] Lijun Yin, Xiaozhou Wei, Yi Sun, Jun Wang, and M.J. Rosato. A 3d facial expression database for facial behavior research. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 211–216, 2006.

[47] Manzhu Yu, Myra Bambacus, Guido Cervone, Keith Clarke, Daniel Duffy, Qunying Huang, Jing Li, Wenwen Li, Zhenlong Li, Qian Liu, Bernd Resch, Jingchao Yang, and Chaowei Yang. Spatiotemporal event detection: a review, 2020.

[48] Ang Zhang, Zhe Min, Zhengyan Zhang, Xing Yang, and Max Q.H. Meng. Anisotropic generalized bayesian coherent point drift for point set registration. *IEEE Transactions on Automation Science and Engineering*, 2022.

[49] Lei Zhang, Sung In Choi, and Soon Yong Park. Robust icp registration using biunique correspondence. pages 80–85, 2011.

[50] Zhiyuan Zhang, Yuchao Dai, and Jiadai Sun. Deep learning based point cloud registration: an overview, 6 2020.

[51] Hao Zhu, Bin Guo, Ke Zou, Yongfu Li, Ka Veng Yuen, Lyudmila Mihaylova, and Henry Leung. A review of point set registration: From pairwise registration to groupwise registration, 3 2019.