Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Master Thesis

Deep Learning-based 4D Point Cloud Analysis to Characterize Tissue Morphogenesis

Petr Ježek

Supervisor: Prof. Dr.-Ing. Johannes Stegmaier
Second supervisor: Ing. Vojtěch Vonásek, Ph.D.
May 2024

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Ježek Petr**	Personal ID number:	**483566**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

Deep Learning-based 4D Point Cloud Analysis to Characterize Tissue Morphogenesis

Master’s thesis title in Czech:

Analýza 4D point cloudů k charakteristice morfogeneze tkání pomocí hlubokého učení

Guidelines:

1. Conduct literature research to identify suitable network architectures capable of predicting spatio-temporal movement patterns within small spatio-temporal neighborhoods.
2. Create a synthetic training dataset that mimics realistic movement patterns based on existing biological knowledge about early embryonic development.
3. Explore extensions or adaptations of existing models like PointNet and PointNet++ to handle time-resolved 3D data, potentially by incorporating temporal information or movement vectors.
4. Consider the use of graph neural networks to model both spatial and temporal connectivity among tracklets.
5. Develop a deep learning-based approach to analyze characteristic movement patterns in the early development of multi-cellular organisms, particularly focusing on the early embryo.
6. Train the envisioned network architectures to predict the presence and degree of predefined movement patterns or combinations of multiple movement patterns.
7. Focus on movement patterns such as convergent extension movements, vortex-like rotation movements, cell intercalations, and tissue invagination movements, commonly observed during the gastrulation phase of early embryonic development.
8. Apply the trained networks to real 3D+t cell tracking data from zebrafish embryos and visualize detected movement patterns in the early stages of development.

Bibliography / sources:

[1] Keller, P. J., Schmidt, A. D., Wittbrodt, J. & Stelzer, E. H. K. Reconstruction of Zebrafish Early Embryonic Development by Scanned Light Sheet Microscopy. Science 322, 1065–1069 (2008).
[2] Shindo, A. Models of convergent extension during morphogenesis. Wiley Interdiscip Rev Dev Biology 7, e293 (2018).
[3] Bensch, R., Song, S., Ronneberger, O. & Driever, W. Non-directional radial intercalation dominates deep cell behavior during zebrafish epiboly. Biol Open 2, 845–854 (2013).
[4] Traub, M. & Stegmaier, J. Towards Automatic Embryo Staging in 3D+T Microscopy Images using Convolutional Neural Networks and PointNets. Arxiv (2019).
[5] Sauder, J., & Sievers, B. (2019). Self-supervised deep learning on point clouds by reconstructing space. Advances in Neural Information Processing Systems, 32.
[6] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., & Bennamoun, M. (2020). Deep learning for 3d point clouds: A survey. IEEE transactions on pattern analysis and machine intelligence, 43(12), 4338-4364.
[7] Rosu, R. A., Schütt, P., Quenzel, J., & Behnke, S. (2022). Latticenet: fast spatio-temporal point cloud segmentation using permutohedral lattices. Autonomous Robots, 46(1), 45-60.
[8] Shi, H., Lin, G., Wang, H., Hung, T. Y., & Wang, Z. (2020). Spsequencenet: Semantic segmentation network on 4d point clouds. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 4574-4583).

Name and workplace of master’s thesis supervisor:

Prof. Dr.-Ing. Johannes Stegmaier RWTH Aachen University, Germany

Name and workplace of second master’s thesis supervisor or consultant:

Ing. Vojtěch Vonásek, Ph.D. Multi-robot Systems FEE

Date of master’s thesis assignment: **12.09.2023** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **16.02.2025**

Prof. Dr.-Ing. Johannes Stegmaier	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I thank Prof. Stegmaier for supervising this thesis, for valuable guidance, and for continuous encouragement. I also thank to the Institute of Imaging and Computer Vision for providing computational resources and great support for my thesis. Most importantly, I would like to thank my family and my fiancée for supporting me and encouraging me in my professional career and personal life.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 24.5.2024

....
Petr Ježek

Abstract

The dynamic processes of cells in the embryo are important for the successful development of the organism. Studying them helps uncover information about tissue formation, patological states, or mutant patterns. In recent years, advancements in fast miscroscopy imaging have made it possible to study these cell movements *in vivo*. However, due to the large amount of data, manual analysis of them becomes impractical.

This thesis aims to develop a deep learning-based approach to classify movement patterns in the early stages of zebrafish development. Since real point clouds are not labeled, synthetic cell movements were simulated to create training and testing datasets for the proposed architectures.

Two point cloud processing networks were evaluated on the point-wise classification task of 3D+t point clouds and were successful for the simulated dataset featuring synthesized movements. Subsequently, both networks were applied on the same task for real embryos, yielding very promising results that could be compared with the embryogenetic events observed in real embryos. Furthermore, many interesting observations were made, confirming the networks' ability to accurately classify time-resolved 3D point clouds.

Keywords: Embryo morphogenesis, Zebrafish, 3D+t Point clouds, Movement pattern recognition, Graph neural networks, Recurrent neural networks

Abstrakt

Dynamické procesy buněk v embryu jsou důležité pro úspěšný vývoj organismu. Jejich studium pomáhá odhalit informace o tvorbě tkání, patologických stavech nebo mutacích. V posledních letech umožnil pokrok v rychlém mikroskopickém zobrazování studovat tyto pohyby buněk *in vivo*. Vzhledem k velkému množství dat se však jejich manuální analýza stává nepraktickou.

Cílem této práce je vyvinout přístup založený na hlubokém učení pro klasifikaci pohybových vzorců v raných fázích vývoje ryb, konkrétně Danio rerio. Protože skutečná mračna bodů nejsou anotovány, byly simulovány syntetické pohyby buněk, aby se vytvořily trénovací a testovací data pro navržené architektury.

Dvě neuronové sítě pro zpracování mračen bodů byly vyhodnoceny na úloze bodové klasifikace 3D+t mračen bodů a byly úspěšné pro simulovaná data obsahující syntetické pohyby. Následně byly obě sítě použity na stejnou úlohu pro skutečná embrya a přinesly velmi slibné výsledky, které bylo možné porovnat s embryogenetickými ději pozorovanými u skutečných embryí.

Kromě toho bylo učiněno mnoho zajímavých pozorování, která potvrdila schopnost neuronových sítí přesně klasifikovat časově závislá 3D mračna bodů.

Klíčová slova: Morfogeneze embrya, Danio rerio, 3D+t Mraky bodů, Rozpoznávání pohybových vzorů, Grafové neuronové sítě, Rekurentní neuronové sítě

Contents

Figures

Tables

Chapter 1

Introduction

The embryonal morphogenetic processes are events in embryo development that comprise cell interactions and cell movements [1]. These dynamic rearrangements are crucial in the successful formation of the developing organism, since they form and modify its tissues [2, 3]. Studying cell movements and cell migration can help uncover more information about the stage of development [1, 4], how cell movements and interactions influence cell fate [5], determine mutant behavior during morphogenesis [6], recognize abnormal migration of cells that form tissues that can cause patological states in the adult organism [4] or how to temporarily synchronize two distinct embryonic processes.

The major event during the morphogenesis of an animal embryo is gastrulation. In this phase, large-scale movements and reorganizations of cell groups can be observed [7]. These movements rearrange cells, so that from the uniform two-dimensional sheet of cells, multiple layers of cells with different identities are arranged in three dimensions [8]. The cells are arranged to later form three germ layers: ectoderm, mesoderm, and endoderm [9]. Each of these layers has its own role in the subsequent development processes of the embryo [10].

Fast microscopy imagining of *in vivo* embryos, such as [11], made it possible to accurately segment the cells [12, 13] and track them over time [11, 12, 14, 15] to create a digital embryo. A digital embryo is a 3D time-resolved point cloud that captures the natural embryo with single-cell precision and provides additional information on the positions and movements of cells (points) over time.

Cell movement characterization and understanding turns increasing attention in developmental biology, which studies embryo development. Using either classical methods [16, 17] or not yet examined deep learning methods. In this thesis, the state-of-the-art point cloud processing networks are carefully reviewed, and later two of the promising architectures, namely PointLSTM [18] and DGCNN [19, 20] are implemented for the task of cell movement recognition. Due to the fact that real point cloud data are unlabeled, since cell movement are very complex, it is necessary to first learn models on synthetic data and later test on real ones. The results can be visually compared to expected movement patterns based on prior knowledge from developmental biology research, such as [7].

The thesis is organized as follows: Firstly, the theoretical background of developmental biology is described (see Sec. 2.1 of Chapter 2). The rest of Chapter 2 contains a detailed review of classical methods for point cloud processing in embryogenesis (see Sec. 2.2), followed by state-of-the-art of deep learning on 3D point clouds (see Sec. 2.3) and 3D+t point clouds (see Sec. 2.4). In the later mentioned section, two implemented networks are briefly introduced and later in Chapter 3 are described in further detail. But firstly, the information about the real dataset as well as the synthetic dataset and movement generation is given in the first six sections of the Chapter 3. The network architectures are described in Sec. 3.6 as well as their use for cell movement classification (see Sec. 3.7). In Chapter 4 the networks are tested on the task of cell movement classification. First, several ablation studies are conducted on the synthesized point clouds to find the optimal parameters of the networks. Later there are several visualizations on artificial point clouds showing how well the network performs visually. In the end, tests on the real embryo point cloud are executed to find out if the network can correctly classify movement patterns of real spatio-temporal groups of cells. Chapter 5 provides a conclusion of the thesis with a short summary and an outlook on future development.

Chapter 2

Theoretical Background and Related Work

This chapter provides a brief introduction to zebrafish embryogenesis followed by methods for movement recognition. First, classical methods for embryo dynamics recognition are described and some related fields, such as gesture recognition, are explained as well. After the classical methods, there is a detailed description of deep learning methods for 3D point cloud analysis. This section is further extended for 3D+t point clouds and methods later used in the Methods and Implementation in Chapter 3.

2.1 Embryogenesis

During early embryo development, all animals undergo dramatic changes in shape [8]. In this work, the focus is on the early stages of development in one of the most studied models of fish called zebrafish (*Danio rerio*). The reason for its wide use in the field of developmental biology is the small size of adult fish, so it can hatch in thousands in a small space [21]. After two days, the embryo is almost finished with organ development, which helps to speed up research on fish development [22].

The embryo develops outside of the mother and is transparent, allowing the use of microscopic imaging techniques, such as digital scanned laser light sheet fluorescence microscopy (DSLM) [11] to capture all cells in the whole embryo with high speed in real time *in vivo* (embryo is alive during acquisition). With acquired data, it is possible to create cell lineages. The cell lineage is a tracked trajectory of a cell from the time it is divided from the mother cell to the time of apoptosis (programmed cell death). During image acquisition, it also happens that a cell can move out of the field of view or the fluorescence reporter could be bleached. These events mean that lineages on the acquised image end earlier than during apoptosis or at the end of experiment. These cell lineage fragments are called tracklets.

Tracking algorithms such as their own pipeline [11] or [12, 15] generate cell lineages for each cell. Moreover, from single cell resolution, it is even possible to extract detailed cell shape by cell segmentation algorithm such as [12, 13]. Determining cell shapes is crucial for understanding cell morphology and cell differentiation [1].

Moreover, by micro-injection of fluorescent dyes, it is possible to create a fate map of cells. The cell fate map is generated from cell lineages to the adult state of the examined embryo, where cells can be assigned to a tissue / organ in which they form in adult fish (more details in Sec. 2.1.2). Zebrafish was also the first vertebrate to be subjected to mutagenesis studies [22]. By injecting nucleic acids into individual blastomeres (cell type in the early embryo stage [10]) one can manipulate fish genes and therefore investigate the impact of various mutations [22].

The stages of development of healthy individuals follow a fixed schedule, which can be seen

[1]cell transitioning from one type to another

in Fig. 2.1. This thesis focusses on stages where dramatic changes in embryo shape take place and groups of cells move deliberately. The most important changes occur in late cleavage and gastrulation [8]. This morphogenetic development spans from 4-5 to 10 hours post fertilisation (hpf) [7, 23].

2.1.1 Fertilization and Cleavage

Embryo development begins with egg fertilization, where sperm and egg fuse and form a zygote. The entry of sperm into the egg marks the ventral side, the opposite side is dorsal [22] (marked in Fig. 2.4). In the next stage, called cleavage, cell divisions occur at the animal pole of the egg yolk (Fig. 2.2). Note that in these stages, the embryo resembles a sphere. The embryo can have an arbitrary orientation in the chorion [24] (the outermost membrane around the embryo), but during preprocessing, the data were rotated so that the animal pole is the pole of the embryo in the northern hemisphere and the vegetal pole is in the southern hemisphere. This helps align different samples with each other and identify common characteristics.

The first several divisions synchronously divide cells located at the animal pole above the yolk cell [7]. Further divisions lead to the formation of a blastoderm at the animal pole with about 1000 cells [22]. In this sphere stage there are three distinct layers of cells (first box in Fig. 2.3). The outer enveloping layer (EVL) with flattened cells, which is the most superficial layer of the blastoderm, consists of only a single layer of cells. EVL serves as a protective cover for the embryo for the first two weeks of development, then it is lost [22]. Yolk syncytial layer (YSL) on the border of egg yolk and blastoderm is created in the tenth division cycle and is formed by cells located in the yolk cytoplasm right under the blastoderm [26]. The deep internal layer is located between the EVL and YSL with more rounded cells. The deep cells then give rise to the embryonic tissues itself [2]. Since there is too much cell mixing during the cleavage stage, it can not yet be determined by cell lineage studies which cell gives rise to which organ of the developed fish, the cell fate map (more on that in Sect. 2.1.2) [27].

Later, those form into the germ layers, ectoderm from cells near animal pole (epiblast), mesoderm from deep cells in the middle and bottom, and endoderm from the bottom ones (hypoblast) [7]. The mesendoderm is a group of cells at the bottom of the blastoderm (blastoderm margin) that can have its fate either in the endoderm or in the mesoderm (see the first box in Fig. 2.3) [8]. However, cells more than 4 cell diameters away from the blastoderm margin develop exclusively into the mesoderm [28].

At later cleavage, the blastoderm starts to spread radially toward the vegetal pole to completely engulf the yolk cell [29]. This spread movement is called epiboly. The percentage of epiboly is defined as how much of the yolk is already covered by spreading blastomere cells [7]. It is a commonly used method to classify the stage of embryo development, e.g., the blastoderm is in 30% of epiboly. The thinning of the blastoderm due to epiboly is called radial intercalation. It disperses deep cells in the deep layer, but not among cells in the enveloping layer [1]. Furthermore, the intercalations are non-directional, which means that the upward, downward, and lateral intercalations with respect to the enveloping layer occur at similar rates [30].

2.1.2 Gastrulation

At about 50% of epiboly, a new stage called gastrulation begins [1], where the embryo undergoes a general reorganization of the cells [7]. The deep cells of the embryo start to have an organ-specific fate, that is, that specific cells from the same region end up being the same organs as can be seen in Fig. 2.4 [22].

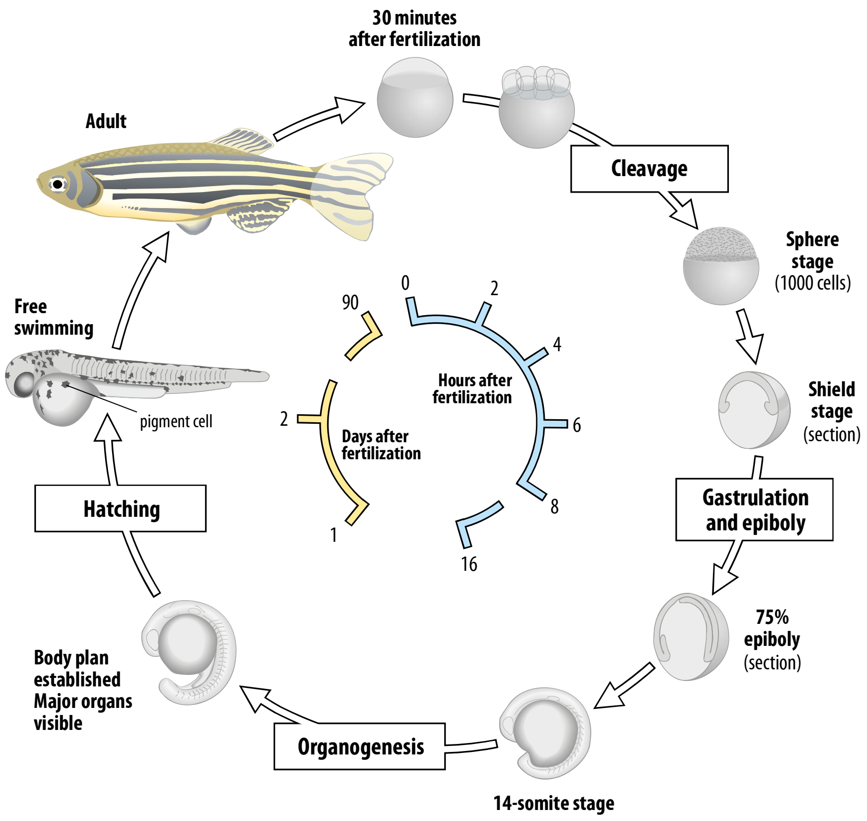


Figure 2.1: Stages of development of the zebrafish [8]

Deeper cells start to create a thickened layer at the edge of the blastoderm called a germ ring (second box in Fig. 2.3 and on real embryo in Fig. 2.9b). The outer layer of the germ ring is called the epiblast (later developing into the ectoderm [10]) and the inner layer the hypoblast (endoderm and mesoderm). Hypoblast cells first move to the center of the embryo and then under the ectoderm toward the animal pole, against the spread of the epiboly (third box in Fig. 2.3) [8]. The ectoderm proceeds with epiboly until it covers the entire embryo [29]. These movements combined together are called involution. In [1] they consider the involution to be the beginning of the gastrulation.

During this time, the hypoblast on the dorsal side thickens locally and forms an embryonic shield [9]. This shield is well visible in Fig. 2.9c. Due to the formation of the shield, this stage is called the shield stage. After shield formation, another movement pattern, known as convergence extension, begins. Convergence movements thicken the cell layer locally [31] on one side of the embryo, which later forms the body of the embryo (fourth box in Fig. 2.3 and Fig. 2.5). Cells intercalate between each other [32], therefore, at the same time, extension movement elongates the accumulated cells toward the animal pole along the antero-posterior direction[2] [31], precursor of the notochord [22]. This forming notochord of the developing animal is visible in Fig. 2.9d-e.

2.1.3 Cell Dynamics in Embryo Development

As already mentioned, gastrulation is a phase where major dynamic events happen during embryogenesis. These cell migration processes are crucial in the successful formation of the new

[2]anterior - head and posterior - tail of the developing fish

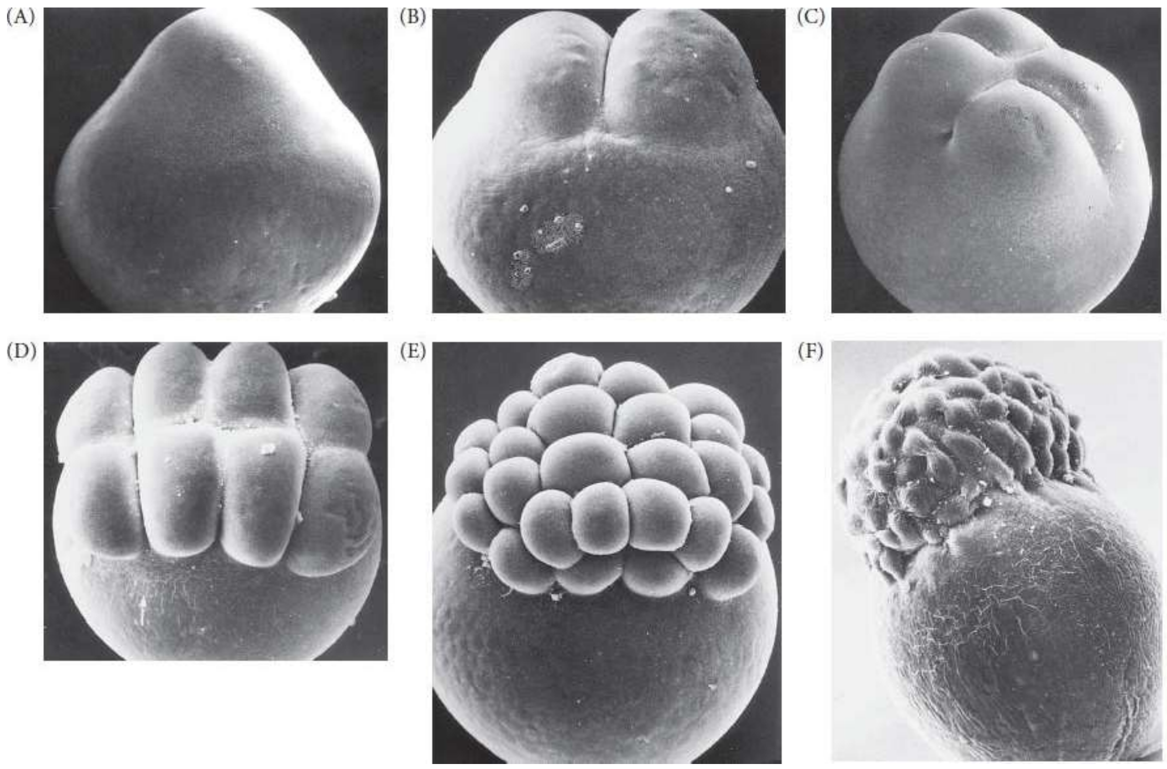


Figure 2.2: First five cleavages of the zygote [25]

organism [3]. The cell ability to travel for relatively long distances as individuals or in groups is called cell migration [8]. Based on cell rearrangements, it can be possible to temporally synchronise stages of development of distinct organisms [33, 34], since the development of different individuals is fundamentally similar, however, each phase of development can be slightly different in individual cell movement. Temporal alignment can enable extracting information about embryo development success, determine mutant behavior [4, 6].

2.2 Classical Methods for Characterization of Cell Movements

Due to rapid progress in the field of microscopy imaging, cell segmentation, and cell tracking, digital embryos can be easily created from individual specimens. Those embryos can be further analysed by means of visual inspection, feature extraction, or deep learning. However, since embryo tracking over time produces a large amount of data, it is mostly necessary to automate the process of embryo analysis. This section focusses on the classical feature extraction, such as deformation patterns, morphogenetic landmarks, and relative cell motion.

2.2.1 Kinematic Analysis of Deformation Patterns

One of the first classical method for analysing embryo morphogenesis [16] determines the deformation patterns and morphogenetic landmarks of embryo tracklets. Both instantaneous and cumulative tissue deformation rates were assessed. Using tensorial analysis, mechanical descriptors were computed to help quantify tissue compression, expansion, rotation, and distortion during the gastrulation phase of the embryo. From instantaneous deformation rates they were able to identify individual gastrulation phases. From cumulative rates,

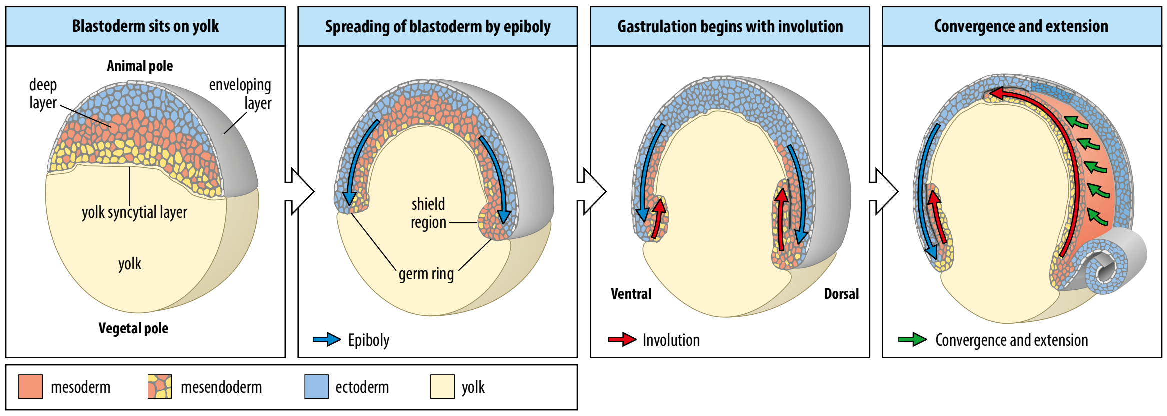


Figure 2.3: Late cleavage and gastrulation phase of zebrafish [8]

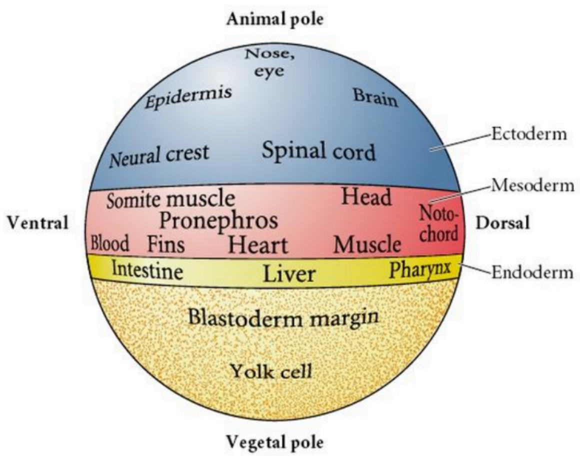


Figure 2.4: Fate map of deep cells at the beginning of gastrulation [22]

the Canonical Lagrangian Biomechanical Profiles (CLBPs) were defined. CLBPs define the biomechanical characteristics of each cell. These characteristics were used to build the Lagrangian Biomechanical Map (LBMap) containing information about mechanical deformations and their correlations with cell fate [16].

For each cell nucleus detected at position $\mathbf{x}_i$, they first computed its approximated displacement field $\mathbf{v}(\mathbf{x}_i)$. This field was smoothed temporarily with Gaussian kernel to obtain $\mathbf{v}_T(\mathbf{x}_i)$ and then spatially with the help of weighted nearest neighbors

$$\mathbf{v}(\mathbf{x}_i)_{TR} = \frac{1}{W_{i,R}} \sum_{j \in N_R(i)} w_{i,R}(\mathbf{x}_j)\mathbf{v}_T(\mathbf{x}_i), \tag{2.1}$$

where $N_R(i)$ are nearest neighbors of the i-th cell in the radius R, $w_{i,R}(\mathbf{x}_j)$ is weight of the neighbor j of cell i and $W_{i,R}$ is the sum of the weights for cell i of all neighbors. From the approximated differentiable velocity vector field, they could find the Incremental Deformation Gradient (IDG) tensor field $\mathbf{f}$ of displacements in the $2R$ region of each cell. This field describes the mapping between two consecutive displacements in time

$$d\mathbf{x}_i^{t+\Delta t} = \mathbf{f} d\mathbf{x}_i^t. \tag{2.2}$$

Next, they defined several descriptors computed as invariants of the IDGs as independent of the spatial coordinates, since they express the course of displacement. The instantaneous

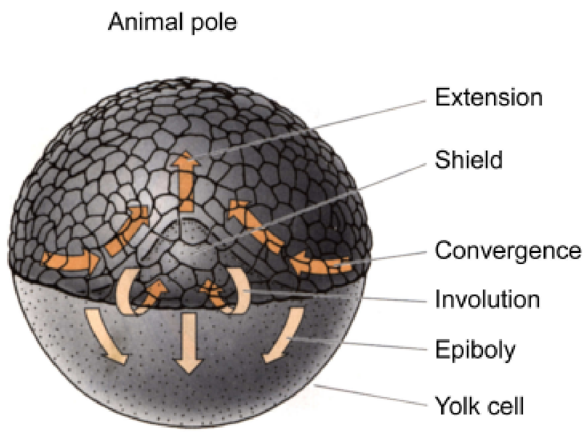


Figure 2.5: Convergent extension movement from the dorsal view [22]

descriptors P, Q_d, D, and τ correspond to compression/expansion, distortion rate, rotation, and a combination of compression/expansion and rotation (Fig. 2.6a).

Furthermore, from the obtained deformation gradients, they built CLBPs that correspond to cumulative biomechanical activity along each cell trajectory. By composing the IDG tensors $\mathbf{f}_i$ of one cell over a time interval $[t_{init}, t]$ using the chain rule to a Finite Time Deformation Gradient $\mathbf{F}_i^{[t_{init},t]}$

$$\mathbf{F}_i^{[t_{init},t]} = \mathbf{f}_i^{t-1}\mathbf{f}_i^{t-2}\ldots\mathbf{f}_i^{t_{init}+1}\mathbf{f}_i^{t_{init}}, \tag{2.3}$$

they could extract cumulative descriptors J, $MIC1/MIC2$ and θ as $\mathbf{F}$ tensor invariants that correspond to cumulative (finite time) compression/expansion, distortion and rotation (Fig. 2.6b). For each cell trajectory, they extracted a feature vector. These features were classified into four clusters using hierarchical clustering and the computed clusters were compared to the embryo fate map. It revealed that cells with similar biomechanical history end up having a similar fate [16]. Additionally, by visualizing individual descriptors, it is possible to detect local movements of cells or a combination of them, which served as a basis for selecting the set of possible classifiable movements in embryo tracklets (Fig. 2.7).

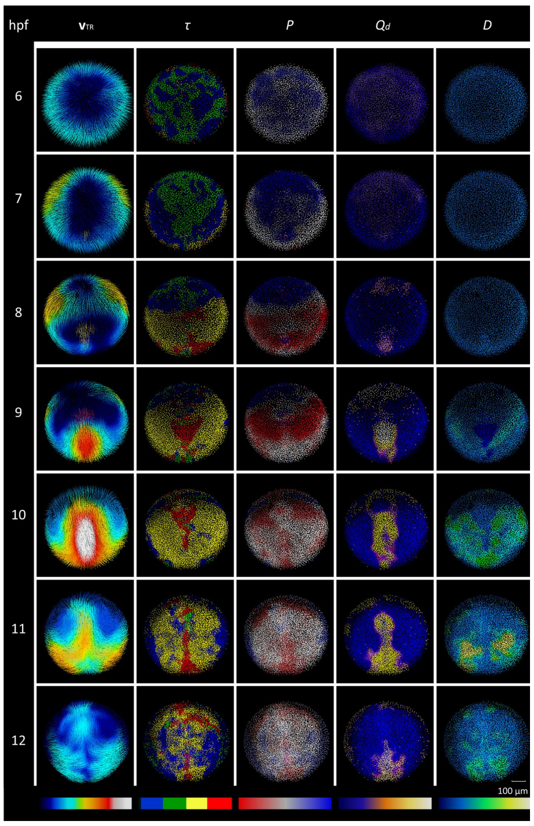
2.2.2 Analysis of Tissue Tectonics

In [17] they link shear rates and rotations to tissue morphogenesis using tensor analysis and continuum vector fields. They measured cell shapes, cell intercalations, tissue strain rates, and tissue rotation rates.

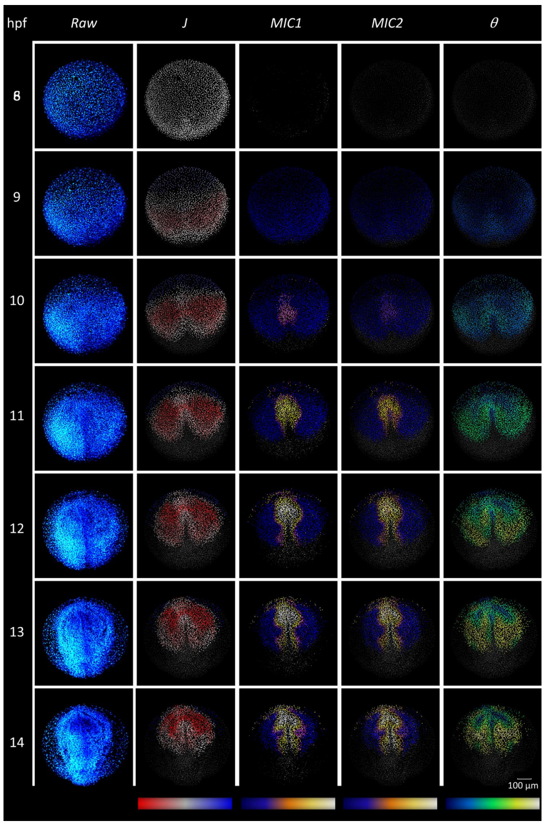
They showed their approach on a zebrafish tissue (Fig. 2.8b) describing the rotations and strain rates according to domain translations (Fig. 2.8a). By visualizing all the descriptors for all the cells in the embryo, they were able to derive conclusions about the overal changes of the cell shape, the gradient of the rotation rates, and the alignment of the cell intercalations.

2.2.3 Other Related Fields

There is another field that actively classifies movement patterns. Although it is remote from embryogenetics, it is a very similar task both in the type of input data and the expected result. Gesture recognition is very similar to cell movement classification, since both gestures and cell movements contain multiple temporally connected frames. Gestures are also recorded as point clouds by tracking markers on a gesturing human. Pioneering work [35] is to extract histogram of gradients (HOG) and classify it based on similarity to training data. Similarly



(a) instantaneous descriptors [16]



(b) cumulative descriptors [16]

Figure 2.6: Two types of descriptors extracted from the embryo point cloud. They show the development of descriptor values over time (in hpf). For example, from the velocity $\mathbf{v}_{TR}$ in (a) it is clearly visible the involution movement from 9 hpf. Shear Q_d also denotes involution (the hypoblast moves in the opposite direction as the epiblast-generating shear). In (b), there is mainly compression movement throughout the entire gastrulation phase (descriptor J) [16].

for sign language gesture classification [36] they used the ensemble of shape function (ESF) descriptor as extracted feature and then used the multilayered random forest method (RFM) for gesture classification.

2.3 Deep Learning on Point Clouds

In recent years, deep learning techniques have dominated many research areas, including robotics, computer vision, or natural language processing [37]. It is possible to train a deep neural network, which is superior in most applications to the classical methods described in Sec. 2.2 provided that they have sufficiently large datasets [38]. Examples of problems from other fields that can be addressed using deep learning methods include image segmentation, video annotation, and classifications with different data modalities.

Various research studies have also been done on biomedical data processing [39], such as cell segmentation [40, 41], cell tracking [42, 43], analysis of cell shape dynamics [44], or restoration and denoising of biomedical images [45]. In the realm of developmental biology, several studies have focused on automatic embryo staging [33, 46]. From these applications in embryo data analysis, researchers can understand genetic perturbations, pursuing drug discovery or cell phenotyping of the developing organism [38, 41]. Cell phenotyping refers to the conglomeration of cellular processes, which results in the morphology and function of

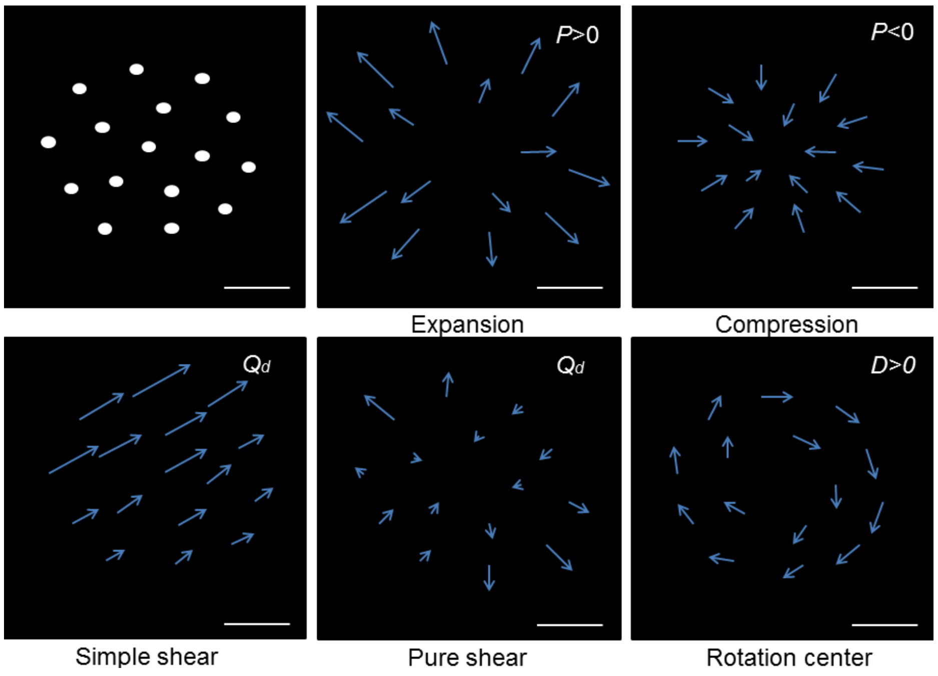


Figure 2.7: Detectable movements using descriptors from [16]

particular cells [47].

Embryonic data are mostly represented by point clouds, where each point corresponds to one cell. An example of embryo point cloud is shown in Fig. 2.9. The captured frames mark different stages of embryo development: late cleavage Fig. 2.9a, beginning of gastrulation Fig. 2.9b, shield stage Fig. 2.9c, forming of the notochord by convergent extension Fig. 2.9d and end of gastrulation Fig. 2.9e.

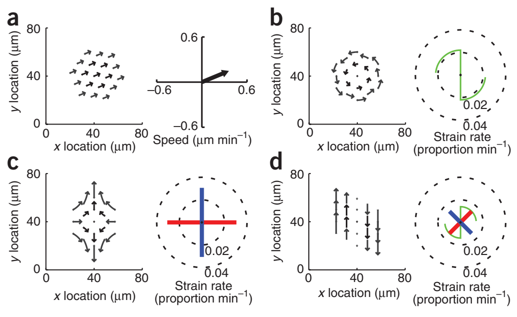
In the field of deep learning on point clouds, there has been a lot of attention in the last several years. The reasons for slow start in developing neural networks for point clouds are significant challenges, such as small scales of datasets, high dimensionality, and the unstructured nature of point clouds [37, 48]. Representing the unstructured data of point clouds as organised data structures is a challenge. Depending on the point cloud representation there are multi-view, volumetric-based, or point-based representations [37].

2.3.1 Multi-view Representation

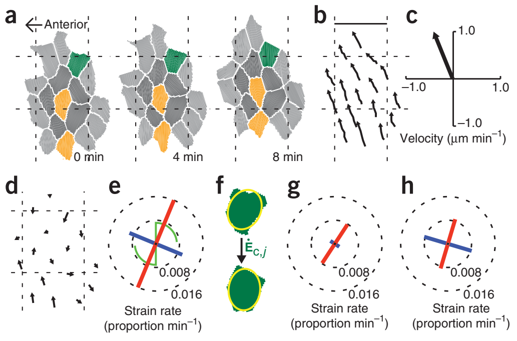
For Multi-view representation the unstructured point cloud is transformed into multiple 2D images. These images enable the extraction of features that describe the whole point cloud [48]. The aggregation process of the view-based features is the main challenge of multi-view methods, such as [49].

2.3.2 Volumetric-based Representation

The point cloud in the volumetric-based representation is voxelized into a grid. Methods for this representation, such as [50] use convolutional neural networks (CNN) on the whole volumetric data. This representation is not suitable for dense data, since memory usage increases cubically with resolution and quantisation errors occur during voxelization [37]. The OctNet network solves this problem by a hierarchical representation of the data [51].

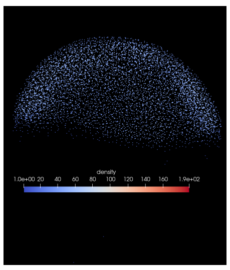


(a) movements with domain representation

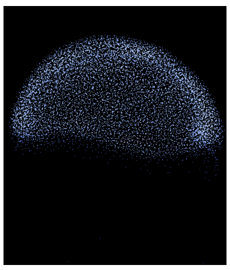


(b) descriptors of zebrafish cells

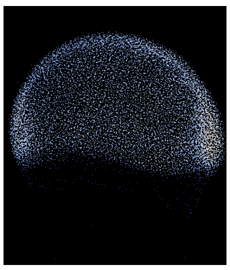
Figure 2.8: In (a) show quantification of speed (a-a), rotation (a-b), strain rates (a-c), and strain with rotation (a-d). The orthogonal principal components of the strain rate are blue and the red size is equal to the amplitude of the strain rate. Rotation is represented by a green scythe, the radius showing radians per minute, and blades in the direction off rotation. In (b) measured descriptors of cells in (b-a) are shown. (b-b) cell centroid trajectories, (b-c) average translation velocity, (b-d) shows the velocity field, (b-e) strain and rotation rate, (b-f) cell shapes approximated by elipses, cell strain rate is computed from elipse evolution from time $t - \Delta t$ to $t + \Delta t$, (b-g) shows area-weighted average of cell shape strain rates, (b-h) cell intercalation strain rates [17].



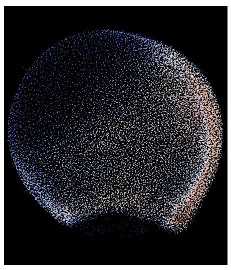
(a) 4.7 hpf



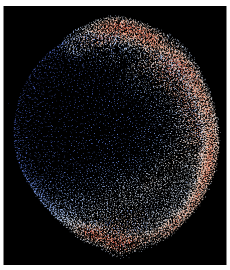
(b) 5.5 hpf



(c) 6.2 hpf



(d) 8.0 hpf



(e) 10.0 hpf

Figure 2.9: Visualization of real embryo development in different stages using Paraview software.

2.3.3 Point-based Representation

Point-based representation directly uses raw point clouds or point clouds augmented with additional information. There is difficulty in processing them due to the need for positional invariance of network processing point clouds and the challenging definition of similarity measurement [52]. Several architectures have been proposed that can process unordered point clouds, such as PointNets [53, 54]. These pioneering works often serve as feature extraction layers in more complex networks, such as [18], which is one of the methods used in this thesis. Generally, point-based methods can be divided into point-wise multilayer perceptrons (MLP), convolution-based, graph-based, and hierarchical-based methods [37]. In this thesis only pointwise and graph-based methods are used, so only these will be described in further detail.

2.3.4 Pointwise MLP Methods

In Pointwise MLP methods, each individual point in the point cloud is taken as a separate element. To process them, MLPs with shared weights are used on all points identically, ensuring that each weight is influenced by every point in the point cloud. Afterwards, a non-linear symmetric function is applied to extract global features. The necessity of using symmetric functions such as MLPs and symmetric non-linearities such as maximum lies in the unorderliness of point clouds [53]. Point clouds are an unordered set, so permutation invariance has to be achieved by using these functions.

In the rest of this section the general architectures of PointNet networks are explained, since parts of those architectures are adapted in more advanced architectures, such as PointLSTM [18], PointRNN [55] or PointConv [56]. In addition, they are used as baseline models for comparison of results.

PointNet

PointNet is the pioneering architecture for processing raw point clouds as input [37]. First, the points are processed by an input transform and a feature transform followed by Multi-Layer Perceptron (MLP) with shared weights. The global feature vector is formed by a max-pooling layer, where the entry in every dimension represents the maximum across all point features in that dimension (shown in Fig. 2.10). This global feature vector is later used for classification, part segmentation, or semantic segmentation [53].

PointNet++

The PointNet model does not adequately capture local features, as all points are considered independently. The extended model of PointNet++ [54] employs a hierarchy by stacking several set abstraction layers (SALs) on each other. SAL consists of the Sampling, Grouping, and PointNet layer, each having different functionality. The sampling layer selects points from the input point cloud that define centroids of the local region and also downsamples the point cloud. The grouping layer creates local regions by finding neighboring points around each centroid. The PointNet layer encodes these local region patterns into a feature vector. [54] By this procedure, PointNet++ can learn both the fine geometric structure and the global features of the entire point cloud. An illustration of the whole network is shown in Fig. 2.11.

2.3.5 Graph-based Methods

In graph-based networks, each point in the point cloud is represented as a vertex in a graph and directed edges are generated based on the neighbors of each point [37]. An illustration of a simplified graph-based network is shown in Fig. 2.12. Firstly a graph from the input points is constructed, then using a learning algorithm, features are extracted. Afterwards, the output is created using a pooling layer. In the rest of this section various types of graph-neural networks related to learning on point clouds are described.

Edge-conditioned Convolution

The feature learning algorithm is different for each graph-based network. In [57] authors introduced a pioneering edge-conditioned convolution for arbitrary graphs and successfully applied it to a point cloud classification. They consider edge weights which are conditioned on the neighborhood of a vertex. They used a max-pooling with graph coarsening which subsamples vertices and edges.

EdgeConv Methods

The neighbors feature extraction is further adapted in DGCNN network [19]. This networks dynamically recomputes the graph in each deep layer of the network, which connects individual

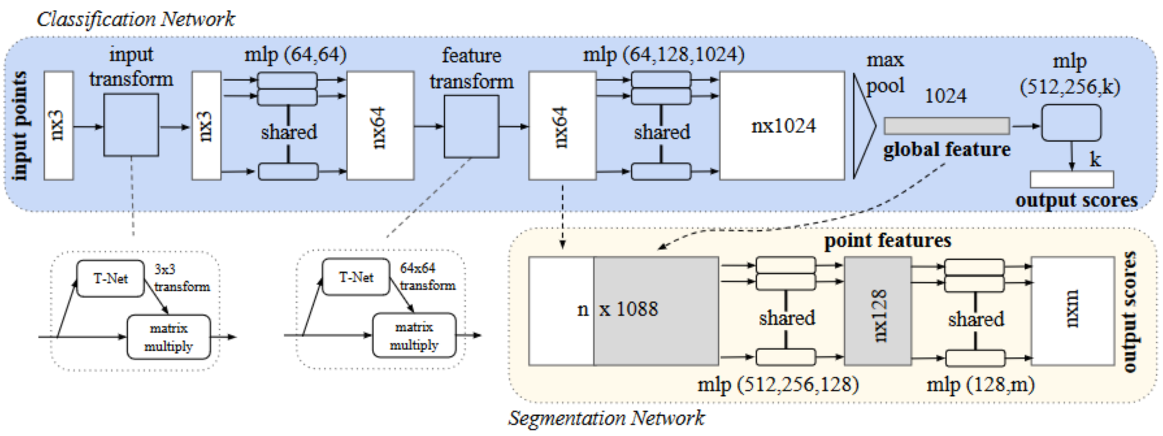


Figure 2.10: The network takes n input points. Those are transformed and shared MLPs are applied. Similarly the feature transform with another MLP layer follows. Lastly, a global feature vector is extracted using max-pooling layer. From the vector, classified labels are obtained by MLP [53].

points on semantic neighborness in deeper layers. Using EdgeConv layer, which combines MLP with max-pooling it extracts high-dimensional feature vectors. An extended version of this network focusing on time-resolved 3D point clouds [20] is used in this thesis for point cloud classification. Further description of both [19] and [20] is in Sec. 3.6.2.

Spectral Domain Methods

Convolution and its filters have a spectral interpretation as well as spatial which was considered previously. Graph signal processing considers convolution as multiplication between signal x as graph coordinates with eigenvectors of graph Laplacian matrix [37]. In PointGCN [58], they approximate the Laplacian by K Chebyshev polynomials with the Gaussian kernel matrix computed on the edges of the graph, multiplied by the learnable weights.

2.3.6 Attention-based Graph Methods

The edge functions of the spatial methods usually connect each vertex with its k-nearest neighbors by a distance-based function. However, the edge cost can be leveraged as in Grid-GCN [59] by including coverage weights and context pooling into distance function to provide semantic reference in edge computing.

2.4 Methods for 3D+t Point Clouds

Point clouds composed of three spatial dimensions and time are the base data representation for perception in robotics and autonomous driving or motion recognition [61]. The time domain means that additional information is contained in the change in shape and density. In addition, new points can appear or disappear over time. In this section state-of-the-art networks used for action recognition, also called sequence-level classification, are described. In addition, the networks used in the thesis for movement classification in embryo point clouds are described in detail. Other ones are only briefly mentioned and reasons are given why the specific network is not implemented. These networks are mostly state-of-the-art of deep learning in gesture recognition, which is a very similar concept to the topic of this thesis.

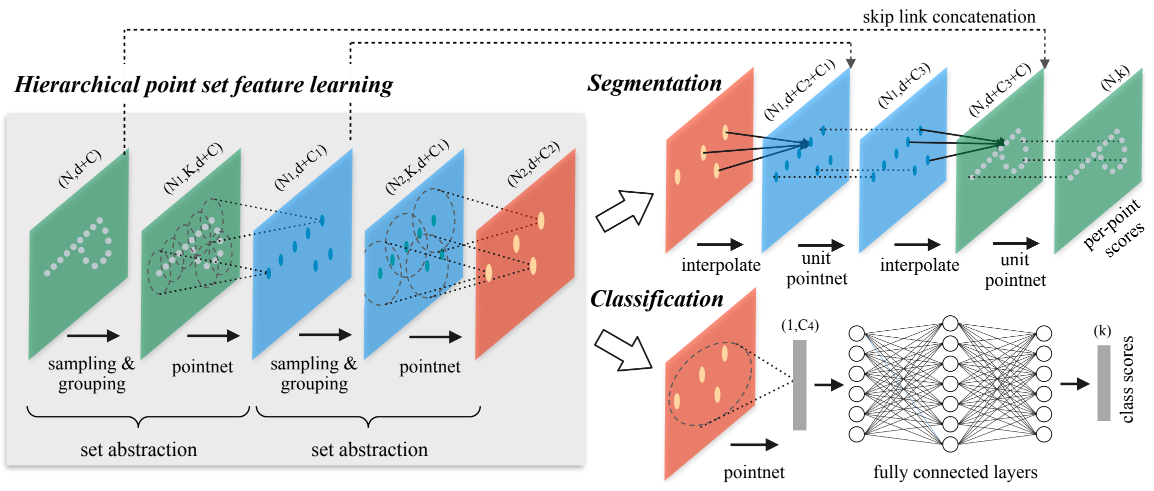


Figure 2.11: Architecture of PointNet++ network. Each set abstraction level contains a Sampling, Grouping and PointNet layer, which progresivelly sample N_i centroids from the point clouds, find neighboring points of the centroids and processes this local neighborhood by shared MLP [54].

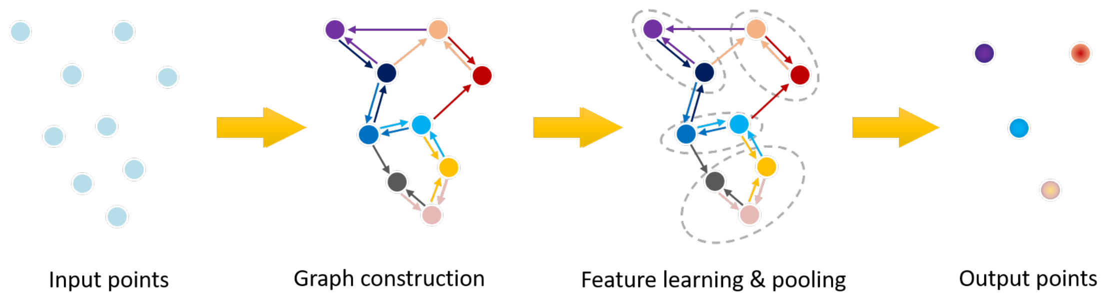


Figure 2.12: Principle of the graph-based networks [60]

There are no surveys yet covering this topic of deep learning, except for a brief listing of existing networks [62] created by the author of several of them.

2.4.1 Attention-based Networks

Transformer-based networks are good at capturing global long-range interactions [63]. Based on the existing architectures, they achieve state-of-the-art results in point cloud 3D+t classification. The Point4D Transformer [64], which architecture is depicted in Fig. 2.13, first generates spatio-temporal local areas by selecting only some local spatio-temporal regions similarly to [65]. Afterwards, the point cloud is subsampled in each frame by FPS and then spatial neighbors are searched. Then a Point 4D Convolution encodes spatio-temporal local areas into a feature vector. These feature vectors are inputs to a self-attention transformer which integrates the features of local areas by capturing long-range relationships accross the entire 3D+t point cloud. Classification scores are obtained using max-pooling folowed by an MLP.

In Point Primitive Transformer (PPTr) [66] they address the drawbacks of the Point4D Transformer, which is hard to optimize for an increasing number of input points and time window extension. They also consider an issue the necessity of loading whole point cloud sequence at once on a graphic card. Due to these drawbacks, they split the architecture on two branches, online and offline. The offline part precomputes the point cloud primitives from

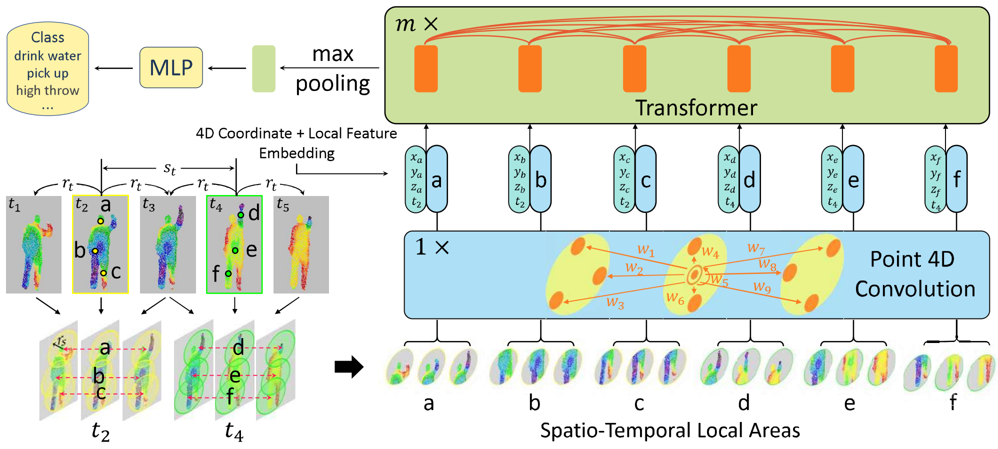


Figure 2.13: Architecture of the P4Transformer [64]. The local spatio-temporal features are input to a 4D convolution followed by attention transformer, which extracts long-range relationships [64]

the long-range sequence of frames. Primitives are groups of points with similar geometric shapes. They used the RANSAC method [67] for fitting the primitives, which interprets and smoothes the feature data that can contain a great amount of errors. Primitives are input to pretrained 3D transformer [64] which is utilized to extract per-point features in every frame. From these features they extract a memory map with size of $C \times M \times L$, where C is a feature dimension, M number of extracted primitives and L number of frames, using point feature max-pooling. The online part processes short point cloud sequences similarly as the offline part. Features from the offline part perform self-attention with the features from the offline part. Despite its effectiveness of PPTr in gesture recognition and its ability to achieve state-of-the-art results, this network has a very complex architecture. Additionally, the absence of the authors' implementation would have made it extremely time-consuming to construct and fine-tune the network from the ground.

Another network from Fan et al. is called Point Spatio-temporal Transformer [68] which is a extension of their previous Point4D Transformer [64]. The whole architecture is the same as Fig. 2.13 only the transformer is augmented with temporal and spatial dimension decoupling. The self-attention mechanism in point cloud sequence calculates an attention score a_{ij}, which indicates the similarity between a point $\mathbf{x}_i^{t_1}$ and $\mathbf{x}_j^{t_2}$. The $\mathbf{x}_i^{t_1}$ features $\mathbf{f}_i^{t_1}$ are encoded by spatial and temporal encoding function with learnable weights which are dependant on spatial and temporal displacement with respect to $\mathbf{x}_j^{t_2}$ to form a encoded feature vector $\mathbf{e}_{i,j}^{t_1,t_2}$. Attention score $a_{i,j}^{t_1,t_2}$ is then multiplied with the encoded feature and the new feature of the point $\mathbf{x}_i^{t_1}$ is obtained as

$$\mathbf{f}_i'^{t1} = \sum_{k=1}^{T} \sum_{j=1}^{N} a_{i,j}^{t1,k} \cdot \mathbf{e}_{i,j}^{t_1,k}, \tag{2.4}$$

where N is the number of points in frame and T is number of all frames in point cloud sequence.

2.4.2 3D+t Convolutional Architectures

Although the point cloud sequences are spatially irregular, unordered in the spatial dimension, but ordered in the temporal dimension, several methods developed a convolution respecting these characteristics. In Point Spatio-temporal network (PSTNet) [65] the authors decoupled the spatial and temporal dimension of the point cloud sequence and processed them separately to address the different scales of spatial and temporal displacements in the point clouds. They propose a spatio-temporal convolution

$$\mathbf{F}_t^{\prime(x,y,z)} = \sum_{k=-l/2}^{l/2} \mathbf{T}_k \cdot \sum_{||(\delta_x,\delta_y,\delta_z)||\leq r} \mathbf{S}^{(\delta_x,\delta_y,\delta_z)} \cdot \mathbf{F}_{t+k}^{(x+\delta_x,y+\delta_y,z+\delta_z)}, \tag{2.5}$$

where updated output feature vector $\mathbf{F}_t^{\prime(x,y,z)}$ of point in time t with coordinates x, y, z is computed using features of points in spatial neighborhood $||(\delta_x, \delta_y, \delta_z)|| \leq r$ and temporal neighborhood $[-l/2, l/2]$ multiplied with both spatial-specific and time-specific kernels $\mathbf{S}^{(\delta_x,\delta_y,\delta_z)}$ and $\mathbf{T}_k$ with learnable weights. With this approach, the authors achieve state-of-the-art results for gesture recognition, better than MeteorNet [69], which does not decouple spatial and time dimensions. This network was not implemented in this thesis because it was already applied for the classification of embryo movements in [70], where it achieved inconclusive results when applied to the real embryo.

Another convolutional method is 3DCNN [71], which converts spatio-temporal point clouds to 3D+t occupancy grids. On these grids, the authors used multidimensional convolution to extract valuable features. However, since the network has to voxelize the point cloud, it is mostly unsuitable for the purpose of this thesis, since it can be vulnerable to discretization error and loss of fine cell movement structure.

2.4.3 Point-based Approaches

The methods directly working with point clouds in a similar manner as PointNet networks [53, 54] usually augment the spatial coordinates of the points with the temporal dimension. The first network to utilize this raw point cloud sequence augmentation is called MeteorNet [69]. This extension of PointNet++ network aggregates information from spatio-temporal neighborhoods. So called Meteor module directly takes point cloud sequence and outputs features

$$\mathbf{h}(\mathbf{x}_i^t) = \max_{\mathbf{x}_j^{t'} \in \mathcal{N}(\mathbf{x}_i^t)} (MLP(\mathbf{f}_j^{t'}, \mathbf{f}_i^t, \mathbf{x}_j^{t'} - \mathbf{x}_i^t))), \tag{2.6}$$

where $\mathbf{h}$ are module outputs, $\mathbf{x}_i^t$ are augmented i-th point coordinates in time t, MLP has shared weights for all points as in PointNets and $\mathcal{N}(\mathbf{x}_i^t)$ is a spatio-temporal neighborhood of point $\mathbf{x}_i^t$. Even though the network uses temporal information from several neighboring frames, capturing longer relations is not possible. Another drawback is determining correct spatio-temporal region. For sparse point clouds, there may be more temporal neighbors than spatial and for dense, there are mostly spatial neighbors. This network is often a backbone of other state-of-the-art networks such as Point Spatio-Temporal Transformer (PST^2) [72], where it is used as a feature extractor.

SequentialPointNet [73] uses PointNet++ based SAL architecture combined in an inovative hyperpoint mixer. The Sampling and Grouping are the same as PointNet++ ones described in 2.3.4. The PointNet as the third stage of SAL is augmented with attention mechanism. The output of this layer is a dot product of attention scores with the grouped features from the grouping layer forwarded through MLP and max-pooling layer. There are two sequential augmented SALs followed by MLP. Classification scores are obtained using max-pooling layer.

FlickerNet [74] is an extension of the PointNet++ network [54] for time-resolved point clouds. Similarly as PointNet++, FlickerNet employs multiple spatio-temporal set abstraction levels (SALs) to extract hierarchical features from the point cloud by mapping a given set of points to a sparser set. In PointNet++, the point cloud is considered as an unordered set, which is an unsuitable representation for action recognition. They present a novel sampling layer called the Spatio-Temporal Sampling layer (STS-layer) and a grouping layer called the Spatio-Temporal Grouping layer (STG-layer). The last section of SAL remains as a simplified PointNet.

Overall output equation of one SAL which takes input point cloud sequence $P_{i,t} = \{x_{i,t} | i = 1, 2, \dots, N, t = 1, 2, \dots, T\}$, where N is number of points and T number of time steps, can be written as

$$y_{i,t} = \text{MLP}\left(\max_{(j,t+\Delta t) \in \mathcal{N}(p_i,t), j=1,\dots,N} \left\{ \text{MLP}\left(g(x_{p_i,t}, x_{j,t+\Delta t}) \right) \right\} \right), \tag{2.7}$$

where MLP are PointNets (shared MLP layers), g is the STG-layer making regions from the spatio-temporal neighborhood $\mathcal{N}(p_i, t)$ of selected centroid points $x_{p_i,t}$ by a STS-layer. The neighborhood of each centroid is from a short time window Δt.

In more detail, STS-layer is implemented as uniform sampling, which is different from PointNet++, where they used Farthest Point Sampling (FPS)[3]. Except for grouping the neighboring points of the centroid, the STG-layer also translates the coordinates of the selected points relative to the centroid, which helps to capture motion change. FlickerNet is a baseline for a PointLSTM network which was implemented in this thesis and is further described in Sec. 3.6.1.

2.4.4 Recurrency-based Networks

The advantage of the recurrency-based networks lies in capturing temporal features by the recurrent layer, mostly LSTM [75] which excells in capturing long-term relationships. The two reviewed networks, PointGest [61] and PointLSTM [18], are very similar in their architectures. Both use a combination of PointNet++ and LSTM. The PointGest network uses PointNet++ for framewise spatial feature extraction, so the authors do not mix the temporal domain with the spatial one. On the other hand, PointLSTM incorporates FlickerNet STS and STG layers [74] to handle the entire point cloud sequence at once. After this step, both networks use LSTM for the extraction of temporal features. The PointGest uses two stacked LSTM layers, while PointLSTM uses only one in its third layer. PointGest network was not applied to the same dataset as PointLSTM, so they can hardly be compared. It also does not have a public implementation available. Therefore, PointLSTM was applied to dynamic point cloud classification and is further introduced in Sec. 3.6.1.

[3]FPS iteratively samples points from the point cloud. In the first iteration, random point is sampled. In each other iteration, the point farthest from the already sampled set it sampled.

Chapter 3
Methods and Implementation

For classification of the movements during embryogenesis, supervised deep learning methods were used in this thesis. For deep learning methods, it is necessary to provide the network with an annotated dataset with a huge amount of training data to successfully train the network. One caveat of real point cloud of embryo development is that it is not possible to annotate by manual labelling, due to excess of abstraction when human work with point cloud in raw form, complexity of movements of the cells in the point cloud and immense amount of points (cells) even in one point cloud. Therefore, it is necessary to create a synthetic annotated dataset, where individual movements can be simulated, and in addition labels of dominant movement for each point in the point cloud are obtained. This dataset later serves as training data for deep learning models.

In this section, an unlabelled dataset of real cell development recording is introduced, and then a description of the synthesized data generation process is provided, including movement characterization, simulation details, visualization tools and processing. Processing includes the application of selected neural networks, described in Sections 3.6.1 and 3.6.2.

3.1 Embryo Data Acquisition

Due to the use of a fast and accurate miscroscope, there is recently the possibility of creating *in vivo* images of a real embryo. A device called digital laser light sheet fluorescence microscopy (DSLM) by [11] rapidly scans the specimen by a micrometre-wide laser beam. By moving it in horizontal and vertical directions, it generates a plane of light that captures the specimen. High accuracy is achieved by illuminating each line with the same intensity, which is important for scanning over long time periods to maintain consistency in later post-processing, mainly cell tracking or segmentation.

The quality of the images is also better than that of simple light-sheet microscopy due to the absence of aperture [11]. Also, an order of magnitude better illumination efficiency is the merit of focussing the illumination power of the light source on a single line. In Fig. 3.1 the whole device is shown. The laser beam from the laser scanner illuminates the specimen, which excites the fluorophores in the specimen along the single line. The emitted light is detected at a right angle to the illumination axis. The F-theta lens converts the laser scanner movement to vertical displacement of the laser light. Tube and illumination lenses are used to focus the laser beam onto the specimen.

The measurements used in this thesis [23] span 2.5 - 17 hpf of the development of zebrafish. Each 3D image was extracted from a stack of 500 frames with a depth spacing of 2 μm, which took 50 seconds to obtain. Each frame covers the field of view 1038×875 μm^2 with pixel size 0.406×0.406 μm^2, which is a single cell resolution. This resolution fits well with the size of a zebrafish embryo with diameter 750 μm.

3.2 Digital Embryo Extraction

From the data obtained by DSLM, cells are segmented to form a digital embryo using [23], where each point corresponds to a cell. This point cloud was cropped to 4.7 - 10 hpf, corresponding to late cleavage and gastrulation.

Segmentation was performed individually on each 3D image from the 500 frame stack. First, the images were filtered using Laplacian of Gaussian (LoG) filters to form a LoG space-scale maximum projection [76]. In this projection, local extrema marked as seed points are found in the 26-neighborhood. Each blob corresponding to the respective seed point is cropped proportionally to the radius of the blob. For fast approximate segmentation, a simple pixel-weighted mask is created for each blob separately. This mask is a dot product of the seed point normal and normalised gradient at the respective pixel multiplied by the distance of the pixel to the seed point. This transformed image is then segmented using the Otsu's threshold method [77]. The tracklets of the respective cells were calculated using the nearest-neighbor tracking algorithm implemented in [14].

3.3 Annotated Synthetic Data

To create a large amount of training data for movement recognition, it is necessary to have a method that generates synthetic annotated data that resemble movements in real embryos. These movements have to be simple enough to be easy to simulate, diverse in all their parameters, and sufficiently capture the complex nature of real embryo movements. They also should serve as elemental descriptors of the complex movement patterns, which could be their superposition, e.g. convergent rotation. Some methods of semi-synthetic dataset generation already exist, such as [78]. However, those movements are not simple, since they also simulate cell division rates and mutual cell forces. They were previously used in [70] but deep learning models learnt on these point clouds achieved inconclusive results when applied on real embryo point clouds.

Synthetic point cloud parameters were extracted from real point cloud visualizations using a Kitware scientific visualization tool Paraview [79]. Data for visualizations were taken from [23]. Other parameters were taken from related literature, such as [1, 7], where a detailed description of the zebrafish parameters is provided, including the radius and thickness of the cell layer during embryo development.

3.4 Point Cloud Generation

The point clouds were generated by sampling random points on a sphere. To get uniformly sampled points, they were sampled in spherical coordinates[1] centered around spherical angles $\theta_c = \pi/2$ and $\phi_c = \pi/2$. In this way, there are no singularities in angle sampling. To vary the density of the point cloud, the neighborhood around the center angle is from a uniform distribution $\delta \in U(\pi/8, \pi/4)$. The angles are then $\theta \in U(\theta_c - \delta, \theta_c + \delta)$ and $\phi \in U(\phi_c - \delta, \phi_c + \delta)$. To correctly simulate the layer of cells on the embryo boundary, the radial distribution was chosen to be in the real range measures on zebrafish embryos, and therefore points are simulated on a spherical shell. From the related literature, the range of cell width $w_c \in [10, 100]\ \mu m$ was taken. Furthermore, the radius of the entire embryo was

[1] r is radial distance, θ is azimuthal angle and ϕ is polar angle

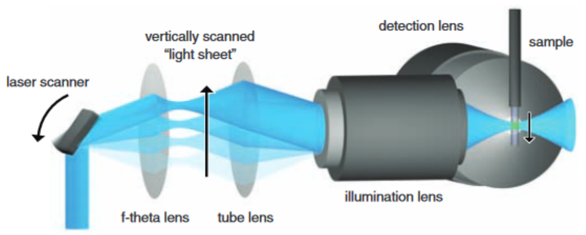


Figure 3.1: DSLM device for capturing the specimen consists of four lenses and a laser scanner. The laser scanner illuminates the specimen, which emits light detected by detection lens. For focussing on the specimen, tube and illumination lenses are used. For vertical movement along the specimen, f-theta lens is used [11].

taken from the literature as $r_c = 400\ \mu m$. The radial distribution is then $r \in [r_c - w_c, r_c]$. In this way, a static first frame of the point cloud is created.

To make the point cloud dynamic, two approaches were tested. In the first case, a simple point cloud, which was later used as training data, the entire point cloud was moving with the predefined movement throughout the simulation time. In the second, an artificial point cloud, which is more complex and is ideal for testing data, only a specific part of the point cloud was moving and changing over time.

For both approaches, there are seven movements to simulate, inspired by Fig. 2.7: nomove, uniform, expansion, compression, rotation, radial, and shear. The movement is applied (position difference of the cell location) in each time frame iteration together with a noise jittering derived from Paraview visualization and checked with related biological literature. Radial jitter is from normal distribution $\mathcal{N}(0,5)\ \mu m$ and angular jitter is from $\mathcal{N}(0, \frac{5}{r_c})$. In the rest of this section, specifications of each movement type are described.

3.4.1 Nomove

This movement is taken as a baseline for all other movements. It is a simple idle state of each cell (point) with added jitter. In later movement classification and movement type quantification, nomove is mostly predicted as a false positive for other movement classes.

3.4.2 Uniform

Uniform movement is a linear movement of cells on the sphere. The speed of this movement was considered from the interval $v = [2, 7]\ \mu m/s$. The direction of velocity is also randomly selected. The angle increase of ϕ is $\frac{v}{r_c}$. The angle θ remains constant to allow the point cloud to move along the equator of the sphere and not encounter a singularity. Later, the variability in motion is ensured by random rotation (discussed in the next section).

3.4.3 Expansion and Compression

Both expansion and compression are simulated in a similar way. First, the points are rotated by the negative angle of the point cloud centroid and then projected onto a plane by stereographic projection from the north pole (0,0,1). In this projection point coordinates are normalised to the unit sphere

$$x_n = \frac{x}{\sqrt{x^2+y^2+z^2}}, \quad y_n = \frac{y}{\sqrt{x^2+y^2+z^2}}, \quad z_n = \frac{z}{\sqrt{x^2+y^2+z^2}}, \tag{3.1}$$

then the projection is

$$\xi = \frac{x_n + y_n i}{1 - z_n} \tag{3.2}$$

and planar coordinates of x and y are

$$x_{pl} = \mathrm{Re}(\xi), \quad y_{pl} = \mathrm{Im}(\xi). \tag{3.3}$$

Second, the vectors of compression and expansion are computed in polar coordinates from the projected points with the substracted centroid. For the expansion vector field, the polar coordinates are updated by the factor $f_{exp} \in [0.990, 0.993]$

$$r_{exp} = r_{pl} \cdot f_{exp}, \quad \theta_{exp} = \theta_{pl} \tag{3.4}$$

and for compression vector field

$$r_{comp} = \frac{r_{pl}}{f_{exp}}, \quad \theta_{comp} = \theta_{pl}. \tag{3.5}$$

Then, the polars are converted back to Cartesian coordinates and then back on the sphere using stereographic backprojection.

$$\bar{\xi} = x_{pl} + y_{pl} i \tag{3.6}$$

as

$$x_{sp} = \frac{2x_{pl}}{1 + \bar{\xi}^2}, \quad y_{sp} = \frac{2y_{pl}}{1 + \bar{\xi}^2}, \quad z_{sp} = \frac{\bar{\xi}^2 - 1}{1 + \bar{\xi}^2}. \tag{3.7}$$

To convert it back to an original size radius but keep the cell distribution, we convert x_{sp}, y_{sp} and z_{sp} to spherical coordinates r_{sp}, θ_{sp} and ϕ_{sp} and replace r_{sp} with the original r.

3.4.4 Rotation

The rotation movement has the same procedure of generating a vector field using stereographic projection, but the updating equation of r and θ in polar coordinates is

$$r_{rot} = r_{pl}, \quad \theta_{rot} = \theta_{pl} \pm (\frac{v}{r_c} + \delta_{rot}), \tag{3.8}$$

where $\delta_{rot} = \frac{2\pi}{1000}$ is additional fixed rotation to the one based on the speed v and embryo radius r_c.

3.4.5 Radial

The radial movement is simply generated in spherical coordinates by only changing the radius of the points

$$r_{rad} = r + \sqrt[3]{v}. \tag{3.9}$$

Only an extending radial movement towards the sphere surface was simulated.

3.4.6 Shear

Similarly, as expansion, compression, and rotation, using stereographic projection, vectors are computed. For shear movement, parametrizable 2D shear matrix is utilized

$$R_{shear} = \begin{pmatrix} 1 & \lambda \\ \mu & 1 + \lambda \cdot \mu \end{pmatrix}, \tag{3.10}$$

where $\lambda = \lambda_{rand} \cdot \delta_s$ and $\mu = \mu_{rand} \cdot \delta_s$, where $\lambda_{rand} \in \pm U(0.2, 1)$ and $\mu_{rand} \in \pm U(0.2, 1)$ are from uniform distribution and have same sign, tuning parameter $\delta_s \in U(0.14, 0.2)$.

3.5 Dataset Creation

As mentioned in the previous section, two types of dataset were created. For training and validation purposes, the simple point cloud type is proposed. The training and validation dataset consist of 1000 point clouds of each movement, each consisting of 2048 points, each with random parameters specific for the embryo and specific for the movement. The embryo parameters (summarised in Tab. 3.1) as well as movement-specific parameters are explained in detail in the previous section.

Moreover, since each simulated point cloud is only a part of sphere surface, it was necessary to randomise rotations of them to cover the whole sphere surface of the embryo. For this reason, random rotation was applied to the dynamic point cloud using the random rotation function [80] that uniformly samples the rotations.

The second point cloud type is only for testing purposes. It should more precisely mimic how a real embryo could dynamically develop. Therefore, in this thesis, it is called an artificial point cloud type. In this simulation, a point is chosen as the central point of movement, and in each frame the k nearest points around the central point of movement are found. The positions of these points are updated with respect to the simulated movement. In the next iteration, the new nearest points around the central moving point are selected. In addition, the simulated movement begins in the j-th frame, to first simulate the inactive point cloud in frames $[1, j]$.

During simulation, it is better visible if the network can correctly classify partially moving point cloud (group of points containing both moving and inactive points).

3.6 Implemented Network Architectures

Two networks were chosen from the methodologies discussed in Sec. 2.4 and implemented for the task of movement classification. The first is PointLSTM, which is an extension of the PointNet++ architecture that incorporates the STG and STS layers from FlickerNet [74], along with a recurrent LSTM layer [81] that is well suited for time-dependent input data. A detailed description is given in the following section. The second network, DGCNN [19], is a graph neural network introducing an innovative EdgeConv layer that extracts point features by weighting its neighbors in a graph constructed with learnable weights. While originally designed for 3D point clouds, this network was adapted for training on 3D+t point clouds. A detailed description can be found in Sec. 3.6.2.

3.6.1 PointLSTM

PointLSTM network is the first implemented network to solve the movement recognition task. The functionality, architecture, and advantages are elaborated further.

This network combines state information from adjacent points in the previous frame with current-frame features to update current-frame states using a weight-shared Long Short-Term Memory (LSTM) layer. Compared to PointNet architectures [53, 54], it is capable of capturing long-term spatio-temporal relationships [18]. The baseline of this network is FlickerNet [74].

The authors mention that when the point clouds are ordered (such as in the case of point tracklets), there exists a connection between a point in the current frame with its corresponding point in the previous frame, as shown in Fig. 3.2a. This relationship can be effectively captured by processing the sequences using an LSTM layer with shared weights. However, in practical scenarios with real point clouds, this relationship may not always exist. Therefore, it is more beneficial to generalise the architecture for unordered point clouds. In

parameter name	range
movement type	[nomove, uniform, extension, compression, rotation, radial, shear]
embryo radius	$r_c = 400\mu m$
sampled angles	$\theta \in U(\theta_c - \delta, \theta_c + \delta)$ $\phi \in U(\phi_c - \delta, \phi_c + \delta))$
sampled radius	$r = [r_c - w_c, r_c]$
cell width	$w_c = [10, 100]\mu m$

Table 3.1: Table of all the parameters used for simulating the embryo

the case of unordered point clouds (Fig. 3.2b), the temporal connection may be unknown or does not exist. In applications where point clouds are used to represent embryos, the connections between points represent cell tracklets. The absence of a connection indicates that the corresponding cell has died (apoptosis), appeared from cell division (mitosis), or moved out of the field of view. During processing, each point is concatenated together with its nearest neighbors from previous frames. This approach of neighborhood processing can be advantageous even for ordered point clouds. The spatial neighborhood provides more insights into the local dynamics than just the individual points itself, considering that their movement is rather stochastic. By incorporating information from the neighborhood, the impact of stochasticity is reduced, and only the main dynamic component, the classified movement, is classified by the network.

LSTM Layer

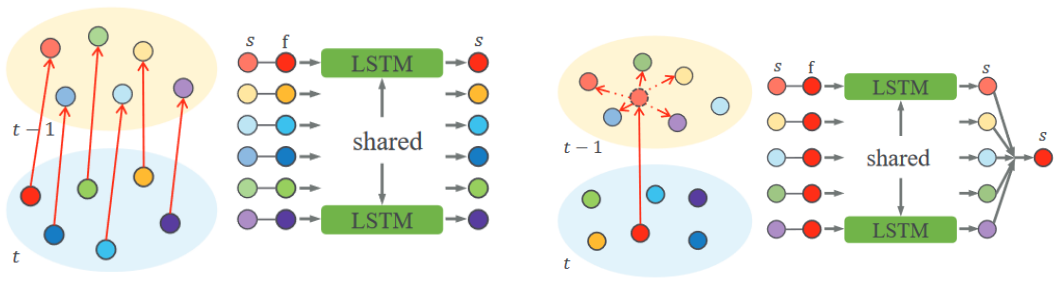
The backbone of the network is taken into account by the time-dependence and recurrency in the decision process by the LSTM network. As mentioned previously, the point features in the current frame are concatenated with neighboring points in the previous frame. Points are processed by weight-sharing LSTM pair-wise for each pair of current features and neighboring points in the previous frame. The following pooling operation updates the point state and extracts local neighborhood information.

The LSTM layer, initially introduced by Hochreiter and Schmidhuber [75], is a form of recursive neural network that excels in long-term modelling. The equation can be used to express the layer dynamics as

$$\boldsymbol{h}^{(t)}, \boldsymbol{c}^{(t)} = LSTM(\boldsymbol{y}^{(t)}, \boldsymbol{h}^{(t-1)}, \boldsymbol{c}^{(t-1)}), \tag{3.11}$$

where $\boldsymbol{y}^{(t)}$ is layer input, $\boldsymbol{h}^{(t-1)}$ is hidden state from past frame and cell state from previous frame is denoted $\boldsymbol{c}^{(t-1)}$. Output is current frame hidden and cell state.

Within the LSTM layer, the input $\mathbf{y}^{(t)}$ is forwarded through the input gate $\mathbf{i}(t)$, forget gate $\mathbf{g}(t)$, output gate $\mathbf{o}(t)$ and cell prediction gate $\tilde{\boldsymbol{c}}^{(t)}$. All gates are equipped with corresponding weights for input-to-hidden and hidden-to-hidden transformations, including biases. Furthermore, these subcalculations are used to calculate the hidden state $\mathbf{h}^{(t)}$ and the cell state $\mathbf{c}^{(t)}$ of the current frame using the Hadamard product ($\odot$).



(a) PointLSTM for ordered PointCloud (b) PointLSTM for unordered PointCloud

Figure 3.2: In (a) the point has connections with previous frames, so they are directly processed by LSTM. In (b), the point is processed with nearest neighbors from previous frames. [18]

The update is represented by equations

$$\begin{aligned}
\mathbf{i}^{(t)} &= \sigma(\mathbf{U}^{(i)}\mathbf{y}^{(t)} + \mathbf{W}^{(i)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(i)}) \\
\mathbf{g}^{(t)} &= \sigma(\mathbf{U}^{(g)}\mathbf{y}^{(t)} + \mathbf{W}^{(g)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(g)}) \\
\mathbf{o}^{(t)} &= \sigma(\mathbf{U}^{(o)}\mathbf{y}^{(t)} + \mathbf{W}^{(o)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(o)}) \\
\tilde{\mathbf{c}}^{(t)} &= \tanh(\mathbf{U}^{(c)}\mathbf{y}^{(t)} + \mathbf{W}^{(c)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(c)}) \\
\mathbf{c}^{(t)} &= \mathbf{g}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)} \\
\mathbf{h}^{(t)} &= \mathbf{o}(t) \odot \tanh(\mathbf{c}^{(t)}),
\end{aligned} \tag{3.12}$$

where $\mathbf{U}^{(\cdot)}, \mathbf{W}^{(\cdot)}$ are input-to-hidden and hidden-to-hidden state weight matrices and $\mathbf{b}^{(\cdot)}$ is bias vector. As activation in the LSTM layer they used the sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ for the input, forget, and output gate, and the hyperbolic tangent $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ for prediction gate and the update of hidden state.

Point-independant Processing

They propose two methods of point cloud processing architecture. First is point-independant, where each pair of points in the point cloud and neighbor in the previous frame has its own cell state and hidden state (e.g. point $p_i^{(t)}$ has neighbors $p_j^{(t-1)}, j \in \{1, \dots n\}$ in the previous frame). The input to the LSTM layer is then

$$\mathbf{y}_{i,j}^{(t)} = [\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t-1)}, \mathbf{f}_i^{(t)}], \tag{3.13}$$

where $\boldsymbol{x}$ is the coordinate vector of the point and $\boldsymbol{f}$ is its feature vector (for the application to the classification of embryo movements, there was no additional feature vector that augmented each point). LSTM update for one point-neighbor pair can then be rewritten as

$$\tilde{\boldsymbol{h}}_{i,j}^{(t)}, \tilde{\boldsymbol{c}}_{i,j}^{(t)} = LSTM(\boldsymbol{y}_{i,j}^{(t)}, \boldsymbol{h}_j^{(t-1)}, \boldsymbol{c}_j^{(t-1)}). \tag{3.14}$$

Updated states of the point in current state are then outputs of max-pooling layer over all computed point-neighbor states.

$$\begin{aligned}
\boldsymbol{h}_i^{(t)} &= g(\tilde{\boldsymbol{h}}_{i,1}^{(t)}, \tilde{\boldsymbol{h}}_{i,2}^{(t)}, \dots, \tilde{\boldsymbol{h}}_{i,n_{t-1}}^{(t)}) \\
\boldsymbol{c}_i^{(t)} &= g(\tilde{\boldsymbol{c}}_{i,1}^{(t)}, \tilde{\boldsymbol{c}}_{i,2}^{(t)}, \dots, \tilde{\boldsymbol{c}}_{i,n_{t-1}}^{(t)}),
\end{aligned} \tag{3.15}$$

The disadvantage of this method is that it is slower for point clouds with a large number of points due to updating many more additional states.

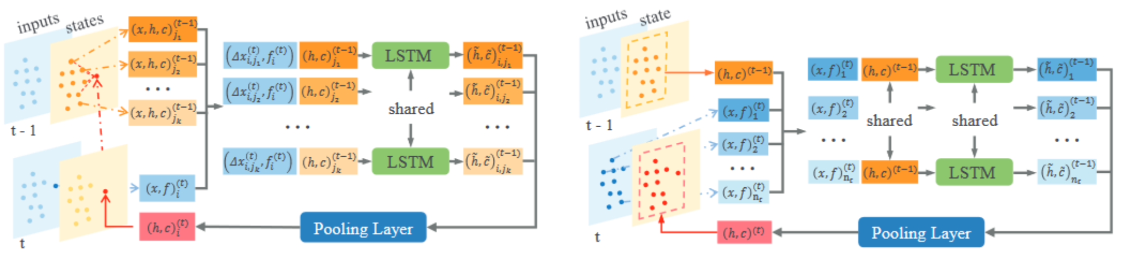


Figure 3.3: Two types of schemes of LSTM. In a point-independant scheme in (a) has each point in the point cloud its hidden and cell state, which is updated based on the inputs and states of previous frame neighborhood. Global states are obtained by maxpooling over point independant states. In (b) is a point-shared scheme, where points share the same hidden and cell state. Global states are obtained by averaging [18].

Point-shared States Processing

The second method of point cloud processing is called point-shared states and simplifies the update process by sharing the cell and hidden states within the frame.

$$\begin{aligned} \boldsymbol{y}_i^{(t)} &= [\boldsymbol{x}_i^{(t)}; \boldsymbol{f}_i^{(t)}] \\ \tilde{\boldsymbol{h}}_i^{(t)}, \tilde{\boldsymbol{c}}_i^{(t)} &= LSTM(\boldsymbol{y}_i^{(t)}, \boldsymbol{h}^{(t-1)}, \boldsymbol{c}^{(t-1)}), \end{aligned} \tag{3.16}$$

Overal hidden and cell state in time t is given as average over all $\tilde{\boldsymbol{h}}_i^{(t)}$ and $\tilde{\boldsymbol{c}}_i^{(t)}$ using the average-pooling layer. The method for point-independent states proved to be better, due to gathering information from neighbors in previous frames by using the difference in the point coordinates.

Network Architecture

In each layer of the network, in front of the MLPs, there is a special layer of grouping the local neighborhood of points from the previous layer that extracts local spatial information from the local neighborhood. It also serves as a downsampling of point clouds between two layers (stages in Fig. 3.4). The effect of gathering information from neighbors is conceptually similar to the receptive field of convolutional neural networks. Two methods of selecting the neighboring points using grouping methods were investigated. These methods were first introduced in MeteorNet [69].

The first method is called direct grouping, which directly finds the nearest neighbor in the previous frame of the point in the current frame (Fig. 3.3a). The second method is the aligned grouping, which is used mostly for rigid objects (Fig. 3.3b). The main objective is to estimate the flow of the scene. However, some scenes are non-rigid, and therefore this method can be less accurate. In the article, the estimate of the point location in the previous frame was done by aligning centroids of the local neighborhood. The flow is then the difference of those centroid coordinates.

$$\begin{aligned} \Delta\overline{\boldsymbol{x}}^{(t)} &= \frac{1}{n_{t-1}}\sum_{i=1}^{n_{t-1}} \boldsymbol{x}_i^{(t-1)} - \frac{1}{n_t}\sum_{i=1}^{n_t} \boldsymbol{x}_i^{(t)} \\ \Delta\boldsymbol{x}_i^{(t)} &\approx \Delta\overline{\boldsymbol{x}}^{(t)}, \text{for } i = 1, \ldots, n_t. \end{aligned} \tag{3.17}$$

The entire network architecture consists of five stages (Fig. 3.4). In stages 1 to 4, the points are first grouped using either direct or aligned grouping, forwarded through MLP with ReLU, followed by the maxpooling operation processing grouped points. The third stage before the grouping method turned out to be the best location for the LSTM layer. In the fifth stage, there is no grouping, but only the MLP and max-pool layer, from which global features are obtained. After the fifth layer, the prediction scores are calculated by softmax.

3.6.2 DGCNN

Dynamic graph convolutional neural network (DGCNN) is the second implemented architecture for point cloud movement recognition. In the following sections, its architecture and functionality are explained. Also, since the original architecture is designed only for 3D point clouds [19], an extended approach was modified from [20]. For working with time-resolved 3D point clouds, the tracklet distance, referred to as the spatial-temporal distance (ST distance) was utilized.

Similarly to the convolutional layers, which effectively work on image data due to utilizing learnable kernels sweeping over the images extracting features, in DGCNN authors created an EdgeConv layer. This new operation works well on point cloud data, because it does not neglect the geometric nature of them [19]. Similarly to convolution, EdgeConv extracts features from the local neighborhood of each point in the point cloud and weights them by the learnable kernels.

EdgeConv is applied in each layer of the network by firstly constructing the local neighborhood graph. This graph is not static, since it is updated in each layer, by computing k-nearest neighbors repeatedly in each layer, which for deeper layers leads to grouping of points in semantic space of features. For each point $\boldsymbol{x}_i$ (vertex in the graph), the output of the edge convolution is

$$\boldsymbol{x}'_{i,m} = \max_{(i,j)\in\mathcal{N}_{\boldsymbol{x}_i}} \boldsymbol{e}_{i,j,m}, \tag{3.18}$$

where $\boldsymbol{x}_i$ are point coordinates (or features in deeper layers) and for each $\boldsymbol{x}_j$ in $\boldsymbol{x}_i$ neighborhood $\mathcal{N}_{\boldsymbol{x}_i}$ edge features are computed. These features $\boldsymbol{e}_{i,j,m}$ describe the relationship between the point and its neighbors. The edge features are computed with a non-linear edge function h_Θ as

$$\boldsymbol{e}_{i,j,m} = h_\Theta(\boldsymbol{x}_j - \boldsymbol{x}_i, \boldsymbol{x}_i) = \mathrm{ReLU}(\theta_m(\boldsymbol{x}_j - \boldsymbol{x}_i) + \phi_m \boldsymbol{x}_i) \tag{3.19}$$

for each of M learnable parameters $\Theta = (\theta_1, \dots, \theta_M, \phi_1, \dots, \phi_M)$. Therefore, it follows the architecture of MLP with shared weights and is also differentiable. In the first panel of Fig. 3.5 the edge convolution is visualized as MLP with inputs $\boldsymbol{x}$, outputs $\boldsymbol{e}$, and edge functions h. In the next schemes, the operation of computing the edge features is visualized.

The overall architecture is depicted in Fig. 3.6, which is very similar to both PointNets described in Sec. 2.3.4 and 2.3.4. For purposes of the thesis, the classification branch was utilized. Also, for a better performance comparison with the first implemented PointLSTM, the first branch is better. As mentioned above, the graph is updated in each layer by creating a new k-nearest neighbor graph in the feature space using a pair-wise distance metric (Euclidean distance). After graph construction an MLP combined with max-pooling extracts the features of the local neighborhoods. After 5 layers, where in each layer the semantic meaning of points is different, outputs of each layer are aggregated to a 1D vector which is used for generating classification scores.

DGCNN works well for 3D point cloud classification and segmentation. But is it possible to apply for time-resolved 3D data? The authors of [20] successfully applied the network on 2D+t point clouds by segmenting the 2D+t point cloud on multiple temporal segments and processing each segment separately. More specifically, they apply a sliding window on the point cloud, so that they get some overlap of neighboring time segments. With this approach,

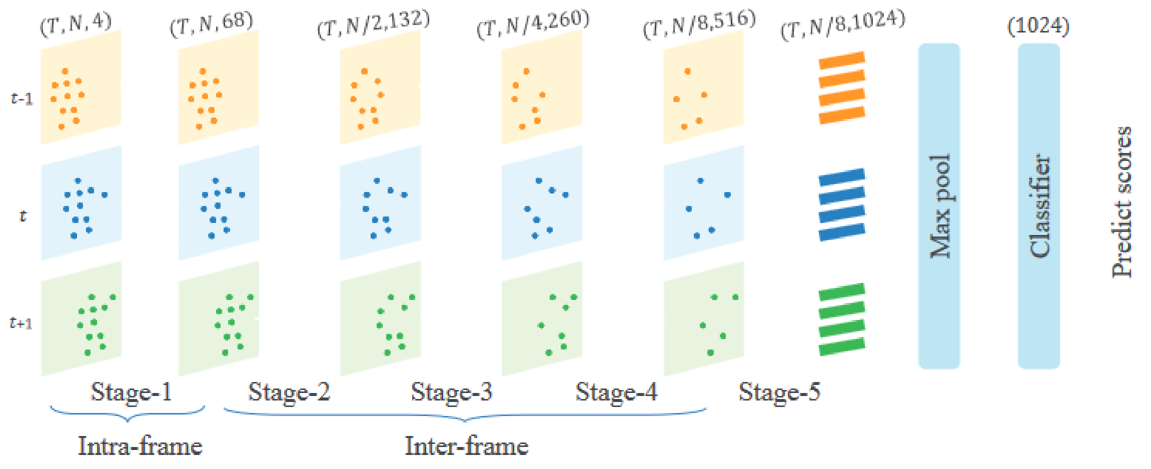


Figure 3.4: Network architecture consisting of five stages. Intra-frame layer operates on raw point cloud, inter-frame on the extracted features from the previous layer. Each stage consists of a grouping operation, MLP and maxpool to aggregate information from local neigborhood. Outputs of the network are prediction scores of each class [18].

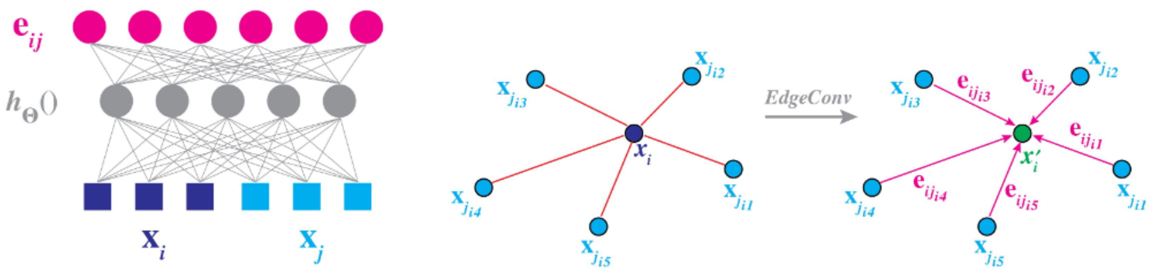


Figure 3.5: EdgeConv layer which processes k-nearest neighbors of each point similarly as MLP by applying edge function h_Θ with learnable weights [19].

they were able to train the network for gesture recognition with state-of-the-art results. However, they construct the knn graph in the spatio-temporal domain without adjustment for the temporal neighbors. In this manner, there is no difference between extracting features from spatial and temporal neighbors.

In this thesis, the idea of [20] was adapted with the difference of using 3D+t point clouds instead of 2D+t. Furthermore, the issue with spatio-temporal neighborhood and the incorporation of information about point tracklet dynamic development was addressed by using Spatio-temporal distance (ST-distance) metric inspired by [82].

Computing the ST-distance

This metrics, originaly for trajectory distance computation, are based on [82]. In each layer of the network k-nearest neighbors are computed not using the Euclidean metric, but a ST-distance defined between two tracklets $\tau_i = (\boldsymbol{f}_1^i, \boldsymbol{f}_2^i, \ldots, \boldsymbol{f}_{end}^i)$ and $\tau_j = (\boldsymbol{f}_1^j, \boldsymbol{f}_2^j, \ldots, \boldsymbol{f}_{end}^j)$ which are indexed in time t. Trajectory points $\boldsymbol{f}$ are feature vectors depending on the layer for which the ST-distance is computed. However, the feature vector indices are the same as original input tracklet ones, so that tracklet consistency is preserved also in higher-dimensional semantic space. ST-distance is computed as

$$d_{\tau_i,\tau_j} = \frac{\sum_{k=k_{start}}^{k_{end}} ||\boldsymbol{f}_k^i - \boldsymbol{f}_{k-k_{start}+1}^j||}{k_{end} - k_{start}}, \tag{3.20}$$

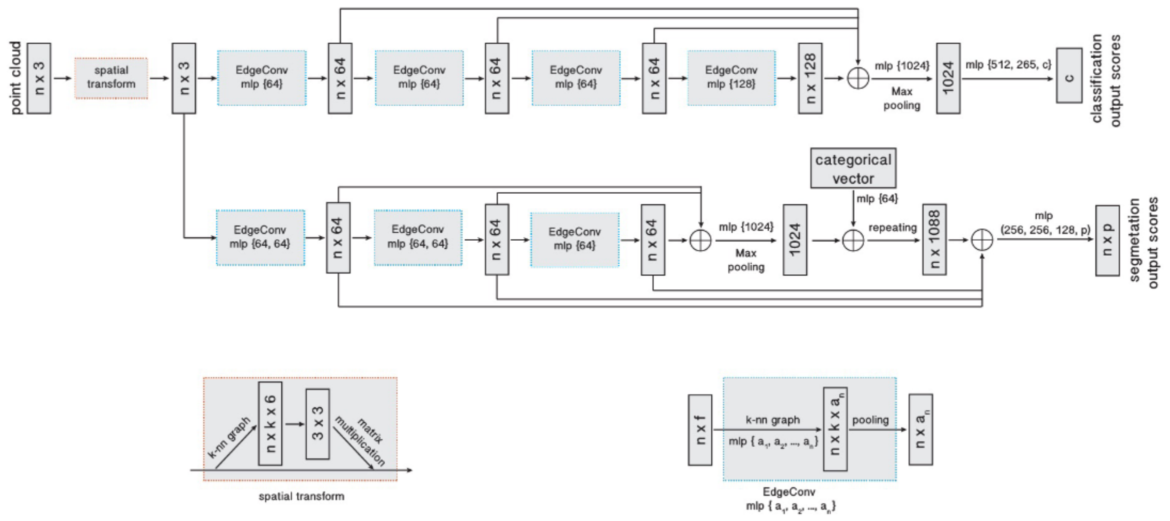


Figure 3.6: DGCNN architecture proposed by [19]. The classification contains EdgeConv layers designed for point cloud inputs. After the last layer, extracted features from each hierarchic level of the network are concatenated and pooled by maxpooling layer followed by MLP. Classification scores are obtained with softmax.

where $k_{start} = \arg\min_k t_1^j - t_k^i$, $k_{end} = \arg\min_k t_{end}^j - t_k^i$. In general, the metric measures the average distance over a temporal intersection of the trajectories. Since in the synthesized dataset the points are simulated from t_0 until t_{end} this distance can be simplified by taking $k_{start} = 0$ and $k_{end} = T$, where T is the number of simulated time steps of the moving point cloud. In the real dataset this can be enforced by only including points (cells) in the dataset, which exist over whole T frame sequence.

3.7 Deep Learning Cell Movement Classification

Both networks were trained on a simple synthetic dataset described in Sec. 3.5 for the task of movement classification. Each network was trained from scratch without any prior knowledge about pattern recognition. In the following subsections, the training parameters of both architectures are described, as well as methods for dataset processing. In addition, details about testing are provided, such as metrics used to validate predictions.

3.7.1 Training

For both architectures, each input point cloud point has coordinates $(x_{i,t}, y_{i,t}, z_{i,t}, \tau_{i,t})$, where x, y, z are the spatial coordinates of each point and τ is the time dimension, resulting in the 4 dimensional feature vector. The input of a network is a tracklet of points consisting of T time frames ($fr = T$) and N points ($pts = N$). The input batch size B differs depending on the number of time frames k and points N. It was chosen to be 4 for larger parameters or 8 for smaller ones. The resulting dimensionality of the input is therefore $B \times k \times N \times 4$.

Each generated simple point cloud consists of 2048 points, which is too large to use as a direct input combined with bigger T. Instead, a random point and its closest neighbors N are chosen as input to the network. This selection process ensures that the network receives completely new data in each iteration, helping to generalize the performance. To prepare each input, normalisation is applied along its spatial dimensions, transforming the coordinates into spherical coordinates, normalising the radius, and then converting them back to Cartesian coordinates. This helps to significantly speed up the calculations [83], reduces the impact of outliers, difference in scale, and improves the convergence of the method.

The labels of each input correspond to a movement class; however, for comparing it with the network output, a one-hot encoding representation was used.

The dataset, as discussed in Sec. 3.5, is balanced and contains 1000 labelled point clouds per class. The dataset was divided to training and validation sets, a validation subset with 33% of the entire dataset. A fixed random seed was used to ensure identical sets for all experiments.

Network Parameters of PointLSTM

The parameters of the PointLSTM network are mostly default values. Adam optimizer [84] with learning rate $lr = 0.0001$, weight decay 0.005. The number of epochs was set by default to 200. The number of neighbors grouped in each layer was set the same as in the default architecture to [16, 48, 48, 24] points depending on the layer. Other parameters of the input point cloud, such as fr and pts, are subject of ablation studies in Sec. 4.1.

The default loss function $loss$ of PointLSTM is cross-entropy Loss (L_{cel}) already combined with softmax

$$L_{cel}(\hat{y}, y) = -\sum_{i=1}^{C} y_i \cdot \log\left(\frac{\exp \hat{y}_i}{\sum_{c=1}^{C} \exp \hat{y}_c}\right), \tag{3.21}$$

where $\hat{y}$ is prediction y is ground-truth one-hot encoded label and C is number of classes.

In the multiclass classification tasks, L_{cel} is often used because it offers prediction scores for each class. The sum of all prediction scores for all classes is equal to one. For example, a prediction of [0.05, 0.05, 0.05, 0.05, 0.1, 0.1, 0.6] means that the predicted class is with 5% each of the first four ground-truth classes, with 10% each of the fifth and sixth classes, and with 60% of the seventh class. In this scenario, the most dominant class is the seventh class. Therefore, the predicted scores give the class probabilities and the most dominant class, giving different insights into which movement appears in which part of the point cloud. Visualizations of both prediction scores and dominant class are presented in the result Chapter 4.

In addition to CEL, other loss functions were tested in Sec. 4.1. Binary Cross-Entropy Loss (BCEL) is similar to CEL but instead of softmax using the sigmoid function. The difference is that class probabilities do not sum up to one, but each class can take on values between 0 and 1. The third and fourth loss functions are modified Mean Square Error (MSE) called Class MSE and Mean Absolute Error (MAE) called Class MAE. Both of them penalize only the prediction of the ground-truth class in the point cloud. So if the ground-truth label is compression, then only the predicted class probability of the compression movement is taken into the loss function. However, the other class probabilities will not be penalized. In the case of e.g. shear movement which is partly expansion and compression, this may help the network to predict the movement even if it is not specified in the ground-truth label and will not be penalized for being an incorrect prediction.

Network Parameters of DGCNN

The default parameters for the DGCNN network are the SGD optimizer $lr = 0.01$, weight decay 0.0001. The number of epochs is the same as that of PointLSTM, 200. However, the SGD turned out to be unsuitable for this task. The loss function is the default CEL. The number of neighbors grouped in layers k, as well as the number of time frames fr, the number of input points pts, and the size of the embedded dimension emb is subject of the ablation study in Sec. 4.1.

3.7.2 Testing

After every five epochs of training, the trained network was validated on validation data. The model was also saved to enable its subsequent testing. Accuracies and confusion matrices were computed during this validation.

When the whole training process is completed, the model is evaluated on the testing data. The testing datasets of simple point clouds and artificial point clouds were generated separately. Each class was simulated five times, so 35 point clouds were taken as testing data for both point cloud types. Several metrics were used to show model performance.

The already mentioned accuracy is a ratio of all correctly classified labels to the number of all labels. The confusion matrix $\boldsymbol{M}_{C\times C}$ is a table with C rows and columns, where columns are predicted labels and rows are ground-truth labels. On diagonal elements $\boldsymbol{M}[i,i]$ is the number of correctly predicted labels i while off-diagonal elements are incorrect classifications, e.g. $\boldsymbol{M}[\mathbf{1},\mathbf{5}] = 10$ means that 10 labels **1** were incorrectly predicted as label **5**. A perfect model will have only diagonal elements greater than zero and off-diagonals equal to zero.

From the confusion matrix, several other useful metrics can be extracted that are useful for model evaluation [85]. Each of these metrics can be extracted for each class, helping to describe the advantages and drawbacks of the model. Precision

$$P_i = \frac{\boldsymbol{M}[i,i]}{\sum_{j=1}^{C} \boldsymbol{M}[j,i]} \tag{3.22}$$

describes how many predictions of class i were correct among all predictions of class i (true and false positive of class i), so it overestimates class i. Recall

$$R_i = \frac{\boldsymbol{M}[i,i]}{\sum_{j=1}^{C} \boldsymbol{M}[i,j]} \tag{3.23}$$

describes how many predictions of class i were correct among all ground-truth i (true positive and false negative of class i), so it underestimates class i. The combination of recall and precision is the F1 score

$$F1_i = \frac{2\boldsymbol{M}[i,i]}{\sum_{j=1}^{C} \boldsymbol{M}[i,j] + \sum_{j=1}^{C} \boldsymbol{M}[j,i]}. \tag{3.24}$$

For the sake of completeness, the accuracy is defined using a confusion matrix as

$$A = \frac{\sum_{i=1}^{C} \boldsymbol{M}[i,i]}{\sum_{j=1}^{C}\sum_{k=1}^{C} \boldsymbol{M}[j,k]}. \tag{3.25}$$

Except for the classification metrics R, P, $F1$, and A, the regression metrics were calculated. Those include CEL, MSE, MAE, Class MSE and Class MAE. All these metrics were already used in the training, so they are described there in Sec. 3.7.1.

3.8 Visualizations

The networks were evaluated both from the numerical point of view, described in the previous section, and prediction visualizations were made to see the qualitative network performance. For this purpose, there are two approaches. The first is for simple and artificial point clouds, and the second is for real embryo development.

The first uses Matplotlib and is inspired by the visualization in the previous thesis [70]. Point clouds are plotted on a grid frame by frame using the Matplotlib animation. The second mentioned is using Paraview software [79], which can effectively plot very large point clouds with many additional features, settings and utilities, such as video extraction, simple colormap changing, and actual frame snapshots. Moreover, the interface looks more professional and the resulting figures/videos have higher visual quality.

Chapter 4

Results

This chapter provides an overview of the results of the evaluation of the PointLSTM network and the DGCNN network on all types of datasets. Firstly, there are ablation studies that were carried out to identify the optimal parameters of both networks, followed by a visual comparison on artificial point clouds. At the end of this chapter, the final test on unlabelled real embryonic data was conducted, including visualization and evaluation of network performance.

4.1 Ablation Studies

Firstly the ability of both networks to distinguish between activity and inactivity was tested. Therefore, for each pair of nomove and the remaining movement, a binary classification task was formulated. In this way, it could be determined whether the network can be used for the 3D+t point cloud classification task.

In the remaining parts of this section, ablation studies were conducted for different parameters of both networks. Both networks with varying parameters were trained on the simple point cloud training dataset and also tested on the simple point cloud testing dataset.

In the case of PointLSTM, different numbers of frames fr, number of input points pts and loss function $loss$ were examined. As baseline parameters $fr = 50$, $pts = 256$ and $loss = CEL$ were taken and in each experiment one parameter changed while others remained fixed. In this way, a collection of parameters with the best results could be determined.

For DGCNN, ablation studies on the selection of optimizers, fr, pts, the number of grouped neighbors k and the dimension size of the embedded dimension emb were carried out. As a baseline $fr = 32$, $pts = 128$, $k = 20$ and $emb = 512$ were chosen. As distance measure, baseline Euclidean distance was used. Some of them are based on original paper. Similarly as in the case of PointLSTM, in each test one parameter is varying, while the others remain fixed.

4.1.1 Single Movement Prediction

Firstly an initial proof-of-concept experiment was conducted, which included six tests, each containing data from the nomove class and one of the remaining movement classes. In this way, it was possible to identify if the network can distinguish movement from background noise and also if the time-based features are seen by the network. Each model learned on different movement classification was evaluated only by classification metrics, because the task is a binary classification.

PointLSTM

Each model of the PointLSTM network was trained for 200 epochs. Input data consisted of point clouds of 64 points over 10 time steps. The ability of the PointLSTM model to differentiate between dynamic data samples and stationary ones is evident from the results presented in Tab. 4.1. Only for shear movement, it has a very low accuracy, which corresponds to the accuracy of pure guessing. The possible reason for failure could be the time window too short to capture the development of shear movement.

Therefore, a second experiment with 32 time frames was carried out. From the results in Tab. 4.2 there is a noticeable improvement in all metrics for all movements tested. This improvement can be attributed to longer time window, which provides more information about dynamic development of the point cloud, hence the LSTM in the network architecture can extract more useful time-based features over longer time period.

DGCNN

Tab. 4.3 displays the accuracy, precision, recall and f1 scores obtained after 1000 iterations of training. The reason for the increasing number of iterations is the slower convergence rate of the network. The model was able to distinguish between movements and noise successfully, as indicated by the results. However, the precision of the shear movement was relatively low at 59.2%. This could be due to the presence of subtle shear movements in the dataset or the use of a short time window of $fr = 10$ frames.

The other movements have precisions above 90%, which is a great result. Radial movement is the most distinguishable from all, probably due to the thinning of the cell layer over time, which is very different from background noise. Nomove class remains the cell layer mostly the same thickness. Recall of all movement classification is nearly 90% for all movements, so about 10% point clouds were incorrectly classified as background noise. Since the network obtained only 10 time frames of the 64-point cloud, the Nomove class could very much resemble subtle movements over the $fr = 10$ frames.

Therefore, similar to the PointLSTM network in the previous section, the second experiment with the same $pts = 64$, but $fr = 32$ time frame point cloud was carried out. In Tab. 4.4, it is visible that, except for radial and shear movements, the movements are correctly distinguished from the inactive class. For radial and shear movement, all inputs were classified as nomove class, therefore precision, recall, and F1 score are 0. Other changes, such as changing the parameters emb and k, did not help to classify the radial and shear correctly.

4.1.2 Optimizer Selection for DGCNN

The testing of DGCNN on multiclass classification begins with determining the correct optimization settings. This test was necessary, since the first attempt to use default SGD optimizer for multi-class classification was not successfull. Therefore, various settings of optimizer, scheduler, learning rate, and distance measure were tested to find out which is the correct approach that can learn on full dataset. For the SGD optimizer (default in [19, 20], further as default) or Adam [84] was chosen, the learning rate scheduler is CosineAnnealingLR (default) or StepLR, the corresponding learning rate is 0.001 (default) or 0.0001 and the distance measure described in Sec. 3.6.2 is Euclidean (default) or ST-distance. The chosen parameter settings are listed below.

Movement	Accuracy	Precision	Recall	F1-score
Uniform	89.5%	97.8%	81.2%	88.7%
Expansion	67.6%	70.5%	62.1%	66.0%
Compression	68.5%	67.8%	72.2%	69.9%
Rotation	70.3%	70.3%	71.9%	71.1%
Radial	78.0%	84.9%	69.0%	76.1%
Shear	49.4%	50.2%	40.3%	44.7%

Table 4.1: Testing results of the PointLSTM network for all pairs of Nomove and one of other movement types for time window of 10 frames

Movement	Accuracy	Precision	Recall	F1-score
Uniform	94.5%	99.3%	89.9%	94.4%
Expansion	95.5%	99.0%	91.9%	95.4%
Compression	96.2%	96.1%	96.4%	96.3%
Rotation	80.8%	89.1%	70.7%	78.9%
Radial	98.3%	100.0%	96.7%	98.3%
Shear	71.5%	81.5%	56.7%	66.9%

Table 4.2: Testing results of the PointLSTM network for all pairs of Nomove and one of other movement types for time window of 32 frames

1. SGD, cos, 0.001, euclid
2. SGD, cos, 0.0001, euclid
3. SGD, cos, 0.001, stdist
4. SGD, cos, 0.0001, stdist
5. SGD, step, 0.001, euclid
6. SGD, step, 0.0001, euclid
7. SGD, step, 0.001, stdist
8. SGD, step, 0.0001, stdist
9. Adam, cos, 0.001, euclid
10. Adam, cos, 0.0001, euclid
11. Adam, cos, 0.001, stdist
12. Adam, cos, 0.0001, stdist
13. Adam, step, 0.001, euclid
14. Adam, step, 0.0001, euclid
15. Adam, step, 0.001, stdist
16. Adam, step, 0.0001, stdist

The other network parameters were set to default values, input frame size $fr = 32$, input number of points $pts = 128$, number of neighbors to construct the knn graph $k = 20$, embedded feature dimension $emb = 1024$, and loss is Cross-entropy loss $loss = CEL$. The number of iterations chosen for this particular test was set to 300, so that all models could converge successfully.

The results confirmed the hypothesis that SGD is not suitable for this task. For each of the tests 1.-8. the model overfitted to one class. Therefore, only tests 9.-16. were considered in the evaluation Tab. 4.5. Models with a learning rate 0.001 generally generate better results; however, if the number of iterations is increased, the 0.0001 learning rate could potentially have better results. The step scheduler in test 13. has slightly better results than cos scheduler in test 9. ST-distance was proven not to be very useful, since models trained with it had overally worse results than models with Euclidean distance. For DGCNN experiments in the following sections, the parameters of the most successful test 13. were chosen as the new default.

Note, that in Sec. 4.1.1 SGD was used as well, so possibly it was the reason for incorrect

Movement	Accuracy	Precision	Recall	F1-score
Uniform	99.5%	100.0%	99.1%	99.6%
Expansion	97.6%	98.5%	96.7%	97.6%
Compression	96.2%	94.5%	98.2%	96.3%
Rotation	91.1%	96.6%	85.4%	90.6%
Radial	100.0%	100.0%	100.0%	100.0%
Shear	59.2%	62.8%	48.4%	54.6%

Table 4.3: Testing results of the DGCNN network for all pairs of Nomove and one of other movement types for time window of 10 frames

Movement	Accuracy	Precision	Recall	F1-score
uniform	100.0%	100.0%	100.0%	100.0%
expansion	99.7%	100.0%	99.4%	99.7%
compression	99.8%	99.7%	100.0%	99.9%
rotation	98.5%	100.0%	97.0%	98.5%
radial	49.2%	0.0%	0.0%	0.0%
shear	49.2%	0.0%	0.0%	0.0%

Table 4.4: Testing results of the DGCNN network for all pairs of Nomove and one of other movement types for time window of 32 frames

activity/inactivity distinction for some classes.

4.1.3 Input Frame Size

In this experiment, the length of the time window was examined as a variable parameter with other parameters fixed. It was tested to a maximum number of 50 time frames, since more would be too much for LSTM memory and high memory consumption for DGCNN. Also they would be only suitable to capture very long-time movements in the real point cloud.

PointLSTM

In the PointLSTM architecture, the input number of points is fixed for 256 points, and loss function to cross-entropy loss. Multiple metrics were extracted for different scenarios. First, the overall classification and regression metrics were calculated across the entire test dataset with all movement classes, and then the per-class metrics were computed only for point clouds simulating the same movement. The latter mentioned is similar to the ablation study metrics in Sec. 4.1.1. The metric across the entire dataset in Tab. 4.6 shows that 32 time frames are the optimal parameter for all the metrics considered, with an accuracy of about 2% better than the default 50 time frames. The mean per-class metrics (Class MSE and Class MAE) also show a dominancy of 32 frames. For shorter time windows, there might not be enough temporal context and for longer window, the LSTM can already "forget" the past points. From the table comparing per-class metrics (Tab. B.1) it is visible that 20 time frames have the highest number of best metrics in all classes. However, in classes such as compression and shear, it performs much worse than models trained on more time frames. Therefore, 32 time frames is the best trade-off among all class metrics.

Setting	Accuracy	CEL	MSE	MAE
9.	73.9%	**0.61**	0.051	0.0951
10.	66.3%	0.82	0.0648	0.114
11.	64.7%	1.08	0.070	0.133
12.	48.3%	1.30	0.081	0.156
13.	**74.4%**	0.63	**0.051**	**0.089**
14.	64.4%	0.80	0.065	0.127
15.	58.9%	1.00	0.071	0.134
16.	56.0%	0.98	0.073	0.139

Table 4.5: Testing results of the PointLSTM network for different optimizer settings

Frames	Accuracy	CEL	MSE	MAE	Mean Class MSE	Mean Class MAE
10	65.3%	0.92	0.069	0.128	0.297	0.449
20	75.4%	0.65	0.049	0.096	0.205	0.337
32	**78.0%**	**0.57**	**0.046**	**0.084**	**0.179**	**0.293**
40	76.9%	0.62	0.047	0.092	0.196	0.321
50	76.2%	0.60	0.048	**0.084**	0.189	0.294

Table 4.6: Testing results of the PointLSTM network for variant number of time frames

DGCNN

The fixed default parameters for DGCNN frame size testing are $pts = 128$, $k = 20$, $emb = 1024$, $loss = CEL$, $opt = Adam$, $sch = step$, $lr = 0.001$, $dist = euclid$. The number of frames fr is examined in the following tests. All metrics in Tab. 4.7 are improving with increasing number of time frames. In a detailed per-class table in Tab. B.7 the trend is similar. Only for Nomove and Compression the metrics are slightly better for 32 and 40 frames.

Note that DGCNN had much worse performance for shorter time windows than PointLSTM. For $fr = 10$, it is almost three times better accuracy for PointLSTM.

4.1.4 Input Number of Points

Similarly as in the previous section, a different number of input points was examined. The other parameters remain fixes.

PointLSTM

For 512 points and default 50 time frames, it was not possible to train the network, since it was a too large amount of data to be processed at once and it did not fit to GPU memory. Therefore, 512 points were taken with only 32 time frames. From Tab. 4.8 it is clearly visible that 256 points is the optimal value. There is a significant positive trend in all metrics when the input number of points is increased. This also holds for the metrics per-class shown in Tab. B.2 as well. Only for 512 points the metrics decrease. Most notably, recalls for Shear and Radial movement decrease significantly and Compression precision is much smaller. To have a more precise view of the prediction, there is a confusion matrix in Tab. B.3 for 512 points and in Tab. B.4 for 256 points and 50 frames. It tells us that the 512-point network classifies more shear movements as compression movement. This tells a lot about the input data. It means that shear movement is very similar to compression, as well as rotation to uniform movement.

Frames	Accuracy	CEL	MSE	MAE	Mean Class MSE	Mean Class MAE
10	23.6%	1.78	0.116	0.230	0.660	0.806
20	56.8%	0.97	0.077	0.144	0.334	0.505
32	71.3%	0.67	0.056	0.111	0.227	0.390
40	73.2%	0.74	0.057	0.123	0.250	0.432
50	**80.8%**	**0.49**	**0.040**	**0.077**	**0.159**	**0.295**

Table 4.7: Testing results of the DGCNN network for variant number of time frames

From the previous experiment, the best current model has 256 points and 32 time frames. From its confusion matrix in B.5 this model achieves significantly better results for Rotation movement (13% higher than the model with 256 points and 50 time frames). Overall, the metrics are very similar, the accuracy is about 2% higher, and the MSE and MAE are nearly the same. Although the results are very similar, the model for 256 points and 32 time frames needs less time frames to learn. This reduces computational complexity and memory requirements. Also, since complex movements in a real embryo can change quickly, it has a faster adaptability to complex movement dynamics.

DGCNN

In this ablation, the number of input points pts is examined for the DGCNN network. Fixed parameters are $fr = 32$, $k = 20$, $emb = 1024$, $loss = CEL$, $opt = Adam$, $sch = step$, $lr = 0.001$, $dist = euclid$. The overall results in Tab. 4.9 show that 128 points are significantly better than taking a larger input point cloud, mostly from CEL, MSE and Class metrics. The same conclusion is from per-class metrics in Tab. B.8, where Class MSE and MAE are lowest or almost the lowest for 128 points. The precision of the Shear movement is nearly 18% lower than for 256 points, however, recall nearly 8% higher. On the other hand, the precision of expansion for 128 points is 35% higher than for 256 points, similarly to the recall being nearly 26% higher. Therefore, the best results are confirmed for $pts = 128$.

4.1.5 Nearest Neighbors

The number of nearest neighbors k used in each layer of DGCNN is tested. Fixed parameters for $fr = 32$, $pts = 128$, $emb = 1024$, $loss = CEL$, $opt = Adam$, $sch = step$, $lr = 0.001$, $dist = euclid$. The results in Tab. 4.10 show that it is better to have a greater number of nearest neighbors, however, the results are not that different compared to the increasing number of input points in Sec. 4.1.3 or Sec. 4.1.4. The best model with 48 neighbors is only 3% more accurate than the neighborhood of 20 points.

4.1.6 Embedded Dimension

The influence of the embedded dimension seem to be negligible. It is interesting that although the overall results in Tab. 4.11 are very similar, the per-class metrics in Tab. B.10 are not. For Nomove, Uniform, and Expansion, the results are significantly better for $emb = 1024$, but for the rest of movements, smaller embedded dimension is more suitable. Only Nomove class has significantly lower Class MSE, MAE and Recall, but otherwise the results are generally better for $emb = 512$.

Points	Accuracy	CEL	MSE	MAE	Mean Class MSE	Mean Class MAE
64	62.0%	0.95	0.069	0.135	0.257	0.407
128	68.8%	0.65	0.058	0.113	0.248	0.394
256	**76.2%**	**0.60**	**0.048**	0.084	**0.189**	0.294
512 (32 fr.)	73.5%	0.63	0.065	**0.081**	0.241	**0.285**

Table 4.8: Testing results of the PointLSTM network for variant number of points

Points	Accuracy	CEL	MSE	MAE	Mean Class MSE	Mean Class MAE
32	60.8%	0.89	0.070	0.143	0.314	0.500
64	65.3%	0.77	0.064	0.124	0.265	0.434
128	**71.3%**	**0.67**	**0.056**	**0.111**	**0.227**	**0.390**
256	70.2%	0.87	0.062	0.123	0.278	0.431

Table 4.9: Testing results of the DGCNN network for variant number of input points

Emb. dim.	Accuracy	CEL	MSE	MAE	Mean Class MSE	Mean Class MAE
512	70.5%	**0.67**	**0.056**	**0.106**	**0.223**	**0.369**
1024	**71.3%**	**0.67**	**0.056**	0.111	0.227	0.390

Table 4.11: Testing results of the DGCNN network for variant number of embedded dimensions

4.1.7 Loss Function

For PointLSTM, different loss functions were tested. Ten types of loss were tested: Cross-entropy loss (including softmax), Sigmoid + MSE, Sigmoid + MAE, Sigmoid + Class MSE, Sigmoid + Class MAE, Softmax + MSE, Softmax + MAE, Softmax + Class MSE, Softmax + Class MAE and Binary Cross-entropy loss (including sigmoid). In the resulting Tab. 4.12 only models that did not overfit to one class are shown. For the remaining ones, the accuracy did not cross the line of $1/7 \sim 0.14$. The best metrics are for Cross-entropy loss, however, it is accompanied with softmax, which is not the best loss for the application of movement magnitude quantification. Softmax provides class scores for all movement classes that sum up to one. Therefore, movements which resemble, e.g. convergent rotation, will be classified as either convergence or rotation. In BCE loss this behavior is overcome by using the sigmoid function instead of softmax. Also, the class loss functions should overcome this problem by only penalizing the ground-truth class prediction.

The results in Tab. 4.12 show that CEL absolutely dominated over all other loss types in all metrics. Even for class metrics, which were optimized for class loss functions, the CEL dominated. The reason is visible in Tab. B.6, where some movement classes are completely ignored by some models. Such as Nomove class for Class MAE, Uniform movement for Class MAE and partly BCEL. This indicates that models overfit to several classes while ignoring others. Therefore, they prove to be unsuitable for the movement classification task. It is possible that the negative feedback that CEL gives to incorrectly classified movement is more important for successful model training than only focussing on target class prediction and ignoring others. Due to very bad results of Class losses and BCEL, only CEL was used for DGCNN.

Neighbors	Accuracy	CEL	MSE	MAE	Mean Class MSE	Mean Class MAE
20	71.3%	0.67	0.056	0.111	0.227	0.390
32	71.5%	0.64	0.055	0.103	0.212	0.361
48	**74.1%**	**0.60**	**0.050**	**0.095**	**0.193**	**0.332**

Table 4.10: Testing results of the DGCNN network for variant number of neighbors

Loss function	Accuracy	CEL	MSE	MAE	Mean Class MSE	Mean Class MAE
CEL	**76.2%**	**0.60**	**0.048**	**0.084**	**0.189**	**0.294**
Class MSE	63.7%	1.58	0.103	0.223	0.616	0.782
Class MAE	53.5%	1.63	0.105	0.224	0.624	0.785
BCEL	60.8%	0.92	0.070	0.155	0.341	0.497

Table 4.12: Testing results of the PointLSTM network for variant loss function

4.1.8 Model with Best Parameters

For PointLSTM the parameters with the best results are $fr = 32$, $pts = 256$, $loss = CEL$ (Tab. 4.6). The optimal parameters of DGCNN were selected one by one in the previous subsections. However, when the model with optimal parameters was trained, it did not achieve better results. Its metrics are shown in 4.13 and detailed per-class metrics in B.11. The only difference between this model and so far the best performing model in Tab. 4.7 is the different number of neighbors and embedding dimension size. Moreover, in Sec. 4.1.5 it was mentioned that the performance difference between models with different number of neighbors is small, as well as the difference in *emb* in Sec. 4.1.6. The optimal parameters are therefore left according to the fifth model in Tab. 4.7: $fr = 50$, $pts = 128$, $k = 20$, $emb = 1024$.

Accuracy	CEL	MSE	MAE	Mean Class MSE	Mean Class MAE
74.1%	0.63	0.051	0.102	0.198	0.354

Table 4.13: Testing results of the DGCNN network for combined optimal parameters

4.1.9 Discussion

Both networks have shown the ability to learn the task of motion pattern recognition. PointLSTM has very stable results across all tested hyperparameters, the maximal difference in accuracy is about 11% except for different loss functions. DGCNN also has very stable results for the number of neighbors, and input number of points, but the number of time frames seems to be the most performance influencing parameter. The difference in accuracy between 10 and 50 frames is greater than 57%. This indicates that the predictive power of PointLSTM is much greater for fewer time frames. However, the performance improvement stops at 32 frames for DGCNN, and after that the network seems to "forget" the points in the past.

4.2 Testing on Artificial Point Cloud

This type of point cloud is mainly for visual evaluation and to determine how well the network reacts to sudden changes in point movement and what movements will be classified on the rim of the change in movement. In the experiments, the ground truth is not provided, but the region, where the movement appears is always well visible. Only the best performing networks from previous ablation studies were tested on this dataset. Both networks performed visually similarly. Therefore, the results of the initial three tests (Uniform, Expansion and Compression) are presented for PointLSTM with optimal parameters, while the results on the remaining tests (Rotation, Radial and Shear) are of DGCNN.

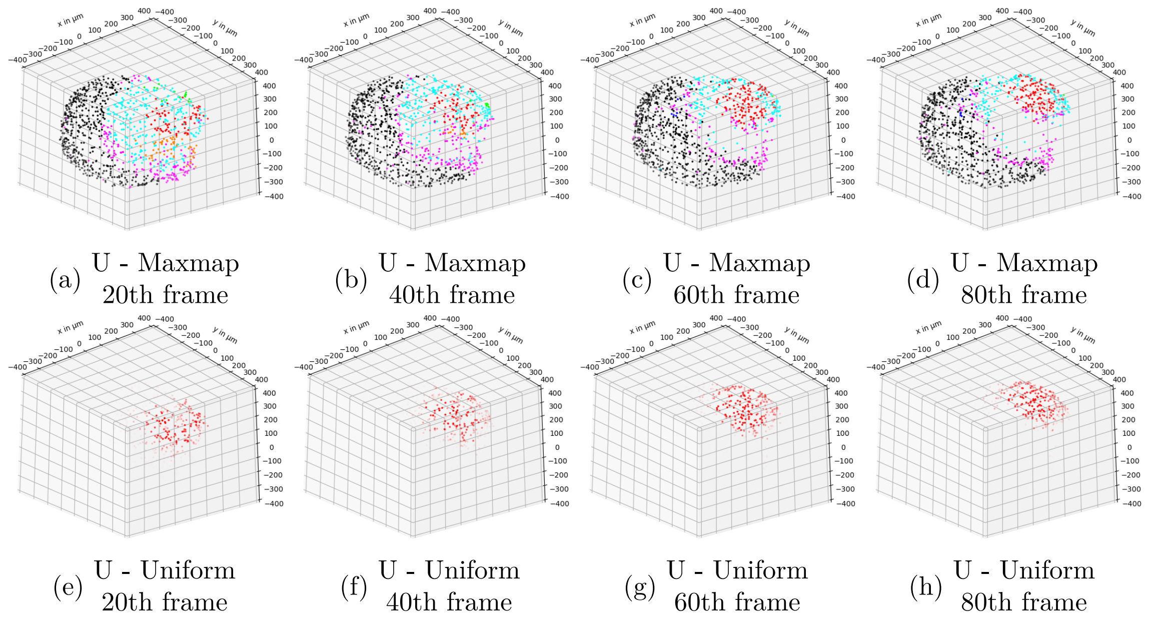
The visual results are presented first on maxmaps, where dominant movement is visible (colormaps of each movement are in each figure caption). The second type of figure denotes the magnitude of the specific movement, where the strength of movement of each cell is expressed by cell opacity. Naturally, not every movement magnitude is visualized in every tesing scenario, because some of them are not classified at all or in the case of Nomove class it is omnipresent where the simulated movement does not appear.

4.2.1 Nomove Class

Since Nomove class is present in each artificial point cloud, the results are visible in each of the following tests as background areas. The prediction of the Nomove class is mostly correct with some exceptions, such as rotation prediction with PointLSTM in Sec. 4.2.5.

4.2.2 Uniform Movement

First movement for testing is uniform. The movement is simulated from the 20th time frame and visualized every 20 frames until the 80th frame. Uniform directional movement is correctly classified, which can be seen in the dominant movement map Figs. 4.1a-d and uniform movement amplitude Figs. 4.1e-h. Expansion movement can be seen in the 20th frame in Fig. 4.1a where the moving cells "expand" away from the static ones. Its magnitude in the 20th and 40th frames is depicted in Figs. B.1a-b. Rotation is predicted in areas similar to uniform (magnitudes in Figs. B.1b-e). According to Tab. B.5 rotation is falsely predicted by PointLSTM in 16% of the cases, so its strong magnitude is understandable. The DGCNN network achieved similar visual results (data not shown), but without initial expansion and with fewer false predicted rotations.



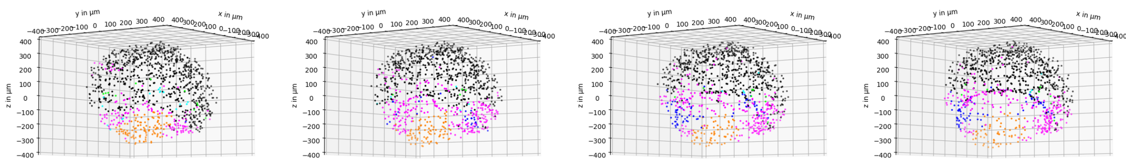
(a) U - Maxmap 20th frame
(b) U - Maxmap 40th frame
(c) U - Maxmap 60th frame
(d) U - Maxmap 80th frame
(e) U - Uniform 20th frame
(f) U - Uniform 40th frame
(g) U - Uniform 60th frame
(h) U - Uniform 80th frame

Figure 4.1: Visualization of the dominant movement mapping of the PointLSTM predictions together with the magnitude of the uniform movement.
Colormap: N - ●, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

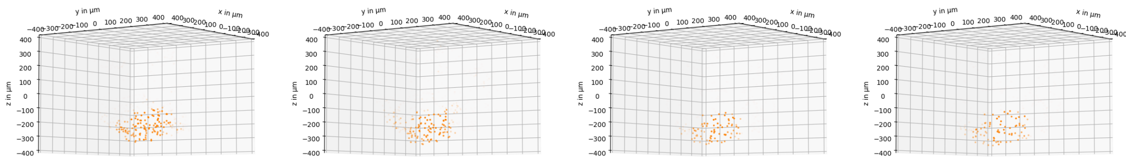
4.2.3 Expansion Movement

The expansion movement is successfully recognized in the lower part of the point cloud over all time frames (Figs. 4.2a-h). The expansion is also not present anywhere else in the rest of the point cloud (Figs. 4.2e-h), which confirms that the network correctly distinguishes between expansion and the Nomove class. Note that there is significant shear and radial movement on the rim of Nomove and expansion, which can be considered as correct behavior. Expanding cells are moving towards Nomove cells, therefore creating shear and radial motion towards the embryo boundary. Their amplitudes for all time frames can be seen in Fig. B.2. Especially shear movement is also partly present where the Nomove class is predicted. Any other movement (except for the Nomove class) is not visibly predicted.

For DGCNN the results are very similar (data not shown), only with some compression movement on the expansion/Nomove boundary.



(a) E - Maxmap 20th frame (b) E - Maxmap 40th frame (c) E - Maxmap 60th frame (d) E - Maxmap 80th frame



(e) E - Expansion 20th frame (f) E - Expansion 40th frame (g) E - Expansion 60th frame (h) E - Expansion 80th frame

Figure 4.2: Visualization of the dominant movement mapping of the PointLSTM predictions together with the magnitude of the expansion movement.
Colormap: N - ●, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

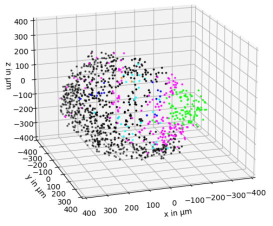
4.2.4 Compression Movement

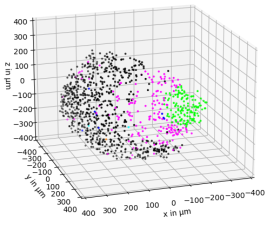
Compression is correctly predicted in the center of the compressing area; however, on the boundary there are many false predictions of rotation and shear (Figs. 4.3a-e). There is also a subtle trace of expansion movement in Fig. 4.3a. DGCNN predictions are much cleaner than the PointLSTM predictions, which means that compression has a higher amplitude in the area where shear is predicted in the PointLSTM case.

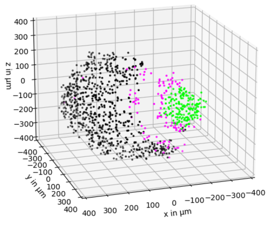
4.2.5 Rotation Movement

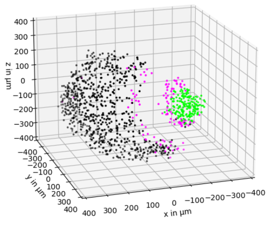
The DGCNN model provides more accurate predictions for rotation motion and in line with the expectation. The rotational magnitude is evident only in the region where the rotation was simulated, as depicted in Figs. 4.4e-h. In the dominant motion map shown in Figs. 4.4a-d, a small region at the center of the counterclockwise rotation is classified as the Nomove class. This is correct as the center remains stationary or moves very slowly during rotation. A detailed view of the Nomove class is illustrated in Figs. B.3a-d.

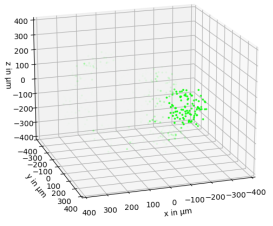
The rotation predictions from PointLSTM (data not shown) are not that clean in the area of simulated rotation. About a third of the cells are classified as inactive.

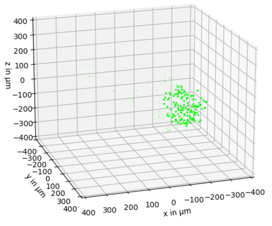

(a) C - Maxmap 20th frame

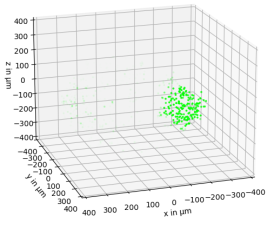

(b) C - Maxmap 40th frame


(c) C - Maxmap 60th frame


(d) C - Maxmap 80th frame


(e) C - Compression 20th frame


(f) C - Compression 40th frame


(g) C - Compression 60th frame

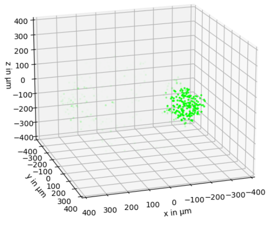
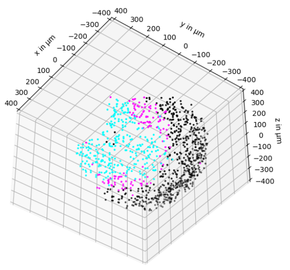
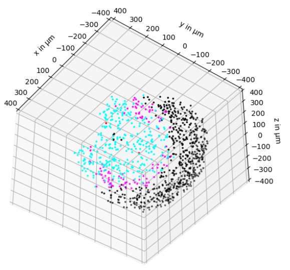
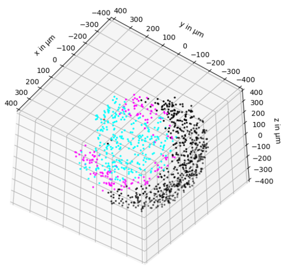
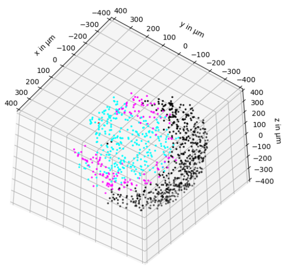

(h) C - Compression 80th frame

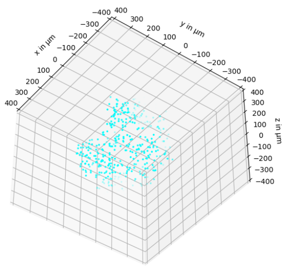
Figure 4.3: Visualization of the dominant movement mapping of the PointLSTM predictions together with the magnitude of compression movement.
Colormap: N - ●, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

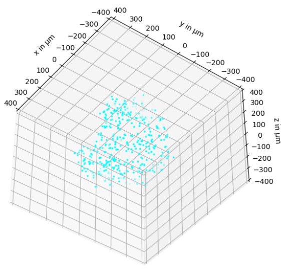

(a) Ro - Maxmap 20th frame

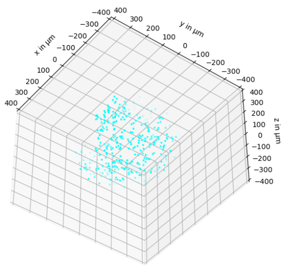

(b) Ro - Maxmap 40th frame


(c) Ro - Maxmap 60th frame


(d) Ro - Maxmap 80th frame


(e) Ro - Rotation 20th frame


(f) Ro - Rotation 40th frame


(g) Ro - Rotation 60th frame

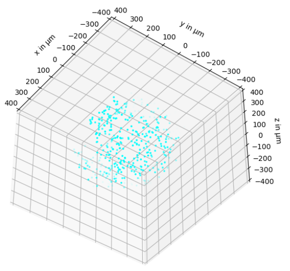
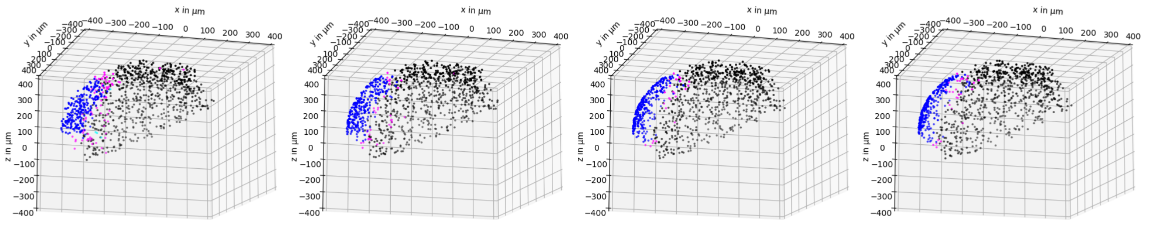

(h) Ro - Rotation 80th frame

Figure 4.4: Visualization of the dominant movement mapping of the DGCNN predictions together with the magnitude of rotation movement.
Colormap: N - ●, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

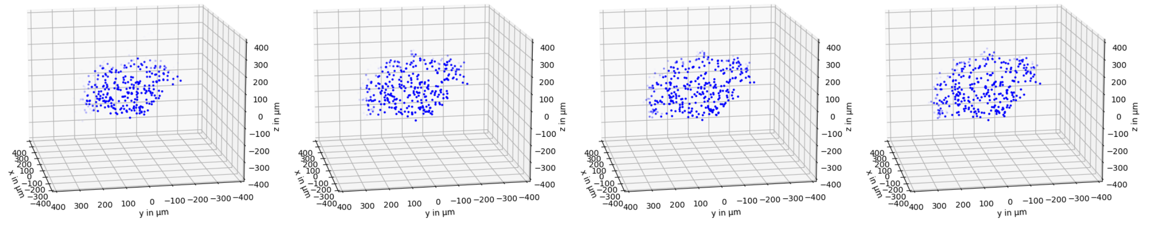
4.2.6 Radial Movement

To visualize the radial motion, a different perspective of the point cloud was selected to show the thinning of the cell layer at the embryo boundary. In Figs. 4.5a-d, it is visible that the radial motion is accurately classified within the moving region without incorrect predictions

(Figs. 4.5e-h). The PointLSTM model performs similarly. It could be expected that at least some expansion movement will be predicted, since the points are slowly expanding when moving radially. However, no sign of expansion was visible for both networks.



(a) Ra - Maxmap 20th frame (b) Ra - Maxmap 40th frame (c) Ra - Maxmap 60th frame (d) Ra - Maxmap 80th frame



(e) Ra - Radial 20th frame (f) Ra - Radial 40th frame (g) Ra - Radial 60th frame (h) Ra - Radial 80th frame

Figure 4.5: Visualization of the dominant movement mapping of the DGCNN predictions together with the magnitude of radial movement.
Colormap: N - ●, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

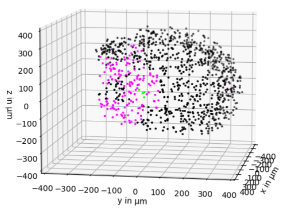
4.2.7 Shear Movement

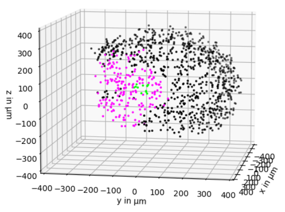
Due to the fact that shear movement involves compression in one direction and expansion in another, it is likely that both expansion and compression may be incorrectly predicted as shear. In Figs. 4.6a-d, compression was predicted in the center of the shear movement region, where some cells move closer together while others move apart and expand. However, the expansion motion has a very weak intensity. The strongest magnitude has shear motion which is in line with the expectation.

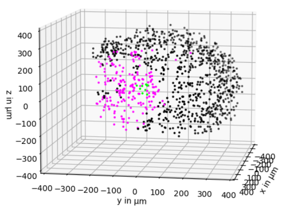
In the case of PointLSTM (data not shown), the expansion motion is predicted with minimal intensity in the "expanding" regions of the shear motion area and slight compression at the center of the shear motion.

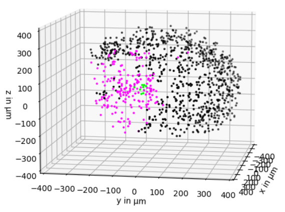
4.2.8 Discussion

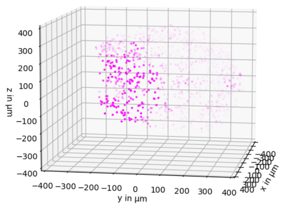
The performance of the networks on the artificial point clouds was mostly successful, since movements, even if they were changing area over time, were correctly predicted. Except for some minor mispredictions, such as rotation for uniform movement in Sec. 4.2.2 or common shear movement instead of Nomove class, the results show that both neural networks can sufficiently classify movements and determine their magnitude. There are some interesting observations during testing, such as shear movements on the rim of movement/Nomove classes, expansion at the initial frame of uniform movement in Sec. 4.2.2, inactive center of rotation in Sec. 4.2.5 or compression in the shear center in Sec. 4.2.7. These observations show that networks can further predict more complex movement configurations.

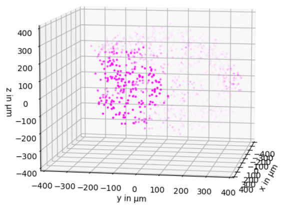

(a) S - Maxmap 20th frame

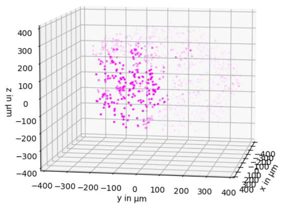

(b) S - Maxmap 40th frame


(c) S - Maxmap 60th frame


(d) S - Maxmap 80th frame


(e) S - Shear 20th frame


(f) S - Shear 40th frame


(g) S - Shear 60th frame

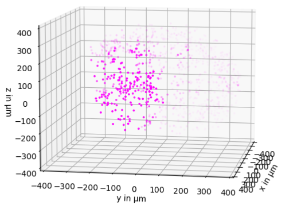

(h) S - Shear 80th frame

Figure 4.6: Visualization of the dominant movement mapping of the DGCNN predictions together with the magnitude of the shear movement.
Colormap: N - ●, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

4.3 Real Point Cloud Testing

This section provides visualization results on the real embryo point cloud of both PointLSTM and DGCNN with the best parameters determined in ablation studies in Sec. 4.1. The predictions were calculated on the real point cloud data, which was segmented into $T \times N_T$ input point clouds, each with size $pts \times fr$, where T is the number of time steps of the entire embryo development record, N_T is the number of cells in the time frame T. All these point clouds were then processed by the networks. Both maxmaps and movement magnitudes were determined for both networks. To speed up the computation and preprocessing, the predictions were calculated only every 5 frames, so $T \rightarrow \frac{T}{5}$. In the rest of this section, predictions of all movements on the whole embryo development are shown in several time steps from 6 to 10 hpf. However, the last frame is only 9.7 hpf since the network classifies inputs that span multiple frames, so the last prediction spans 9.7 to 10.0 hpf.

To compare the results with some state-of-the-art methods, the descriptors of [16] were computed to better evaluate the predictions of the networks.

4.3.1 Nomove

This "movement" or inactivity of cells should generally appear in places where the cells move only slightly or do not move at all. Both networks predicted this movement on the animal pole (PointLSTM is stronger), which is visible in Fig. 4.7. This is reasonable since on this pole there is a source of cell expansion and, therefore, cells do not move on the utmost part of the embryo. In further frames, it appears that cells do not move in the posterior area, where the tail is later formed. These areas are mostly places where the cell speed is low, as well as the rotation and compression descriptors, which are visible from Fig. 4.15.

4.3.2 Uniform

The success rate of predicting uniform movement can be validated by the speed of the cells (from Fig. 4.15), where most of the directional movement of cells occurs during the closing of

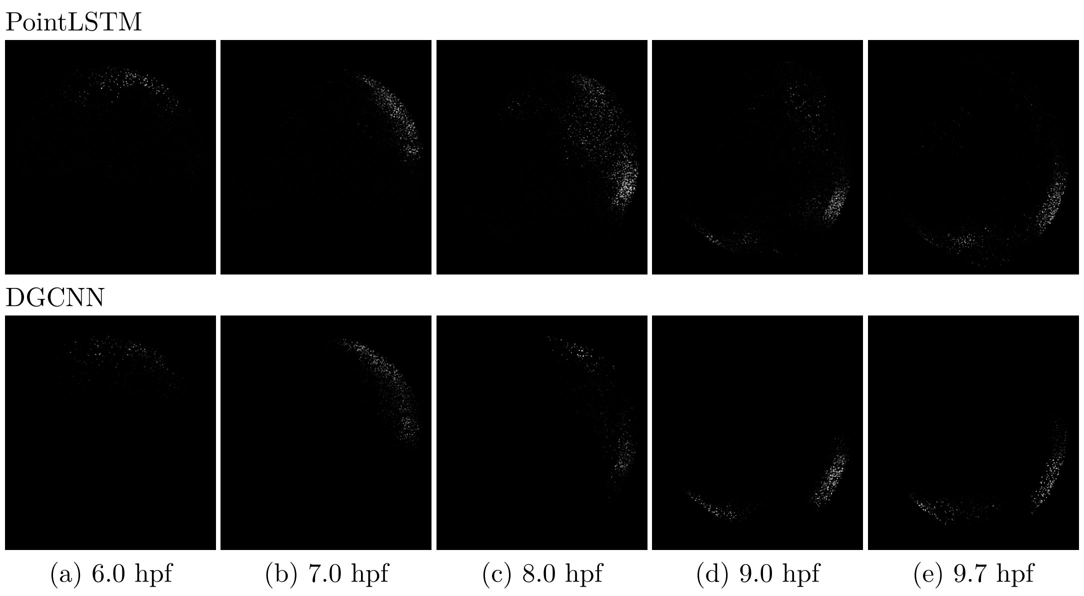


Figure 4.7: Visualization of per-class predictions of Nomove on real embryo.
Colormap: N - ○, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

the blastoderm towards the vegetal pole. This movement is also visible during the formation of the notochord on the dorsal side in 9-10 hpf, mostly in the anterior (head) area on the animal pole. This is the time of convergent extension. The predictions of both networks correctly predict the spread of the blastoderm toward the vegetal pole (Fig. 4.8), especially at 7-8 hpf there could be visible massive uniform movement near the vegetal pole, which could sign the involution movement of the cells. From 8-10 hpf, uniform movement appears mainly on the dorsal side, where the notochord forms. Stronger predictions are from the PointLSTM network.

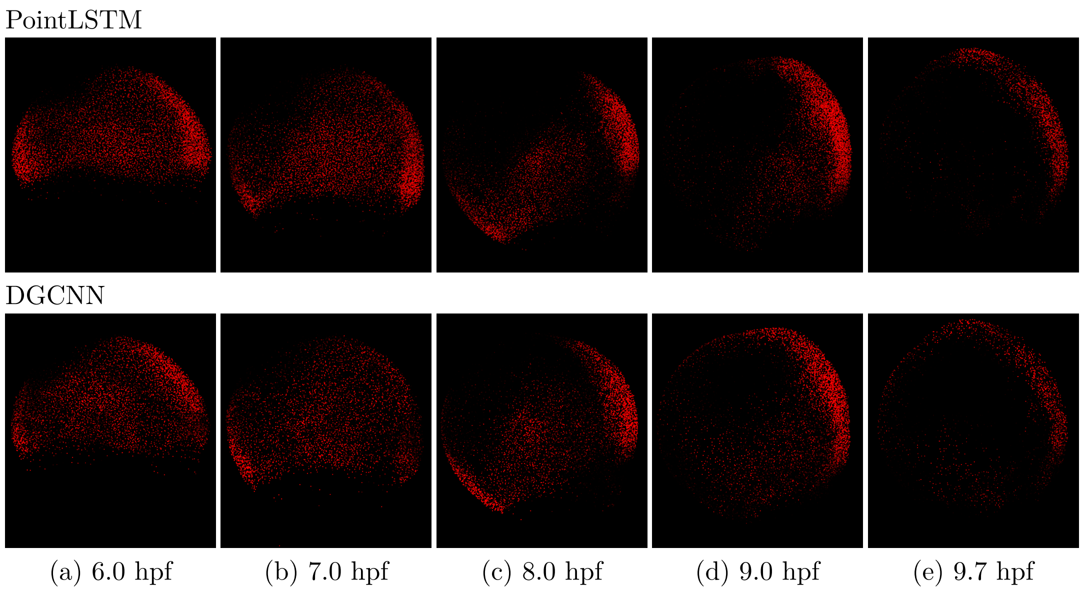
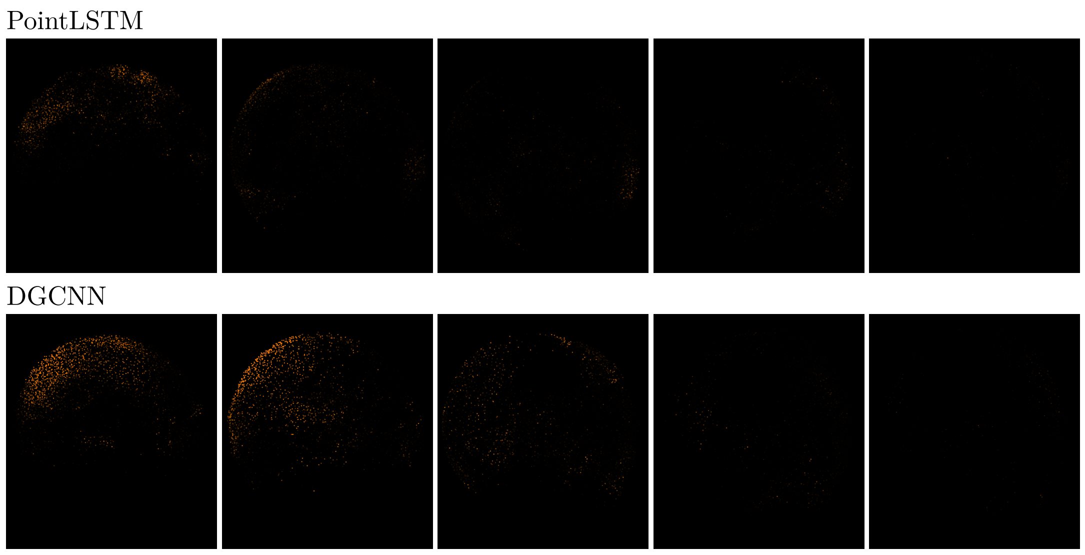


Figure 4.8: Visualization of per-class predictions of Uniform on real embryo.
Colormap: N - ○, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

4.3.3 Expansion

Expansion can be validated by P in Fig. 4.15 (marked red) and tau (blue and green). Expansion predictions are better predicted by the DGCNN network at 6-7 hpf near the animal pole. This area corresponds to the green tau in Figs. 4.15a-b. Most of other expansion predictions are scattered over all of the embryo cells, visible from Fig. 4.9, which is also correct, since epiboly is both a uniform directional movement and an expansion towards the vegetal pole. Some expansion predictions are also on the dorsal side in 7-9 hpf, where the convergent extension movement starts to form the notochord along the antero-posterior axis.



(a) 6.0 hpf (b) 7.0 hpf (c) 8.0 hpf (d) 9.0 hpf (e) 9.7 hpf

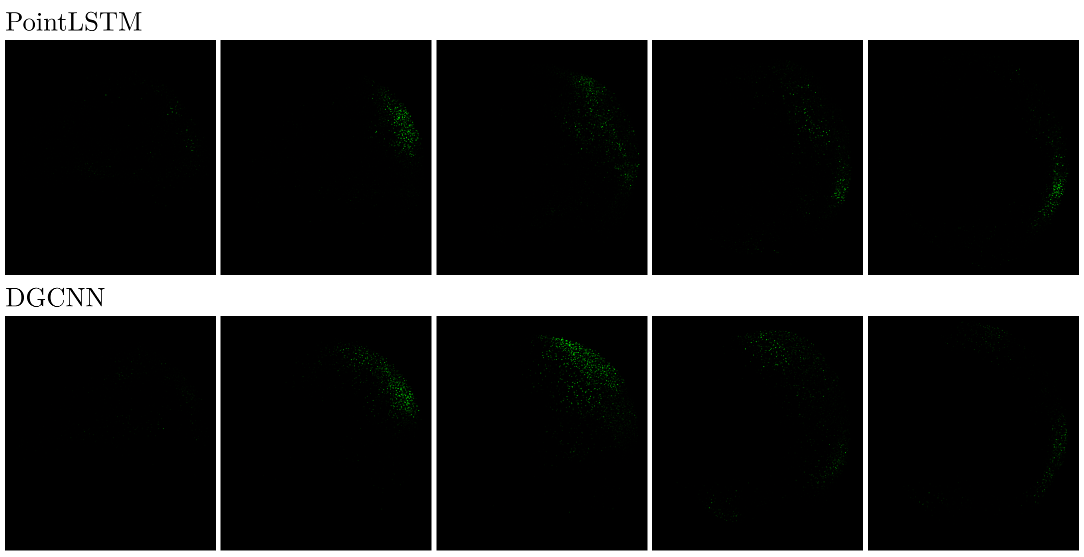
Figure 4.9: Visualization of per-class predictions of Expansion on real embryo.
Colormap: N - ○, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

4.3.4 Compression

Similarly P in Fig. 4.15 signs compression (deeper blue) and tau (yellow and red). The only predicted compressions that can be seen are along the antero-posterior axis in 7-10 hpf, where cells converge to form the notochord. This is also evident from the increasing density of cells in that area (Fig. 4.15). Higher prediction magnitude can be visible for the PointLSTM network depicted in Fig. 4.10. The limited strength might be due to the fact that it is a convergent extension on this axis, so other movements such as expansion and shear are also present in that area. Some compression is also visible at places of increased cell density.

4.3.5 Rotation

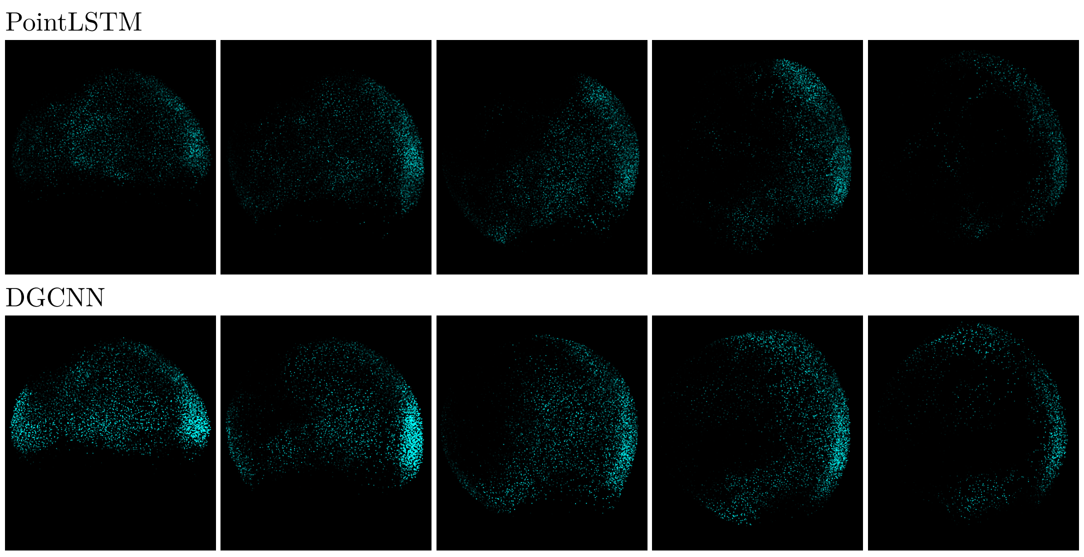
Rotation is very hard to validate, since tau (blue and yellow) only tells about the presence of rotation but not about its amplitude. It covers mostly the whole embryo, probably because there is always some local cell rotation present. The predicted rotation in Fig. 4.11 is almost all over the embryo without the ventral area at 8-10 hpf for both networks. This area has a very low density of cells, which are mostly moving radially, so it is correct. A dense area of rotation is again on the dorsal side of the embryo, where cells converge and mix together. Rotation strength is higher for DGCNN, which neccessarily does not have to be a good result. Since rotational descriptors are not available for the embryo, the only other possibility is to



(a) 6.0 hpf (b) 7.0 hpf (c) 8.0 hpf (d) 9.0 hpf (e) 9.7 hpf

Figure 4.10: Visualization of per-class predictions of Compression on real embryo. Colormap: N - ○, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

compare the results with the related literature of [16]. From Fig. 2.6, even if the embryo is different and the view is from the anterior side, the rotations in the rotational descriptor D are also inconclusive until 9 hpf. From this time on, they appear mostly along the antero-posterior axis, which is visible in network prediction of both PointLSTM and DGCNN. However, further analysis can not be suffitiently conducted.



(a) 6.0 hpf (b) 7.0 hpf (c) 8.0 hpf (d) 9.0 hpf (e) 9.7 hpf

Figure 4.11: Visualization of per-class predictions of Rotation on real embryo. Colormap: N - ○, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

4.3.6 Radial

Radial movement is very interesting, since it defines the shape changes of the embryo. However, there is no descriptor from [16] that can validate the predictions. It can be observed that the cell layer on the ventral side is thinning and, accordingly, the predictions of both networks correctly classify this as radial movement (Fig. 4.12). However, in other areas of the embryo, where the cell density decreases, such as along the dorso-ventral axis, the predicted radial movement has also a reasonable magnitude. At 8-9 hpf, mostly for PointLSTM, dense radial movement is predicted near the animal pole and the vegetal pole. At the animal pole, there is evident thickening of the cell layer (visible in Figs. 4.15d-e) which later forms the head of the zebrafish. At the vegetal pole, this thickening later forms the tail.

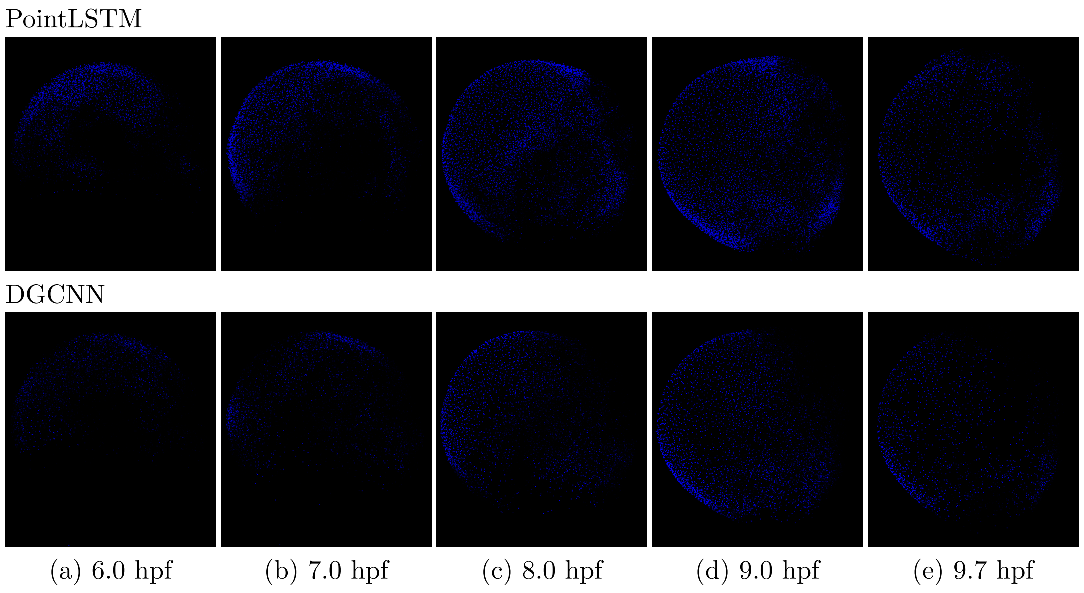


Figure 4.12: Visualization of per-class predictions of Radial on real embryo.
Colormap: N - ○, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

4.3.7 Shear

The shear movement was supposed to be similar to the distortion rate Q_d in Fig. 4.15, however, Q_d mostly indicates shear along the midline throughout the depth of tissues resulting from the relative movements of the hypoblast and epiblast [16]. Therefore, it is observed mainly during involution at 6-7 hpf and at the animal pole at 9-10 hpf. The shear movement considered in this thesis is mostly a combination of expansion and compression, so it is more similar to convergent extension than to involution. From the predictions in Fig. 4.13 it is clearly visible that it is concentrated on the antero-posterior axis throughout the embryo development.

4.3.8 Discussion

The predictions of both networks on the real point cloud show their ability to classify complex movements. The results are not perfect, since, e.g. more compression could be expected on the antro-posterior axis while less focus on rotation in the same area. Otherwise, there are many positive results that correspond to local speed and density, and to the descriptors from [16] visualized in Fig. 4.15. Uniform movement of the closing blastoderm in Sec. 4.3.2, expansion near the animal pole in Sec. 4.3.3 at 6-7 hpf, compression at the dorsal side during notochord formation in Sec. 4.3.4, thinning of the cell layer at ventral side in Sec. 4.3.6 and increased amount of shear movement on the antero-posterior axis in Sec. 4.3.7 are valuable observations which prove that the classification task is successful.

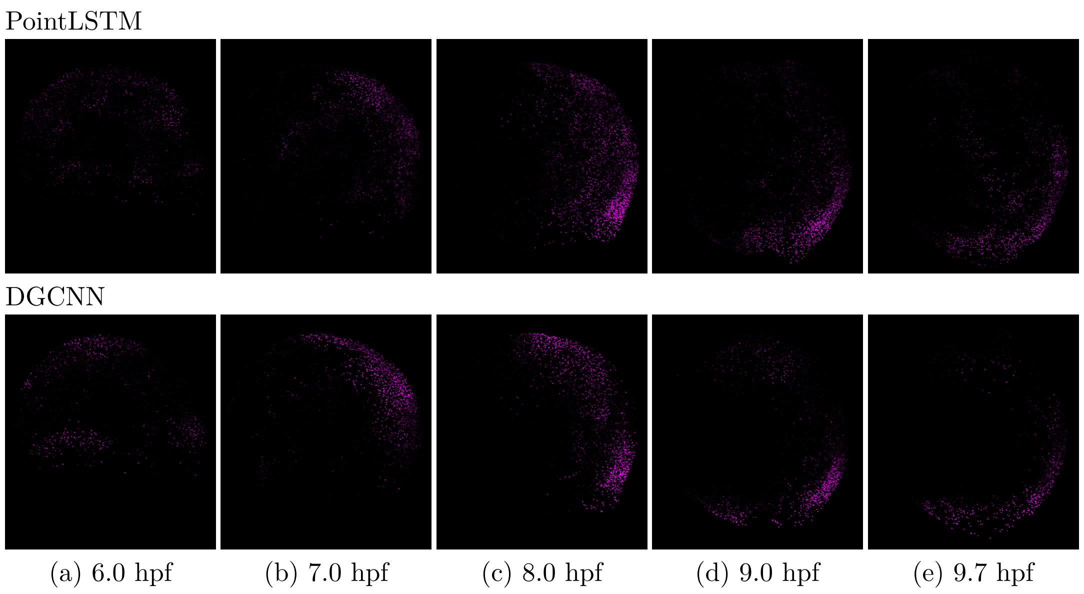


Figure 4.13: Visualization of per-class predictions of Shear on real embryo.
Colormap: N - ○, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

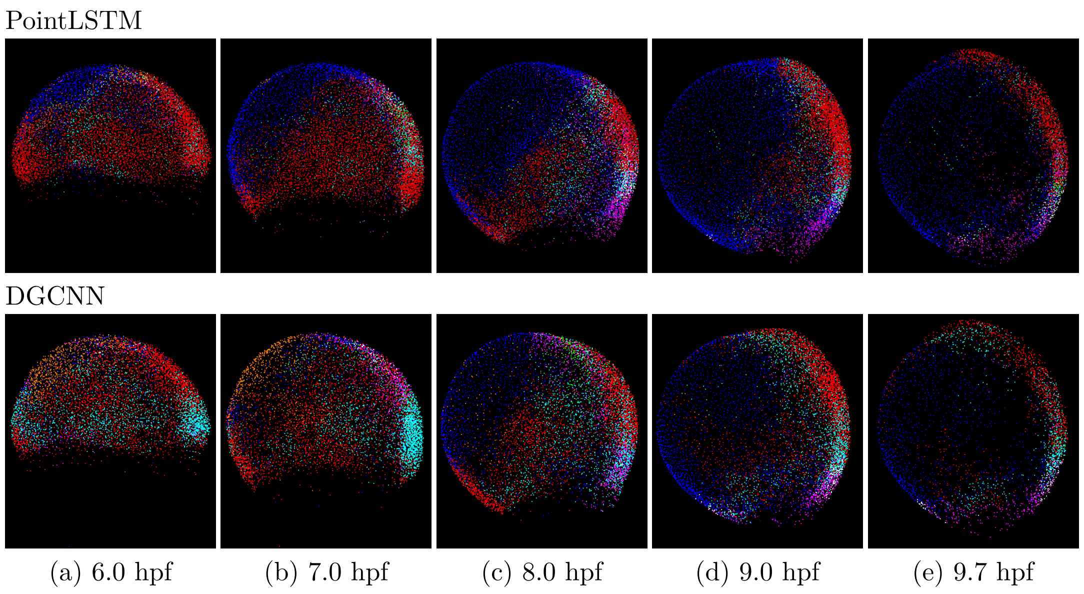


Figure 4.14: Visualization of maxmap of PointLSTM and DGCNN of real embryo.
Colormap: N - ○, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

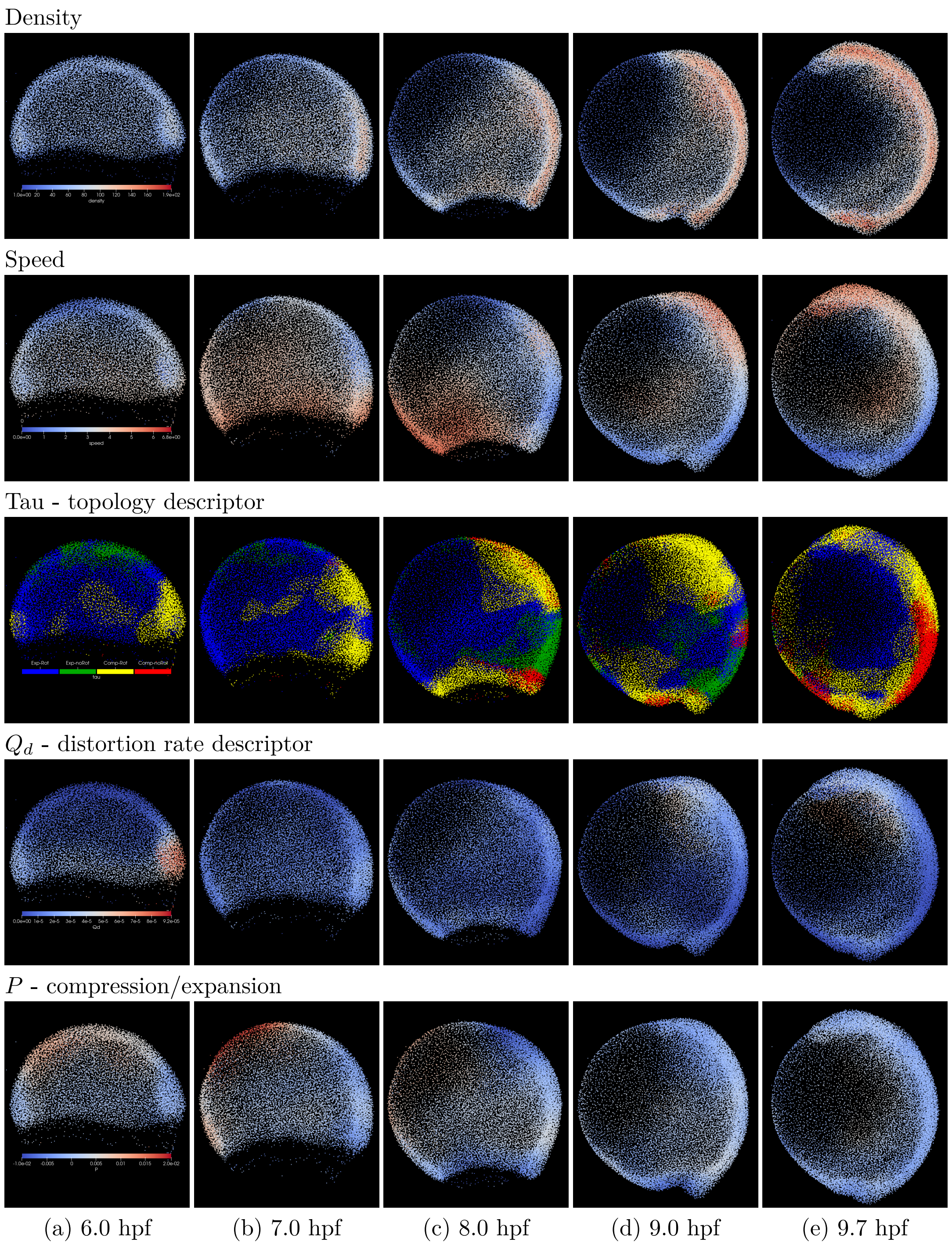


Figure 4.15: Visualization of kinematic descriptors on real embryo based on [16]. Colormap of tau: Expansion-Rotation - ●, Expansion-noRotation - ●, Compression-Rotation - ●, Compression-noRotation - ●

Chapter 5
Conclusion

The goal of this thesis was to develop deep learning models that can recognize spatio-temporal patterns in 3D+t point clouds of zebrafish gastrulation. Due to the fact that the real point clouds are not annotated, it was necessary to create a synthetic dataset to train the neural network. For this purpose, two different types of point clouds were simulated.

A simple simulation was used for training, validation, and testing where the entire point cloud consists of random points that are part of a spherical shell. All points simulate one of seven predefined movements, including an inactive nomove class. The second type is the artificial point cloud, which is created similarly, but only a dynamically changing group of points in time is simulating the movement. The latter is used only for testing.

The ability of both networks, namely PointLSTM and DGCNN, to correctly distinguish movement from inactivity is first tested by point-wise binary classification between all pairs of movement and nomove. Since the experiments were successful, the networks could be further used for multi-class classification and movement magnitude prediction. The parameters of both networks were found to optimize their performance in multiple ablation studies. The resulting architectures were tested on all types of point clouds: simple, artificial, and real.

To correctly assess the network performance, both classification and regression metrics were calculated. For a simple point cloud type, DGCNN achieved better classification and regression results than PointLSTM with respective accuracies 80.8% and 78.0%. The visual comparison was then conducted on the artificial point clouds. The visual results were similar for both networks; therefore, in each experiment, the predictions of one single network are visualized. Overall, the predictions were in line with expectations. Moreover, many interesting observations were noted, such as the inactive center of rotation during rotation movement simulation or shear movement on the border of moving and inactive cells.

Since the results were successful for both architectures, they were both further applied for movement classification and its magnitude prediction on real point clouds. Due to not having labels of the real measurements, the performance is compared only by visually. For each movement type, predictions are assessed and reasoned based on knowledge about movement patterns from the developmental biology related literature. As for artificial point clouds, multiple interesting observations comparable to real events in embryo development were noted, such as uniform movements during epiboly on the blastoderm edge, expansion movement on the animal pole in early embryo stages, compression and shear movement on the antero-posterior axis, or radial movements on the ventral side. Therefore, the goal of the thesis to recognize spatio-temporal patterns in real zebrafish embryos was successful.

Possible extensions of the deep learning approach can be divided into several ideas. The first is faster and memory-efficient preprocessing of point clouds. To obtain predictions on real data it was necessary to separate the entire 3D+t point cloud into $T \times N_T$ point cloud sequences of size $pts \times fr$. Due to the fact that this segmentation was done before the prediction, it required a large amount of memory to store the preprocessed real point cloud.

Second, the inability of the PointLSTM to improve its performance on longer time windows. This issue is probably due to the LSTM layer already forgetting the points from the previous frames. The solution can be to use an attention-based architecture, which can also capture long-term relationships and may be superior to LSTM. It is also debatable whether longer time windows are really necessary, since the movements in real embryo scenarios are not as long.

The third is the opposite problem to the second one. Real embryo movements can be much faster than 32 or 50 frames, and therefore it would be better to predict movements from shorter time periods to successfully detect these patterns. PointLSTM has shown solid results for short time windows; however, DGCNN is not effective for short-time movements at all. Therefore, further research can be directed to networks containing recurrent layers or previously mentioned attention mechanisms.

Lastly, the type of movements on which the networks are trained. The networks are trained on seven movements, which is a reasonable amount; however, the training dataset could contain combinations of several movements, e.g. a directionally moving convergent rotation. This could increase the prediction ability of networks for more complex movements.

In conclusion, this work was the first successful attempt of movement pattern recognition in digital embryos with deep learning methods and opens many possibilities for further research.

Bibliography

[1] Rachel M. Warga and Charles B. Kimmel, “Cell movements during epiboly and gastrulation in zebrafish,” *Development*, vol. 108, pp. 569–580, Apr. 1990.

[2] S. E. Lepage and A. E. E. Bruce, “Zebrafish epiboly: mechanics and mechanisms,” *The International Journal of Developmental Biology*, vol. 54, pp. 1213–1228, June 2010. Number: 8-9 Publisher: UPV/EHU Press.

[3] T. Lecuit and L. Le Goff, “Orchestrating size and shape during morphogenesis.,” *Nature*, vol. 450, pp. 189–192, 2007.

[4] A. Aman and T. Piotrowski, “Cell migration during morphogenesis,” *Developmental Biology*, vol. 341, pp. 20–33, May 2010.

[5] C.-P. Heisenberg and L. Solnica-Krezel, “Back and forth between cell fate specification and movement during vertebrate gastrulation,” *Current Opinion in Genetics & Development*, vol. 18, pp. 311–316, Aug. 2008.

[6] L. Kodjabachian, I. B. Dawid, and R. Toyama, “Gastrulation in Zebrafish: What Mutants Teach Us,” *Developmental Biology*, vol. 213, pp. 231–245, Sept. 1999.

[7] C. B. Kimmel, W. W. Ballard, S. R. Kimmel, B. Ullmann, and T. F. Schilling, “Stages of embryonic development of the zebrafish,” *Developmental Dynamics*, vol. 203, pp. 253–310, July 1995.

[8] L. Wolpert, *Principles of development.* Oxford, United Kingdom ; New York, NY: Oxford University Press, sixth edition ed., 2019.

[9] C. B. Kimmel, R. M. Warga, and T. F. Schilling, “Origin and organization of the zebrafish fate map,” *Development*, vol. 108, pp. 581–594, Apr. 1990.

[10] S. J. Arnold and E. J. Robertson, “Making a commitment: cell lineage allocation and axis patterning in the early mouse embryo,” *Nature Reviews Molecular Cell Biology*, vol. 10, pp. 91–103, Feb. 2009. Number: 2 Publisher: Nature Publishing Group.

[11] P. J. Keller, A. D. Schmidt, J. Wittbrodt, and E. H. Stelzer, “Reconstruction of Zebrafish Early Embryonic Development by Scanned Light Sheet Microscopy,” *Science*, vol. 322, pp. 1065–1069, Nov. 2008.

[12] F. Amat, W. Lemon, D. P. Mossing, K. McDole, Y. Wan, K. Branson, E. W. Myers, and P. J. Keller, “Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data,” *Nature Methods*, vol. 11, pp. 951–958, Sept. 2014. Number: 9 Publisher: Nature Publishing Group.

[13] J. Stegmaier, F. Amat, W. C. Lemon, K. McDole, Y. Wan, G. Teodoro, R. Mikut, and P. Keller, “Real-Time Three-Dimensional Cell Segmentation in Large-Scale Microscopy Data of Developing Embryos,” *Developmental Cell*, vol. 36, pp. 225–240, Jan. 2016.

[14] B. Schott, M. Traub, C. Schlagenhauf, M. Takamiya, T. Antritter, A. Bartschat, K. Löffler, D. Blessing, J. C. Otte, A. Y. Kobitski, G. U. Nienhaus, U. Strähle, R. Mikut, and J. Stegmaier, “EmbryoMiner: A new framework for interactive knowledge discovery in large-scale cell tracking data of developing embryos,” *PLOS Computational Biology*, vol. 14, p. e1006128, Apr. 2018.

[15] C. Wolff, J.-Y. Tinevez, T. Pietzsch, E. Stamataki, B. Harich, L. Guignard, S. Preibisch, S. Shorte, P. J. Keller, P. Tomancak, and A. Pavlopoulos, “Multi-view light-sheet imaging and tracking with the MaMuT software reveals the cell lineage of a direct developing arthropod limb,” *eLife*, vol. 7, p. e34410, Mar. 2018.

[16] D. Pastor-Escuredo, B. Lombardot, T. Savy, A. Boyreau, J. M. Goicolea, A. Santos, P. Bourgine, J. C. del Álamo, N. Peyriéras, and M. J. Ledesma-Carbayo, “Kinematic analysis of cell lineage reveals coherent and robust mechanical deformation patterns in zebrafish gastrulation,” *bioRxiv*, p. 054353, 2016.

[17] G. B. Blanchard, A. J. Kabla, N. L. Schultz, L. C. Butler, B. Sanson, N. Gorfinkiel, L. Mahadevan, and R. J. Adams, “Tissue tectonics: morphogenetic strain rates, cell shape change and intercalation,” *Nature Methods*, vol. 6, pp. 458–464, June 2009. Number: 6 Publisher: Nature Publishing Group.

[18] Y. Min, Y. Zhang, X. Chai, and X. Chen, “An Efficient PointLSTM for Point Clouds Based Gesture Recognition,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Seattle, WA, USA), pp. 5760–5769, IEEE, June 2020.

[19] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic Graph CNN for Learning on Point Clouds,” June 2019. arXiv:1801.07829.

[20] J. Chen, J. Meng, X. Wang, and J. Yuan, “Dynamic Graph CNN for Event-Camera Based Gesture Recognition,” in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, (Seville, Spain), pp. 1–5, IEEE, Oct. 2020.

[21] M. Westerfield, *The zebrafish book. A guide for the laboratory use of zebrafish (Danio rerio)*. Univ. of Oregon Press, Eugene, 4th ed. ed., 2000.

[22] S. F. Gilbert and M. J. F. Barresi, *Developmental biology*. New York, NY: Oxford University Press, twelfth edition ed., 2020. OCLC: 1088671593.

[23] A. Y. Kobitski, J. C. Otte, M. Takamiya, B. Schäfer, J. Mertes, J. Stegmaier, S. Rastegar, F. Rindone, V. Hartmann, R. Stotzka, A. García, J. Van Wezel, R. Mikut, U. Strähle, and G. U. Nienhaus, “An ensemble-averaged, cell density-based digital model of zebrafish embryo development derived from light-sheet microscopy data with single-cell resolution,” *Scientific Reports*, vol. 5, p. 8601, Feb. 2015.

[24] N. H. Hart and M. Donovan, “Fine structure of the chorion and site of sperm entry in the egg of Brachydanio,” *Journal of Experimental Zoology*, vol. 227, no. 2, pp. 277–296, 1983.

[25] H. W. Beams and R. G. Kessel, “Cytokinesis: A Comparative Study of Cytoplasmic Division in Animal Cells: Views regarding the mechanism of cytokinesis are presented

together with scanning and transmission electron microscope studies on the contraction ring, cell cortex, and membrane specializations associated with the cleavage furrow," *American Scientist*, vol. 64, no. 3, pp. 279–290, 1976. Publisher: Sigma Xi, The Scientific Research Society.

[26] L. Carvalho and C.-P. Heisenberg, "The yolk syncytial layer in early zebrafish development," *Trends in Cell Biology*, vol. 20, pp. 586–592, Oct. 2010.

[27] C. B. Kimmel and R. M. Warga, "Indeterminate cell lineage of the zebrafish embryo," *Developmental Biology*, vol. 124, pp. 269–280, Nov. 1987.

[28] R. M. Warga and C. Nüsslein-Volhard, "Origin and development of the zebrafish endoderm," *Development*, vol. 126, pp. 827–838, Feb. 1999.

[29] M. Behrndt, G. Salbreux, P. Campinho, R. Hauschild, F. Oswald, J. Roensch, S. Grill, and C.-P. Heisenberg, "Forces Driving Epithelial Spreading in Zebrafish Gastrulation," *Science (New York, N.Y.)*, vol. 338, pp. 257–60, Oct. 2012.

[30] R. Bensch, S. Song, O. Ronneberger, and W. Driever, "Non-directional radial intercalation dominates deep cell behavior during zebrafish epiboly," *Biology Open*, vol. 2, pp. 845–854, Aug. 2013.

[31] R. Keller, L. Davidson, A. Edlund, T. Elul, M. Ezin, D. Shook, and P. Skoglund, "Mechanisms of convergence and extension by cell intercalation.," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 355, pp. 897–922, July 2000.

[32] A. Shindo, "Models of convergent extension during morphogenesis," *WIREs Developmental Biology*, vol. 7, no. 1, p. e293, 2018.

[33] M. Traub and J. Stegmaier, "Towards Automatic Embryo Staging in 3D+t Microscopy Images Using Convolutional Neural Networks and PointNets," in *Simulation and Synthesis in Medical Imaging* (N. Burgos, D. Svoboda, J. M. Wolterink, and C. Zhao, eds.), vol. 12417, pp. 153–163, Cham: Springer International Publishing, 2020. Series Title: Lecture Notes in Computer Science.

[34] L. Guignard, C. Godin, U.-M. Fiuza, L. Hufnagel, P. Lemaire, and G. Malandain, "Spatio-temporal registration of embryo images," in *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, (Beijing, China), pp. 778–781, IEEE, Apr. 2014.

[35] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *International workshop on automatic face and gesture recognition*, vol. 12, pp. 296–301, Citeseer, 1995.

[36] A. Kuznetsova, L. Leal-Taixe, and B. Rosenhahn, "Real-Time Sign Language Recognition Using a Consumer Depth Camera," in *2013 IEEE International Conference on Computer Vision Workshops*, (Sydney, Australia), pp. 83–90, IEEE, Dec. 2013.

[37] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep Learning for 3D Point Clouds: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, pp. 4338–4364, Dec. 2021.

[38] P. Villoutreix, "What machine learning can do for developmental biology," *Development*, vol. 148, p. dev188474, Jan. 2021.

[39] A. Hallou, H. G. Yevick, B. Dumitrascu, and V. Uhlmann, "Deep learning for bioimage analysis in developmental biology," *Development*, vol. 148, p. dev199616, Sept. 2021.

[40] N. F. Greenwald, G. Miller, E. Moen, A. Kong, A. Kagel, T. Dougherty, C. C. Fullaway, B. J. McIntosh, K. X. Leow, M. S. Schwartz, C. Pavelchek, S. Cui, I. Camplisson, O. Bar-Tal, J. Singh, M. Fong, G. Chaudhry, Z. Abraham, J. Moseley, S. Warshawsky, E. Soon, S. Greenbaum, T. Risom, T. Hollmann, S. C. Bendall, L. Keren, W. Graf, M. Angelo, and D. Van Valen, "Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning," *Nature Biotechnology*, vol. 40, pp. 555–565, Apr. 2022. Number: 4 Publisher: Nature Publishing Group.

[41] J. C. Caicedo, J. Roth, A. Goodman, T. Becker, K. W. Karhohs, M. Broisin, C. Molnar, C. McQuin, S. Singh, F. J. Theis, and A. E. Carpenter, "Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images," *Cytometry. Part A: The Journal of the International Society for Analytical Cytology*, vol. 95, pp. 952–965, Sept. 2019.

[42] T. He, H. Mao, J. Guo, and Z. Yi, "Cell tracking using deep neural networks with multi-task learning," *Image and Vision Computing*, vol. 60, pp. 142–153, Apr. 2017.

[43] K. Liu, H. Qiao, J. Wu, H. Wang, L. Fang, and Q. Dai, "Fast 3D cell tracking with wide-field fluorescence microscopy through deep learning," May 2018. arXiv:1805.05139.

[44] J. Cao, G. Guan, V. W. S. Ho, M.-K. Wong, L.-Y. Chan, C. Tang, Z. Zhao, and H. Yan, "Establishment of a morphological atlas of the Caenorhabditis elegans embryo using deep-learning-based 4D segmentation," *Nature Communications*, vol. 11, p. 6254, Dec. 2020. Number: 1 Publisher: Nature Publishing Group.

[45] M. Weigert, U. Schmidt, T. Boothe, A. Müller, A. Dibrov, A. Jain, B. Wilhelm, D. Schmidt, C. Broaddus, S. Culley, M. Rocha-Martins, F. Segovia-Miranda, C. Norden, R. Henriques, M. Zerial, M. Solimena, J. Rink, P. Tomancak, L. Royer, F. Jug, and E. W. Myers, "Content-aware image restoration: pushing the limits of fluorescence microscopy," *Nature Methods*, vol. 15, pp. 1090–1097, Dec. 2018. Number: 12 Publisher: Nature Publishing Group.

[46] R. A. Jones, M. J. Renshaw, and D. J. Barry, "Automated staging of zebrafish embryos with deep learning," *Life Science Alliance*, vol. 7, Jan. 2024. Publisher: Life Science Alliance Section: Methods.

[47] J.-Y. Sul, C.-w. K. Wu, F. Zeng, J. Jochems, M. T. Lee, T. K. Kim, T. Peritz, P. Buckley, D. J. Cappelleri, M. Maronski, M. Kim, V. Kumar, D. Meaney, J. Kim, and J. Eberwine, "Transcriptome transfer produces a predictable cellular phenotype," *Proceedings of the National Academy of Sciences*, vol. 106, pp. 7624–7629, May 2009. Publisher: Proceedings of the National Academy of Sciences.

[48] W. Liu, J. Sun, W. Li, T. Hu, and P. Wang, "Deep Learning on Point Clouds and Its Application: A Survey," *Sensors*, vol. 19, p. 4188, Jan. 2019. Number: 19 Publisher: Multidisciplinary Digital Publishing Institute.

[49] X. Wei, R. Yu, and J. Sun, "View-GCN: View-Based Graph Convolutional Network for 3D Shape Analysis," pp. 1850–1859, 2020.

[50] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Hamburg, Germany), pp. 922–928, IEEE, Sept. 2015.

[51] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning Deep 3D Representations at High Resolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Honolulu, HI), pp. 6620–6629, IEEE, July 2017.

[52] J. Sauder and B. Sievers, "Self-Supervised Deep Learning on Point Clouds by Reconstructing Space," in *Advances in Neural Information Processing Systems*, vol. 32, pp. 12962–12972, Curran Associates, Inc., 2019.

[53] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," 2016. Publisher: arXiv Version Number: 2.

[54] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," June 2017. arXiv:1706.02413.

[55] H. Fan and Y. Yang, "PointRNN: Point Recurrent Neural Network for Moving Point Cloud Processing," Nov. 2019. arXiv:1910.08287.

[56] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep Convolutional Networks on 3D Point Clouds," 2018. Publisher: arXiv Version Number: 3.

[57] M. Simonovsky and N. Komodakis, "Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs," 2017. Publisher: arXiv Version Number: 3.

[58] Y. Zhang and M. Rabbat, "A Graph-CNN for 3D Point Cloud Classification," 2018. Publisher: arXiv Version Number: 1.

[59] Q. Xu, X. Sun, C.-Y. Wu, P. Wang, and U. Neumann, "Grid-GCN for Fast and Scalable Point Cloud Learning," 2019. Publisher: arXiv Version Number: 5.

[60] E. Camuffo, D. Mari, and S. Milani, "Recent Advancements in Learning Algorithms for Point Clouds: An Updated Overview," *Sensors*, vol. 22, p. 1357, Jan. 2022. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[61] D. Salami, S. Palipana, M. Kodali, and S. Sigg, "Motion Pattern Recognition in 4D Point Clouds," in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, (Espoo, Finland), pp. 1–6, IEEE, Sept. 2020.

[62] H. Fan, "Awesome-Dynamic-Point-Cloud-Analysis." `https://github.com/hehefan/Awesome-Dynamic-Point-Cloud-Analysis`, Feb. 2024.

[63] D. Lu, Q. Xie, M. Wei, K. Gao, L. Xu, and J. Li, "Transformers in 3D Point Clouds: A Survey," Sept. 2022. arXiv:2205.07417.

[64] H. Fan, Y. Yang, and M. Kankanhalli, "Point 4D Transformer Networks for Spatio-Temporal Modeling in Point Cloud Videos," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Nashville, TN, USA), pp. 14199–14208, IEEE, June 2021.

[65] H. Fan, X. Yu, Y. Ding, Y. Yang, and M. Kankanhalli, "PSTNet: Point Spatio-Temporal Convolution on Point Cloud Sequences," May 2022. arXiv:2205.13713.

[66] H. Wen, Y. Liu, J. Huang, B. Duan, and L. Yi, "Point Primitive Transformer for Long-Term 4D Point Cloud Video Understanding," Dec. 2022. arXiv:2208.00281.

[67] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[68] H. Fan, Y. Yang, and M. Kankanhalli, "Point Spatio-Temporal Transformer Networks for Point Cloud Video Modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, pp. 2181–2192, Feb. 2023.

[69] X. Liu, M. Yan, and J. Bohg, "MeteorNet: Deep Learning on Dynamic 3D Point Cloud Sequences," pp. 9246–9255, 2019.

[70] D. M. Nogueira, "Automatic Identification of Spatio-Temporal Patterns to Characterize Embryogenesis," 2023.

[71] J. Owoyemi and K. Hashimoto, "Spatiotemporal Learning of Dynamic Gestures from 3D Point Cloud Data," 2018. Publisher: arXiv Version Number: 1.

[72] Y. Wei, H. Liu, T. Xie, Q. Ke, and Y. Guo, "Spatial-Temporal Transformer for 3D Point Cloud Sequences," pp. 1171–1180, 2022.

[73] X. Li, Q. Huang, Z. Wang, Z. Hou, and T. Yang, "SequentialPointNet: A strong frame-level parallel point cloud sequence network for 3D action recognition," Mar. 2022. arXiv:2111.08492.

[74] Y. Min, "FlickerNet: Adaptive 3D Gesture Recognition from Sparse Point Clouds," in *2019 British Machine Vision Conference (BMVC)*, vol. 30, (Cardiff, UK), p. 5, Sept. 2019.

[75] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.

[76] J. Stegmaier, *New methods to improve large-scale microscopy image analysis with prior knowledge and uncertainty.* KIT Scientific Publishing, 2017.

[77] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, Jan. 1979.

[78] J. Stegmaier, J. Arz, B. Schott, J. C. Otte, A. Kobitski, G. U. Nienhaus, U. Strahle, P. Sanders, and R. Mikut, "Generating semi-synthetic validation benchmarks for embryomics," in *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, (Prague, Czech Republic), pp. 684–688, IEEE, Apr. 2016.

[79] U. Ayachit, *The ParaView guide: updated for ParaView version 4.3.* Clifton Park, NY: Kitware Inc, full color version ed., 2015.

[80] "scipy.spatial.transform.Rotation.random, SciPy v1.11.4 Manual." `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.random.html`.

[81] R. Huang, W. Zhang, A. Kundu, C. Pantofaru, D. A. Ross, T. Funkhouser, and A. Fathi, "An LSTM Approach to Temporal 3D Object Detection in LiDAR Point Clouds," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm,

eds.), Lecture Notes in Computer Science, (Cham), pp. 266–282, Springer International Publishing, 2020.

[82] B. Schott, “Interactive and Quantitative Knowledge-Discovery in Large-Scale 3D Tracking Data,” 2018. Medium: PDF Publisher: Karlsruhe.

[83] J. Sola and J. Sevilla, “Importance of input data normalization for the application of neural networks to complex industrial problems,” *IEEE Transactions on Nuclear Science*, vol. 44, pp. 1464–1468, June 1997.

[84] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Jan. 2017. arXiv:1412.6980.

[85] M. Grandini, E. Bagli, and G. Visani, “Metrics for Multi-Class Classification: an Overview,” Aug. 2020. arXiv:2008.05756.

Appendix A

List of Abbreviations

BCEL	Binary Cross-Entropy Loss
CEL	Cross-Entropy Loss
CLBP	Canonical Lagrangian Biomechanical Profiles
CNN	Convolutional Neural Network
DSLM	Digital Scanned Laser light sheet fluorescence Microscopy
DGCNN	Dynamic Graph Convolutional Neural Network
ESF	Ensemble of Shape Function
EVL	Enveloping Layer
FPS	Farthest Point Sampling
HOG	Histogram Of Gradients
hpf	hours post fertilization
IDG	Incremental Deformation Gradient
knn	k-nearest neighbours
LSTM	Long Short-Term Memory
LoG	Laplacian of Gaussian
MAE	Mean Absolute Error
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
PPTr	Point Primitive Transformer
PSTNet	Point Spatio-Temporal Network
RFM	Random Forest Method
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
SAL	Set Abstraction Layer
SGD	Stochastic gradient descent
STS	Spatio-Temporal Sampling
STG	Spatio-Temporal Grouping
YSL	Yolk Syncytial Layer

Appendix B
Additional Results

B.1 Supplementary Tables

In this section the per-class metrics tables are locates as well as some confusion matrices of the best performing models of both networks.

B.1.1 PointLSTM

Movement	Frames	Class MSE	Class MAE	Precision	Recall
Nomove	10	0.129	0.309	57.9%	89.6%
	20	**0.084**	**0.231**	81.0%	**93.2%**
	32	0.103	0.254	91.1%	90.2%
	40	0.145	0.310	89.2%	85.6%
	50	0.105	0.246	**91.5%**	89.9%
Uniform	10	0.157	0.301	53.1%	82.3%
	20	**0.128**	**0.281**	**58.9%**	**86.3%**
	32	0.135	0.292	56.4%	84.1%
	40	0.184	0.362	55.9%	80.7%
	50	0.153	0.304	51.7%	79.1%
Expansion	10	0.340	0.479	83.9%	63.0%
	20	0.318	0.449	86.6%	63.3%
	32	**0.261**	**0.372**	91.5%	**69.7%**
	40	0.298	0.408	**92.5%**	67.6%
	50	0.312	0.425	91.8%	65.6%
Compression	10	0.315	0.470	72.2%	66.8%
	20	0.152	0.307	73.8%	85.2%
	32	0.029	0.115	71.1%	98.2%
	40	0.099	0.233	**77.8%**	91.3%
	50	**0.013**	**0.063**	73.6%	**99.0%**
Rotation	10	0.411	0.559	50.2%	47.2%
	20	**0.275**	**0.427**	60.8%	**61.7%**
	32	0.299	0.458	**71.6%**	56.8%
	40	0.350	0.522	59.3%	50.6%
	50	0.371	0.558	59.8%	44.5%
Radial	10	0.132	0.278	94.2%	87.7%
	20	0.008	0.041	99.6%	99.4%
	32	0.003	0.019	99.0%	**99.7%**
	40	**0.003**	**0.009**	99.3%	**99.7%**
	50	0.003	0.012	**100%**	**99.7%**
Shear	10	0.595	0.748	57.9%	20.7%
	20	0.472	0.626	79.3%	38.8%
	32	0.424	0.541	**82.7%**	47.4%
	40	**0.295**	**0.400**	75.3%	**62.9%**
	50	0.364	0.453	80.4%	55.2%

Table B.1: Per-class metrics for fr of the PointLSTM network

Movement	Frames	Class MSE	Class MAE	Precision	Recall
Nomove	64	0.190	0.399	54.4%	89.1%
	128	0.090	0.249	67.0%	94.7%
	256	0.105	0.246	**91.5%**	89.9%
	512 (32 fr)	**0.073**	**0.149**	78.9%	**91.7%**
Uniform	64	0.262	0.444	53.1%	58.9%
	128	0.299	0.449	**58.3%**	54.5%
	256	0.153	0.304	51.7%	79.1%
	512 (32 fr)	**0.028**	**0.045**	53.4%	**96.7%**
Expansion	64	0.511	0.646	82.1%	47.5%
	128	0.441	0.578	87.2%	52.6%
	256	0.312	0.425	91.8%	65.6%
	512 (32 fr)	**0.259**	**0.337**	**95.0%**	**70.0%**
Compression	64	0.318	0.493	72.0%	66.8%
	128	0.252	0.443	**80.9%**	67.5%
	256	0.013	0.063	73.6%	99.0%
	512 (32 fr)	**0.000**	**0.001**	63.8%	**100%**
Rotation	64	0.399	0.617	34.9%	40.8%
	128	0.386	0.593	44.5%	**47.5%**
	256	**0.371**	**0.558**	59.8%	44.5%
	512 (32 fr)	0.551	0.611	**78.7%**	39.0%
Radial	64	0.007	0.059	**99.9%**	**99.6%**
	128	**0.003**	0.019	**100%**	**99.7%**
	256	**0.003**	**0.012**	**100%**	**99.7%**
	512 (32 fr)	0.189	0.224	**100%**	78.6%
Shear	64	0.110	0.189	53.7%	31.2%
	128	**0.262**	**0.428**	56.4%	**64.9%**
	256	0.364	0.453	80.4%	55.2%
	512 (32 fr)	0.585	0.628	**87.4%**	38.6%

Table B.2: Per-class metrics for *pts* of the PointLSTM network

	N	U	E	C	Ro	Ra	S
N	**0.92**	0.00	0.00	0.04	0.03	0.00	0.02
U	0.00	**0.97**	0.00	0.00	0.03	0.00	0.01
E	0.00	0.24	**0.70**	0.00	0.03	0.00	0.03
C	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00
Ro	0.00	**0.60**	0.00	0.00	0.39	0.00	0.00
Ra	0.19	0.00	0.00	0.02	0.00	**0.79**	0.00
S	0.05	0.00	0.04	**0.51**	0.02	0.00	0.39

Table B.3: Confusion matrix for testing with 512 points with 32 time frames

	N	U	E	C	Ro	Ra	S
N	**0.90**	0.00	0.00	0.00	0.04	0.00	0.06
U	0.01	**0.79**	0.00	0.00	0.19	0.00	0.01
E	0.01	0.21	**0.66**	0.00	0.06	0.00	0.06
C	0.00	0.00	0.00	**0.99**	0.00	0.00	0.01
Ro	0.03	**0.53**	0.00	0.00	0.44	0.00	0.00
Ra	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00
S	0.03	0.00	0.06	0.35	0.00	0.00	**0.55**

Table B.4: Confusion matrix for testing with 256 points with 50 time frames

	N	U	E	C	Ro	Ra	S
N	**0.90**	0.00	0.00	0.02	0.00	0.00	0.07
U	0.00	**0.84**	0.00	0.00	0.16	0.00	0.00
E	0.00	0.23	**0.70**	0.00	0.05	0.01	0.01
C	0.00	0.00	0.00	**0.98**	0.01	0.00	0.01
Ro	0.01	0.42	0.00	0.00	**0.57**	0.00	0.00
Ra	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00
S	0.07	0.00	0.06	0.38	0.01	0.00	**0.47**

Table B.5: Confusion matrix for testing with 256 points with 32 time frames

Movement	Frames	Class MSE	Class MAE	Precision	Recall
Nomove	CLE	**0.105**	**0.246**	**91.5%**	**89.9%**
	Class MSE	0.659	0.812	47.3%	85.5%
	Class MAE	0.765	0.875	0.0%	0.0%
	BCEL	0.487	0.697	37.3%	83.9%
Uniform	CLE	**0.153**	**0.304**	**51.7%**	**79.1%**
	Class MSE	0.698	0.835	51.2%	32.5%
	Class MAE	0.769	0.877	0.0%	0.0%
	BCEL	0.506	0.711	30.7%	1.1%
Expansion	CLE	**0.312**	**0.425**	91.8%	65.6%
	Class MSE	0.660	0.809	**93.8%**	45.7%
	Class MAE	0.645	0.800	93.0%	48.2%
	BCEL	0.406	0.568	91.1%	**67.1%**
Compression	CLE	**0.013**	**0.063**	**73.6%**	99.0%
	Class MSE	0.481	0.693	68.5%	**99.9%**
	Class MAE	0.474	0.689	49.7%	**100%**
	BCEL	0.034	0.157	70.7%	99.8%
Rotation	CLE	**0.371**	**0.558**	**59.8%**	44.5%
	Class MSE	0.682	0.826	39.4%	38.7%
	Class MAE	0.543	0.736	35.1%	**95.5%**
	BCEL	0.454	0.673	29.1%	27.6%
Radial	CLE	**0.003**	**0.012**	**100%**	99.7%
	Class MSE	0.477	0.691	100%	99.0%
	Class MAE	0.476	0.689	100%	99.2%
	BCEL	**0.003**	0.043	99.4%	**99.9%**
Shear	CLE	**0.364**	**0.453**	**80.4%**	**55.2%**
	Class MSE	0.655	0.805	70.0%	44.7%
	Class MAE	0.693	0.829	42.3%	31.9%
	BCEL	0.496	0.631	75.3%	46.1%

Table B.6: Per-class metrics for *loss* of the PointLSTM network

B.1.2 DGCNN

Movement	Frames	Class MSE	Class MAE	Precision	Recall
Nomove	10	0.652	0.807	0.0%	0.0%
	20	0.163	0.379	43.7%	94.0%
	32	**0.101**	**0.285**	71.7%	95.2%
	40	0.180	0.400	81.0%	**95.8%**
	50	0.128	0.293	**89.0%**	85.2%
Uniform	10	0.636	0.797	1.3%	0.0%
	20	0.248	0.413	52.8%	58.9%
	32	0.155	0.295	55.6%	78.6%
	40	0.109	0.284	58.5%	**92.4%**
	50	**0.074**	**0.164**	**62.7%**	89.6%
Expansion	10	0.598	0.745	75.0%	51.7%
	20	0.441	0.607	83.7%	52.4%
	32	0.316	0.470	91.6%	63.3%
	40	0.386	0.565	85.5%	62.6%
	50	**0.260**	**0.361**	**95.5%**	**69.5%**
Compression	10	0.692	0.823	38.5%	30.2%
	20	0.559	0.713	70.3%	35.1%
	32	0.239	0.442	72.3%	75.0%
	40	**0.086**	**0.287**	61.5%	**100%**
	50	0.144	0.339	**77.4%**	82.5%
Rotation	10	0.603	0.776	14.8%	56.2%
	20	0.307	0.529	37.4%	53.2%
	32	0.271	0.449	62.4%	57.7%
	40	0.258	0.461	70.0%	58.8%
	50	**0.201**	**0.328**	**80.6%**	**70.2%**
Radial	10	0.659	0.811	26.3%	26.7%
	20	0.082	0.175	98.3%	92.2%
	32	0.102	0.186	99.5%	90.7%
	40	0.104	0.249	**100%**	**93.6%**
	50	**0.069**	**0.156**	**100%**	**93.6%**

Shear	10	0.782	0.884	**99.9%**	0.5%
	20	0.541	0.721	47.5%	11.8%
	32	0.406	0.603	55.7%	38.7%
	40	0.627	0.777	89.6%	9.2%
	50	**0.238**	**0.422**	62.9%	**63.6%**

Table B.7: Per-class metrics for fr of the DGCNN network

Movement	Frames	Class MSE	Class MAE	Precision	Recall
Nomove	32	0.175	0.386	57.1%	91.9%
	64	0.133	0.329	68.6%	92.7%
	128	**0.101**	**0.285**	71.7%	**95.2%**
	256	0.264	0.497	**79.8%**	84.6%
Uniform	32	0.269	0.461	57.6%	54.9%
	64	0.218	0.397	51.7%	68.4%
	128	0.155	0.295	55.6%	78.6%
	256	**0.130**	**0.241**	**58.2%**	**83.8%**
Expansion	32	0.412	0.575	75.5%	53.1%
	64	0.425	0.575	86.5%	51.9%
	128	**0.316**	**0.470**	**91.6%**	**63.3%**
	256	0.493	0.643	56.2%	47.7%
Compression	32	0.307	0.525	60.6%	78.3%
	64	0.311	0.524	69.8%	62.6%
	128	**0.239**	**0.442**	**72.3%**	**75.0%**
	256	0.282	0.471	70.2%	72.5%
Rotation	32	0.325	0.554	36.4%	53.4%
	64	0.316	0.526	45.7%	49.5%
	128	0.271	0.449	**62.4%**	57.7%
	256	**0.189**	**0.347**	60.8%	**73.4%**
Radial	32	0.129	0.244	**100%**	90.7%
	64	**0.009**	**0.035**	98.0%	**99.3%**
	128	0.102	0.186	99.5%	90.7%
	256	0.037	0.128	99.1%	98.3%
Shear	32	0.580	0.757	44.5%	3.1%
	64	0.445	0.652	44.3%	32.5%
	128	**0.406**	**0.603**	55.7%	**38.7%**
	256	0.552	0.692	**73.5%**	31.0%

Table B.8: Per-class metrics for pts of the DGCNN network

Movement	Frames	Class MSE	Class MAE	Precision	Recall
Nomove	20	0.101	0.285	71.7%	95.2%
	32	0.080	0.245	85.7%	95.3%
	48	**0.066**	**0.217**	**86.2%**	**96.3%**
Uniform	20	0.155	0.295	55.6%	78.6%
	32	0.146	**0.259**	**51.3%**	78.6%
	48	**0.138**	0.261	51.2%	**80.2%**
Expansion	20	0.316	0.470	91.6%	63.3%
	32	**0.299**	**0.447**	91.3%	**64.7%**
	48	0.323	0.461	**92.4%**	61.0%
Compression	20	0.239	0.442	72.3%	75.0%
	32	0.310	0.498	**83.1%**	53.7%
	48	**0.136**	**0.312**	74.6%	**87.0%**
Rotation	20	**0.271**	**0.449**	**62.4%**	**57.7%**
	32	0.357	0.533	57.6%	44.9%
	48	0.344	0.523	57.9%	45.4%
Radial	20	0.102	0.186	99.5%	90.7%
	32	0.052	0.118	**100%**	94.8%
	48	**0.010**	**0.031**	99.3%	**99.0%**
Shear	20	0.406	0.603	55.7%	38.7%
	32	**0.240**	**0.430**	53.6%	**68.2%**
	48	0.336	0.516	**70.8%**	50.1%

Table B.9: Per-class metrics for k of the DGCNN network

Movement	Frames	Class MSE	Class MAE	Precision	Recall
Nomove	512	0.265	0.462	**88.2%**	71.5%
	1024	**0.101**	**0.285**	71.7%	**95.2%**
Uniform	512	0.192	0.324	**59.3%**	71.9%
	1024	**0.155**	**0.295**	55.6%	**78.6%**
Expansion	512	0.326	0.474	**92.7%**	62.3%
	1024	**0.316**	**0.470**	91.6%	**63.3%**
Compression	512	**0.191**	**0.392**	69.4%	**76.9%**
	1024	0.239	0.442	**72.3%**	75.0%
Rotation	512	**0.213**	**0.359**	55.8%	**68.3%**
	1024	0.271	0.449	**62.4%**	57.7%
Radial	512	**0.005**	**0.009**	91.8%	**99.4%**
	1024	0.102	0.186	**99.5%**	90.7%
Shear	512	**0.372**	**0.566**	48.2%	**42.9%**
	1024	0.406	0.603	**55.7%**	38.7%

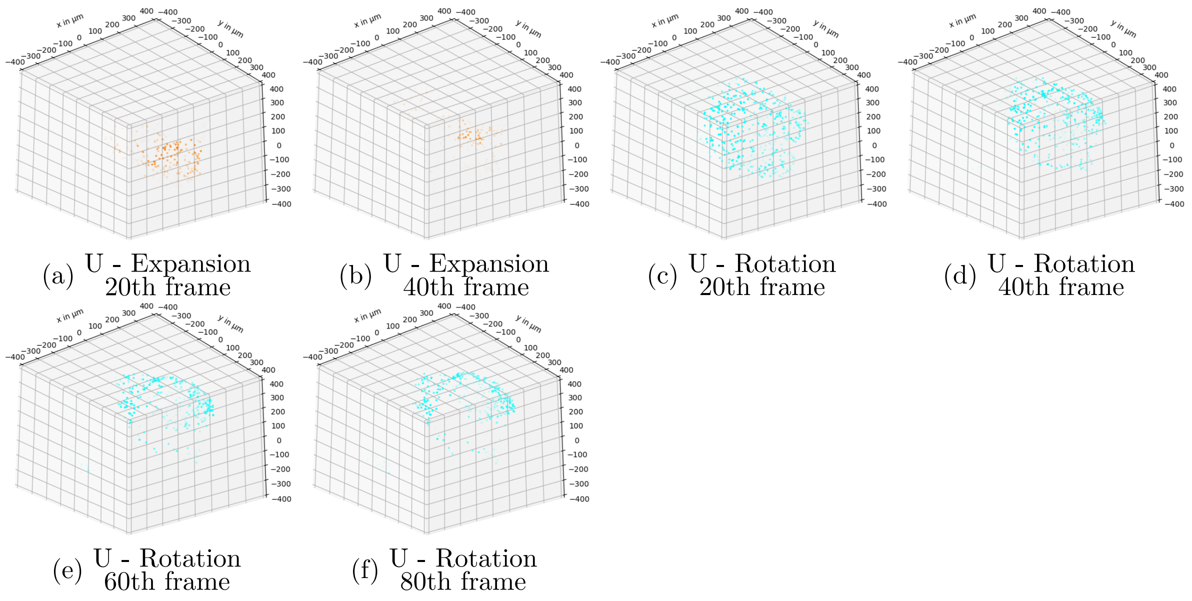
Table B.10: Per-class metrics for *emb* of the DGCNN network

Movement	Class MSE	Class MAE	Precision	Recall
Nomove	0.082	0.254	84.6%	95.6%
Uniform	0.156	0.312	56.4%	74.8%
Expansion	0.424	0.571	91.5%	52.2%
Compression	0.183	0.381	74.3%	77.2%
Rotation	0.190	0.380	64.0%	72.3%
Radial	0.042	0.105	100%	95.9%
Shear	0.311	0.490	59.9%	50.7%

Table B.11: Per-class metrics for combined optimal parameters of DGCNN network

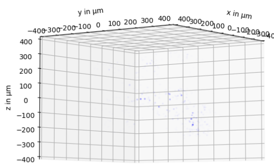
B.2 Supplementary Figures

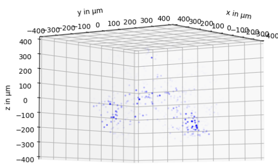
B.2.1 Artificial Point Cloud Testing

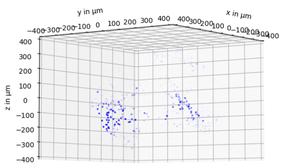


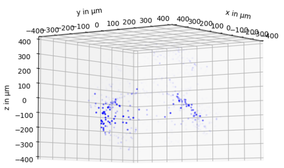
(a) U - Expansion 20th frame (b) U - Expansion 40th frame (c) U - Rotation 20th frame (d) U - Rotation 40th frame (e) U - Rotation 60th frame (f) U - Rotation 80th frame

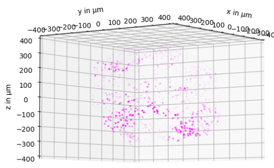
Figure B.1: Visualization of the magnitude of other movements predicted in the uniform point cloud.
Colormap: N - ●, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●

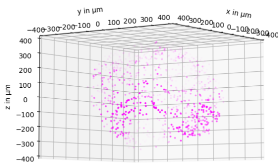

(a) E - Radial 20th frame

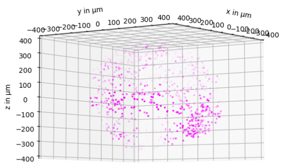

(b) E - Radial 40th frame


(c) E - Radial 60th frame


(d) E - Radial 80th frame


(e) E - Shear 20th frame


(f) E - Shear 40th frame


(g) E - Shear 60th frame

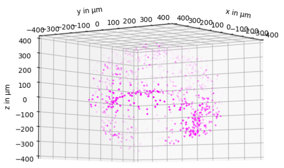
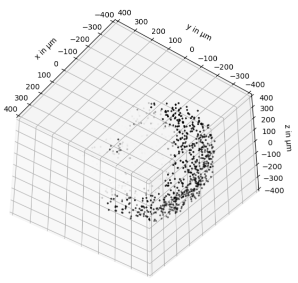
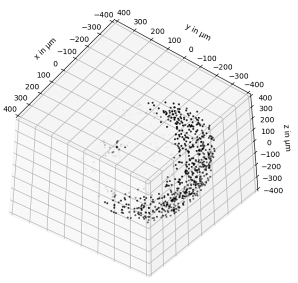
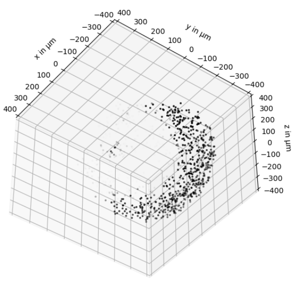

(h) E - Shear 80th frame

Figure B.2: Visualization of the magnitude of other movements predicted in the expansion point cloud.
Colormap: N - ●, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●


(a) Ro - Nomove 20th frame


(b) Ro - Nomove 40th frame


(c) Ro - Nomove 60th frame

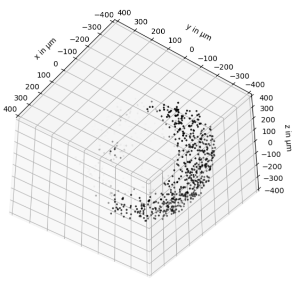

(d) Ro - Nomove 80th frame

Figure B.3: Visualization of the magnitude of other movements predicted in the rotation point cloud.
Colormap: N - ●, U - ●, E - ●, C - ●, Ro - ●, Ra - ●, S - ●