Improving
the Motion Processing Hierarchy for
Attending to Visual Motion

Xiao Lei Zhang

A Thesis Submitted to
The Faculty of Graduate Studies
In Partial Fulfilment of the Requirements
For the Degree of **Master of Applied Science**

Graduate Program in Electrical Engineering and Computer Science
York University
Toronto, Ontario

November 2023

Abstract

Visual motion has been studied for decades now. Attention to motion using Selective Tuning involves a top-down selection mechanism within a feed-forward motion hierarchy. Researchers have proposed various models for the motion hierarchy. In this thesis, we introduce a learnable hierarchy, based on fully convolutional networks, ST-Motion-Net. The Selective Tuning model for visual attention is demonstrated on ST-Motion-Net to localize motion patterns and segment moving objects. We create two datasets, Blender-MP and Blender-Complex, to evaluate ST-Motion-Net on motion pattern detection, localization, and motion segmentation tasks. ST-Motion-Net achieves excellent performance on motion pattern detection and localization for each area of ST-Motion-Net. For motion segmentation, we evaluate 2-Frame-Area-V1 of ST-Motion-Net on the task. 2-Frame-V1 contains neurons that respond to translation motion, given 2 most recent frames of a temporal sequence. 2-Frame-V1 achieves 86.84% IoU on Blender-MP-Test, which surpass some state-of-the-art models. On Blender-Complex-Test, 2-Frame-V1 reaches 52.61% IoU, which also achieves state-of-the-art performance.

Acknowledgements

I would like to thank the Graduate program for the funding and the scholarship from York University.

I would like to express special thanks to my supervisor, Professor John K. Tsotsos, for his continuous support throughout my study at York University.

I am truly grateful to my committee members, Prof. Michael Brown and Prof. George Z. H. Zhu, for reviewing my thesis.

I would like to thank Bao Xin Chen, Calden Wloka, and Mahdi Biparva for helping in training, dataset creation, using the ST Library, and the benchmarks.

Finally, I would like to thank my parents for their unconditional support.

Contents

List of Tables

List of Figures

List of Abbreviations

ALC	Agglomerative Lossy Compression
BMS-26	Berkeley Motion Segmentation Dataset
BU	Bottom-up
CIFAR-10	Canadian Institute for Advanced Research image dataset with 10 classes
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRF	Conditional Random Field
CVAT	Computer Vision Annotation Tool
DAVIS	Densely Annotated Video Segmentation Dataset
EM	Expectation Maximization
E-MRF	Extended Markov Random Field
FBMS-59	Freiburg-Berkeley Motion Segmentation Dataset
FCN	Fully Convolutional Network
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
GT	Ground Truth
HTC	Hybrid Task Cascade
IoU	Intersection over Union Score
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
KITTI MOD	KITTI Moving Object Detection Dataset
LSA	Local Subspace Affinity
LiDAR	Light Detection and Ranging
MAP	Maximum A-Posteriori Probability

MCS	Maximum Consensus Subspace
MCV	Motion Cost Volume
MLE	Maximum Likelihood Estimation
MNIST	Modified National Institute of Standards and Technology database
MS COCO	Microsoft Common Objects in Context Dataset
MSE	Mean Squared Error
MST	Medial Superior Temporal Area
MT	Middle Temporal Visual Area
MoA-Net	Motion Angle Network
PASCAL	Pattern Analysis, Statistical Modelling and Computational Learning
PASCAL VOC	PASCAL Visual Object Classes Dataset
PF	Particle Filter
P-WTA	Parametric Winner-take-all
RANSAC	RANdom SAmple Consensus
RGB	Image with Red, Green, and Blue channels
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
ReLU	Rectified Linear Unit
SCC	Spectral Curvature Clustering
SLIC	Simple Linear Iterative Clustering
SSC	Sparse Subspace Clustering
SSN	Selective Segmentation Network
ST	Selective Tuning
SVD	Singular Value Decomposition
SfM	Structure from Motion
TD	Top-down

UFO	Unidentified Flying Object
V1	Primary Visual Cortex
V2	Secondary Visual Cortex
VGG	CNN Architecture developed by the Visual Geometry Group
WTA	Winner-take-all
YTVOS	YouTube Video Object Segmentation Dataset

Chapter 1

Introduction

Visual motion has been a hot area of research for decades. Attention to visual motion is about selectively focusing on motion in the visual input with a model for visual attention. Attention to motion using the Selective Tuning (ST) model involves a top-down (TD) selection mechanism within a motion-processing hierarchy [26], a neurally-inspired model of the primate visual motion system. The TD process sends attention signals downwards through the hierarchy to compute selections for each layer, until the task requirements are fulfilled. Researchers have proposed various models for this hierarchical system, including a feedforward motion-processing pyramid [26] [27] and a motion estimation model using orientation tensor formalism [28].

Selective Tuning mechanisms are demonstrated on a feed-forward motion-processing pyramid in [26] [27] to attend to motion and localize and label motion patterns. The model uses realistic feature pyramids to extract the motion features, but the filters are not learned ones. The output from the lower areas of the model may become noisy. A motion estimation model [28] is later proposed, which extracts the velocity field from spatiotemporal input using quadrature filters. However, it is difficult to implement ST within this network. In this thesis, we create a learnable motion processing hierarchy that supports the ST model for different tasks, such as motion pattern localization [26] and motion segmentation.

1.1 Motivation

Computer Vision has advanced rapidly in recent years with the emergence of deep learning methods. This leads to many useful applications, such as Object detection [58] [60], Object

tracking [61] [62] [63] [64], Image segmentation [58] [59], Image classification [7], Image super-resolution [65] [66] [67] [68], Optical flow estimation [12] [53] [54], and Motion segmentation [47] [48] [56].

Among these areas, there is one area that remains unsolved in computer vision research, attending to visual motion. Although vision transformers are claimed to implement attention mechanisms, a recent study [174] suggests the purely feedforward vision transformers perform similarity grouping (e.g. colour) rather than attention. Attending to motion using the ST model employs a top-down selection mechanism within a motion hierarchy [26]. Once the bottom-up (BU) or feedforward process extracts the motion features, attention signals can be initialized to attend to motion patterns. There are various tasks, which can be performed on such systems, such as motion pattern detection, localization, and recognition [26].

Several motion hierarchies have been proposed over the years for attending to visual motion with the ST model. However, these models are either not trained [26] [27] or difficult to implement ST [28]. Thus, we develop a learnable motion hierarchy for attending to motion with ST. The improved hierarchy is also a neurally-inspired model of the primate visual motion system (similar to [26] [27]). It models motion representations in numerous areas of the primate cortex and is inspired by what is known about monkey visual motion analysis [26] [129]. Thus, the model is viewed from a computational neuroscience perspective rather than an application-oriented or engineering perspective. There are other systems for motion processing (e.g. FlowNet 2.0 [54]), but they may not satisfy human vision constraints or consider only translation motion (i.e. optical flow) in the image plane. There are many kinds of complex motion patterns, such as rotation (clockwise or counterclockwise), radial motion (isotropic expansion or contraction), deformation (normal and shear strain), and mixed motion. Few systems (e.g. [26] [27]) can deal with these complex motions, and we explore a system with these capabilities. Moreover, the improved hierarchy can attend to and localize all these motion patterns (not just translation motion in the image plane). There are not many systems with these features either

(e.g. [26] [27] can handle all the motions besides deformation). Thus, designing a learnable motion processing hierarchy with improved functionalities (e.g. detection of additional motion features) and TD attention using ST is one method to improve the motion processing hierarchy for attending to visual motion. The improved hierarchy can be tested on various tasks, such as motion pattern localization and motion segmentation. The motion hierarchy can also be used to solve some practical problems in areas, such as autonomous vehicles (e.g. segment moving vehicles and pedestrians) and sport analytics (e.g. detect and localize motion patterns of people and equipment).

1.2 Contributions

The purpose of this thesis is to create an improved motion processing hierarchy that attends to visual motion using ST. As mentioned earlier, one improvement is to design a learnable motion processing hierarchy. We also implement additional features on the motion hierarchy, such as additional neural populations (more speed selectivity types than [26] [27] and selecitvity for deformation motion) and RGB input sequence (i.e. images with red, blue, and green channels).

In this thesis, we propose a learnable feedforward motion processing hierarchy based on fully convolutional networks (FCNs) [19] [22], ST-Motion-Net. FCN is a convolutional neural network (CNN) without any fully connected layers which supports input of arbitrary size. CNN is a type of feed-forward neural network that can be used to process visual data (e.g. images) with learned filters. ST-Motion-Net supports RGB input video frames and neurons of 5 distinct speed selectivity types. The outputs from the lower areas of the hierarchy are no longer noisy compared to [26] [27], as the filters are learned from accurate ground truth (GT) feature maps. ST-Motion-Net models 4 areas (V1, MT, MST, and 7a) of the primate visual motion system and each area sends its output to the next area. The Primary Visual Cortex (V1) is an area in the

visual cortex that receives visual data from the lateral geniculate nucleus, where visual input from the eyes travels through to reach the visual cortex. V1 neurons are selective for a variety of features (e.g. direction and speed of motion) [177]. The Middle Temporal Visual Area (MT) is an area in the visual cortex that is involved in visual motion perception; MT neurons are also selective for the direction of speed gradient (i.e. spatial derivatives of speed) [177]. The Medial Superior Temporal Area (MST) receives input from MT and can analyse more complex motion patterns (e.g. expansion or contraction and rotations) [177]. Area 7a receives input from Area MST and can process complex motion patterns; 7a neurons also respond to rotation or radial motion regardless of direction [175] [176]. For ST-Motion-Net, neurons in Area V1 respond to local translation motion, given a temporal sequence of input frames. Neurons in Area MT respond to local translation motion and speed gradient (i.e. spatial derivatives of speed). Neurons in Area MST respond to translation and spiral motion patterns (clockwise or counterclockwise rotation, isotropic expansion or contraction, and mixed motion). Area 7a contains neurons that respond to translation and spiral motion, and has additional neurons that respond to rotation, radial motion, and deformation from normal or shear strain regardless of direction. Thus, the hierarchy covers the full spectrum of affine motions (i.e. image translation, rotation, divergence, and deformation). We implement ST within this hierarchy to attend to motion and localize motion patterns. In addition, we test the model on a motion segmentation task.

We create two datasets in Blender [37] to benchmark the motion-processing hierarchy. The first dataset contains sequences of textured geometric objects moving slowly on still backgrounds with different motion patterns including translation, radial motion, rotation, and deformation. Shadows or illumination on the objects may change. GT output feature maps for each area, motion pattern localization masks, and motion segmentation masks are available for each sequence. The dataset contains 20 sequences of length 10 for both the validation and test dataset. We call this dataset the Blender Motion Pattern (Blender-MP) dataset.

In addition, we create another dataset to test the hierarchy on real-world objects with complex motion patterns. This dataset contains 25 sequences of textured real-world objects [98] – [119] moving on still backgrounds with different motion patterns, including translation, radial motion, rotation, deformation, and mixed motion patterns (e.g. rotation and radial motion). Some moving objects may overlap during motion and may exit or re-enter the scene. Shadows or illumination on the objects may change. GT output feature maps, motion pattern localization masks, and motion segmentation masks are also available for each sequence. The dataset contains 25 sequences of length 1000. We use 4 of the sequences for validation and 21 of the sequences for testing. We call this dataset the Blender-Complex dataset.

1.3 Thesis Outline

This thesis contains 6 chapters.

- **Chapter 1** introduces the motivation of this thesis and describes the contributions of this thesis in improving the motion-processing hierarchy for attending to visual motion.

- **Chapter 2** reviews the literature on motion segmentation. The background literature on motion processing hierarchy, Selective Tuning, and Selective Tuning of Convolutional Neural Networks are also reviewed.

- **Chapter 3** describes the motion hierarchy, ST-Motion-Net, and tests ST-Motion-Net on the motion pattern detection task.

- **Chapter 4** tests ST-Motion-Net on the motion pattern localization task.

- **Chapter 5** tests 2-Frame Area V1 of ST-Motion-Net on the motion segmentation task and compares the results with some recent motion segmentation models.

- **Chapter 6** concludes this thesis and provides future direction.

Chapter 2

Background

In this chapter, we will review the literature on motion segmentation. We will also review the literature on motion processing hierarchy, Selective Tuning, and Selective Tuning of Convolutional Neural Networks (CNNs).

2.1 Motion Segmentation

The goal of motion segmentation is to segment all independently moving objects in a video sequence [30]. Other than motion of objects, camera motion may be present in the video. There are 2 types of motion segmentation. For binary motion segmentation, the goal is to assign each pixel of a video frame with one of 2 labels, static environment or independently moving objects [30]. For multi-label or instance motion segmentation, the goal is to assign each pixel of a video frame with one of k labels, which include each independently moving object and the static environment [30]. Thus, each independently moving object is assigned a separate label. In this thesis, we test the motion hierarchy on the binary motion segmentation task, which computes a binary mask of the moving objects for each video frame. However, we will review works pertaining to both tasks.

Zappella et al. (2009) [33] divide motion segmentation methods into 6 categories, which include image difference, optical flow, wavelet, statistical, layers, and manifold clustering. There are also recent works that use deep learning [131].

2.1.1 Image Difference

Image difference methods segment moving objects from the intensity difference of two frames [33]. Thresholding can be used on the difference to compute a binary mask of the moving objects. For the basic version of this method, it is sensitive to noise or changes in lighting and does not handle camera motion [121]. Image difference methods can handle occlusions, multiple objects, rigid / non-rigid motion, and articulated motion.

There are works on image difference that address these problems. Cavallaro et al. (2000) [122] employ image difference in a probability test to adjust the thresholds locally and label the pixels of each frame. This system does not require any manual threshold tuning and is less sensitive to noise. It can also handle temporary stopping of moving objects [33].

Cheng and Chen (2006) [123] compute image difference from discrete wavelet transforms of the images to reduce noise. This method can handle noise in the images. Colombari et al. (2006) [124] model each pixel of the background as the median of the colours of the pixel over time. This creates a mosaic of the background. Each frame is then warped onto the mosaic of the background and moving objects are segmented by differencing each frame with the mosaic frame. The resulting binary mask is cleaned using morphology [34]. This method can handle noise and very slow camera motion. It can also handle temporary stopping of moving objects [33].

There are also methods that compute image differences from three frames. In the traditional three-frame differencing method, the differences are computed between the current frame and the previous frame and between the current frame and the next frame [35]. The difference maps are processed using thresholding operation to obtain binary masks, The binary masks are then combined using AND operation. This method is less sensitive to noise in the images.

Zhang et al. (2012) [35] combine the three-frame differencing method with morphology (erosion and dilation operations) [34]. This method is less sensitive to noise and changes in lighting than the three-frame differencing method. Some false negative predictions on the binary mask can also be corrected with morphology. Sengar et al. (2016) [36] propose an improved three-frame difference method for motion segmentation. The traditional three-frame difference method loses information due to using the AND operator. This method replaces the AND operator with MAX operator to combine the two difference maps to reduce the loss of information. The combined difference map is then divided into non-overlapping blocks of size 8x8. A pixel belongs to a moving object if the block sum is greater than some threshold and the difference is greater than the block mean. This algorithm reduces noise on the mask. The binary mask is computed and postprocessed using morphological operations and connected components. This method is less sensitive to noise than the basic three-frame differencing method.

2.1.2 Statistical Methods

Statistical Methods employ statistical theory to classify each pixel of a video frame as independently moving object or static environment. Statistical methods can handle multiple objects, occlusions, and temporary stopping of moving objects [33]. For some statistical methods, they require prior knowledge (e.g. number of motion regions). They may perform poorly in new situations that are not represented by the model (e.g. a model that only handles slow motion). There are numerous frameworks used in statistical methods, such as Maximum A-Posteriori Probability (MAP), Particle Filter (PF), and Expectation Maximization (EM) [33].

The MAP framework computes posterior probability using Bayes' rule [121]. MAP assigns each pixel to the class that maximizes the posterior probability. For motion segmentation, MAP is often combined with other methods to segment moving objects [33].

Cremers and Soatto (2005) [125] combine MAP with level sets to segment moving objects in a video. They minimize an energy functional with gradient descent to estimate the motion field and find the boundaries for the segmentation. This method can handle rigid or articulated motion [33]. It can find interior motion boundaries of objects (e.g. hole of a doughnut). It requires the number of moving regions as prior knowledge. It assumes the motion is slow (less than or equal to four pixels between frames). This model does not handle changes in lighting, as the model assumes the objects do not change brightness over time. The multi-set scheme is a local scheme. It does not perform well if objects enter the scene from locations far away from the evolving boundary.

Shen et al. (2007) [126] employ the MAP framework in a joint formulation for motion estimation, segmentation, and super resolution. They solve the formulation with a cyclic coordinate descent process, which estimates the motion fields, instance motion segmentation map, and high-resolution image in an alternate manner, given the other two. The joint formulation improves performance for each task. For motion segmentation, this method can handle rigid, non-rigid, and articulated motion [33]. It can also handle noisy images. This method requires the number of independently moving objects as prior knowledge.

Particle Filters (PF) track the evolution of a variable over time [121]. It represents the posterior distribution of the variable at time t using weighted particles. Each particle represents a state of the variable and particles with the same state can coexist. For a given state, the posterior probability of the variable can be estimated as the sum of the likelihood weights of the particles that represent the state. Each iteration of this algorithm consists of prediction and update. For prediction, the state of each particle is modified by drawing a sample from the new state distribution. For update, the weights of the particles are updated based on new observation. The particles are then resampled with replacement to eliminate unlikely particles, which have low weights.

Vaswani et al. (2007) [127] employ the PF algorithm in a geometric active contour framework for motion segmentation and tracking. They use the PF algorithm to estimate the conditional probability distribution of the state vector at time t, which contains an affine transformation for the contour at time t-1 and the contour at time t, conditioned on all observations up to time t. For motion segmentation, this method can handle rigid / non-rigid motion, and partial occlusions [33]. Shape information is included to segment objects with major occlusions. However, this limits the model's ability to segment highly deformable objects under such scenarios. This method also performs poorly if the object is completely occluded for many frames. For each test sequence, the number of particles needs to be set manually.

The Expectation Maximization (EM) algorithm is an iterative algorithm for finding the parameters of a Maximum Likelihood Estimation (MLE) [121]. Given a statistical model, EM finds the parameters of the model that most likely represent the observed data [121] (i.e. maximizes the likelihood function). EM is used in situations where the equations for finding the parameters do not have closed-form solution, such as a Gaussian Mixture Model (GMM). GMM is a probabilistic model that represents a mixture of a finite number of (multivariate) Gaussian distributions. Each iteration of EM consists of two steps, E-Step and M-Step. The E-Step computes a variational lower bound of the log likelihood function based on current parameter estimates. The M-Step then maximizes the lower bound with respect to the parameters to find the new parameter estimates. EM generally converges to a local optimum or saddle point of the likelihood function. The result depends on the initial parameters of the model.

Stolkin et al. (2008) [128] present a method for motion segmentation under conditions of extremely poor visibility, EM / Extended-Markov Random Field (E-MRF) algorithm. They make an initial estimate of the camera parameters (position and orientation coordinates) for a frame from prior camera trajectory and project a 3D model of the moving object onto the image plane at the estimated camera position. This produces a predicted binary image of the object. They then compute a segmentation mask using the E-MRF model, given the predicted binary image

and the input frame. The segmentation result is then compared with the predicted binary image to compute a new estimate of the camera parameters. The camera parameters can then be fed back into the algorithm to form an iterative scheme, which is a variant of the EM algorithm. The segmentation improves after each iteration of the EM algorithm. The model is learned for each frame of the video sequence. This method can handle changing visibility and lighting conditions and rigid motion. There are two weighting constants in the classification probability that determine the significance of information from the input image and the predicted binary image. These constants are set manually based on visibility conditions of the video sequence.

2.1.3 Wavelets

Wavelet methods employ wavelets to analyse different frequency components of the video frames [33]. Wavelet multiscale decomposition can be used as a pre-processing step to reduce noise. They rely on other methods, such as optical flow (See 2.1.4), to segment the moving objects. Wavelet methods can handle simple motions, such as translation motion in front of the camera.

Wiskott (1997) [130] uses Gabor and Mallat-wavelet transforms of a video sequence to segment moving objects and the static environment. The model computes instance motion segmentation of a video sequence. The model first estimates a low-resolution optical flow field for two successive frames based on Gabor-wavelet responses. A histogram over the flow vectors is then created and the peaks provide information on the motion vectors of different objects, which are called motion hypotheses. Mallat-wavelet transform is employed to compute grey-value gradient for the edges on the frames. For each edge pixel and motion hypothesis, the gradient of an edge in one frame is compared with the gradient of an edge in the next frame which is taken from the edge pixel displaced by the motion hypothesis. This computes an accordance value for each motion hypothesis, which is high if the gradients are similar. Each

edge pixel is then assigned to the motion hypothesis class that achieves the highest accordance value. This method can segment multiple objects, but only the edges of the objects are segmented. The motion model is pure translation and can handle translation motion in front of the camera. It does not handle arbitrary motion patterns in general. The optical flow estimation algorithm can estimate motion with a displacement up to 8 pixels.

Kong et al. (1998) [132] employ Galilean wavelets for motion segmentation. They first use Galilean wavelet transform to estimate the optical flow and edges of two successive frames. The pixels of each frame are then clustered into regions with similar motion. The contours of moving objects are then determined based on the edges of each cluster. This method can segment the contours of moving objects. The optical flow estimations from the wavelet transforms are robust against image noise, blur, and motion jitter. This method can also handle multiple objects and temporary occlusions [33]. It can also handle more complex motions involving translation, deformation, and camera motion. Clustering thresholds are tuned manually.

2.1.4 Optical Flow

Optical flow methods segment moving objects from the optical flow field of each video frame [33]. Optical flow field contains the velocity of each pixel on the frame. Thus, moving objects can be segmented from the flow map using different methods, such as clustering. Optical flow can be used with other methods (e.g. statistical, frame difference) to handle occlusions. It is sensitive to noise and changes in lighting conditions. Errors in optical flow estimation can also affect the performance of optical flow-based methods. Anthwal and Ganotra (2019) [133] give a summary of various optical flow-based methods for motion segmentation.

Li et al. (2015) [39] present a method for motion segmentation that combines TV-L optical flow with SLIC (Simple Linear Iterative Clustering) super-pixels to segment a moving object from

moving camera environment. Optical flow is computed using TV-L optical flow estimation [88] for the current frame. Super-pixel segmentation is computed for the current frame using SLIC super pixel segmentation [87]. Then the flow gradients are computed for the boundaries of the segmented super pixels. Next, a thresholding operation is applied to the magnitude of the flow gradients to find boundaries between the moving object and the environment. This computes an initial contour of the moving object. The initial contour is then refined using the dilation operation from mathematical morphology. This method can segment a single moving object under camera motion. It can handle rigid, non-rigid, and articulated motion. The flow estimation [88] is robust against noise. However, the segmentation is based on the initial contour and may contain 'holes'.

Bideau et al. (2016) [2] present a probabilistic model for motion segmentation based on optical flow. This method segments the current frame given a prior motion segmentation of the previous frame and the optical flow [134] for the previous and current frame. The first frame is segmented automatically during an initialization stage. This method can handle multiple objects, noise, rigid / non-rigid motion, articulated motion, and camera motion (translation and rotation). Some moving objects may be over-segmented or under-segmented (e.g. thin parts of objects, regions of homogeneous intensity, occlusion boundaries). This method achieves state-of-the-art performance on BMS-26 (Berkeley Motion Segmentation) [47] [96], Complex Backgrounds [25], and Camouflaged Animals [76] dataset. BMS-26 is a motion segmentation dataset with 26 video sequences and pixel-level segmentation annotation of the moving objects.

Huang et al. (2018) [38] employ homography matrices to create a background model to estimate camera motion. A dual-mode judge system then thresholds the difference between the optical flow from optical flow estimation (FlowNet 2.0 [54]) and that from the background model to segment the moving objects. The dual-mode judge system increases the accuracy of judgement and improves the system's adaptation to different situations. This system can handle

occlusions, noise, and multiple objects. It can also handle rigid / non-rigid motion, articulated motion, and camera motion. However, it is sensitive to shadows of moving objects.

2.1.5 Layers

Layer-based methods divide an image into layers of uniform motion to segment the moving objects [33]. Each layer has a depth level or 'transparency' level that indicates overlaps of layers. Layer-based methods can handle occlusions. However, they are highly complex and have various parameters which need to be tuned.

Wang and Adelson (1993) [135] present the first layers-based method for motion segmentation. They estimate optical flow for each frame with a multi-scale coarse to fine algorithm from [136]. They then create affine motion models for subregions of the frame using regression. Pixels are then grouped under the motion model that best describes the motion of the pixel. The clustering algorithm is run for multiple iterations to compute accurate motion segmentation for the frame. Current segmentation results are used as initialization to segment the next frame. Occlusion relationships between the layers can be determined based on temporal coherence of object shape and texture. Occlusions are indicated by an alpha map of the layer, which defines the transparency of the layer for each point. This method can handle multiple objects and occlusions. It is tested on the first 30 frames of the "flower garden" sequence, which contains a tree, a flower bed, and row of houses moving towards the left with different motions.

Kumar et al. (2008) [137] present a method that creates a layered representation of the scene for motion segmentation given a video sequence. They first obtain rigidly moving components for each pair of frames based on image motion. The frame is divided into patches and the best transformation is found for each patch to map the points to their positions on the next frame. They find the transformations with a Conditional Random Field (CRF) model, a type

of discriminative undirected probabilistic graphical model in which the probability of a random variable, conditioned on the observations, depends only on its neighbours in the graph. The points from the patches are then clustered into rigidly moving components based on their transformations. Next, an initial estimate of the parameters (masks, number of segments, etc.) is obtained by combining the rigid components via clustering. The segmentation masks are then refined, and the layer order parameters are determined by minimizing an energy function via graph cuts [138] [139]. This method can handle multiple objects, temporary stopping, occlusions, lighting, noise, motion blur, and camera motion (static and translating) [33]. It can handle rigid or articulated motion. It cannot handle very fast motion due to the small number of possible transformations. Errors may occur due to similar object and environment pixels. This method may also under-segment objects in shorter sequences. The algorithm has several stages and is computationally inefficient.

Min and Medioni (2008) [140] present a local spatiotemporal approach for motion segmentation. They represent each pixel of a video sequence in 5D space which consists of the position (x, y), time (t) and velocity (v_x, v_y) of the pixel. They then extract each moving object as a 3D smooth layer using a tensor voting framework [141]. A 3D smooth layer is a set of pixels in a 3D region with similar motion. It enforces the spatiotemporal smoothness constraint to segment the moving objects. This constraint assumes each pixel moves smoothly in time and neighbouring pixels of the same object move in a similar way. It does not require any particular camera or motion model (e.g. affine) and can handle objects with rigid, non-rigid, or articulated motion. This method can handle multiple objects, temporary stopping, occlusions, noise, and camera motion [142]. However, shadows of objects may be segmented. Some objects are also over-segmented or under-segmented (e.g. thin branches). There are various parameters, which are tuned manually for each sequence. This method has several processing stages and is computationally inefficient. The GPU (Graphics Processing Unit) implementation is significantly faster than the CPU-based code (Central Processing Unit). GPU is a piece of hardware

designed for graphics processing and parallel computing. CPU is the main processor of a computer, which executes instructions from computer programs.

2.1.6 Manifold Clustering

Manifold clustering methods project the original data to a lower dimensional subspace (if necessary) and cluster together the data points (e.g. point trajectories for motion segmentation) with similar embeddings [33]. Manifold clustering can be naturally used to solve the motion segmentation problem. They can segment moving objects even if they stop moving for some time, as the features on the trajectories can be tracked even if the object stops moving temporarily. These methods are non-causal since they require information from the entire video (i.e. trajectories) to segment each frame [30]. There are various motion segmentation methods that belong to the manifold clustering category. Zappella (2011) [142] divide manifold clustering methods into 5 subcategories, Iterative Methods, Statistics, Sparse Representation, Factorization Methods, and Subspace Estimation.

Iterative Methods start with an initial estimation of the clusters and iteratively refine the clusters to learn a model that describes the manifolds [142]. Iterative methods can handle noise, but require some prior knowledge to segment the moving objects (e.g. number of clusters, dimensions of local subspaces). They cannot handle occlusions in general. In addition, they are sensitive to initial estimations, as some initializations are not guaranteed to converge.

Ho et al. (2003) [143] present a clustering algorithm, K-Subspace Clustering, which can be employed for motion segmentation. This algorithm is a variant of K-means, which models each cluster as a low dimensional linear subspace. It iteratively assigns each point to the nearest subspace and computes an updated subspace (i.e. basis of the subspace) that minimizes the sum of the squares of the distance to each point of the cluster. The algorithm terminates if the subspaces stop changing. This method can handle multiple objects, rigid / non-rigid motion, and

articulated motion [142]. It requires the number of clusters and subspace dimensions as prior knowledge. It is sensitive to the initialization of clusters. For some cluster initializations, this algorithm converges to a local minimum, which may be far from optimal.

Da Silva and Costeira (2008) [144] present a subspace segmentation algorithm for motion segmentation. They segment linear subspaces spanned by tracked trajectories of points which belong to the moving objects. They segment the subspaces by finding the Maximum Consensus Subspace (MCS), which is a subspace supported by most of the data. The search is formulated as a Grassmannian optimization problem. The algorithm then recursively looks for subspaces of lower dimensions within the subspace. This method can handle multiple objects, rigid / non-rigid motion, articulated motion, and affine camera motion. It is sensitive to initial estimations, as some initializations converge to a local optimum. There are several parameters that can be tuned, including the minimum support of a subspace and the minimum / maximum dimensions of a subspace.

Statistical methods employ statistical theory for manifold clustering [142]. Some statistical models are built to handle noise. The performance degrades if noise is not considered or modelled properly. They can also handle various situations that are represented by the model. However, they cannot handle occlusions in general. They may require some prior knowledge (e.g. number of clusters, subspace dimensions) for segmentation.

Fishler and Bolles (1981) [145] introduce the RANdom SAmple Consensus (RANSAC) algorithm which estimates the parameters of a model given a dataset with outliers. RANSAC randomly samples n data points from the dataset to estimate model parameters. It then computes the residual of each point in the dataset using the model and points whose residual is below a threshold are called inliers. These steps are repeated until the number of inliers of a model is above some threshold or enough samples have been drawn [142].

RANSAC can be employed in models for motion segmentation [142]. Yan and Pollefeys (2007) [146] present a RANSAC approach for motion segmentation. They apply RANSAC on

the data (i.e. trajectories) to find a model of similar shape and motion [147] and the corresponding inliers of the model. They then remove the set of inliers from the data and repeat the process until the data is exhausted or no more models can be found. The consensus sets (i.e. sets of inliers) can then be used to segment the moving objects. This method can handle rigid and articulated motion. It does not require the number of clusters as prior knowledge for segmentation.

There are also motion segmentation methods that employ other statistical methods (e.g. EM algorithm) for manifold clustering [142]. These are described in Chapter 2 of Zappella (2011) [142].

Sparse representation methods employ compressive sensing [148] for motion segmentation. The idea is to use the least amount of information to represent the generated subspace (i.e. cluster) for each motion [142]. Sparse methods can handle occlusions of moving objects. They tend to have some parameters that are tuned for each sequence.

Rao et al. (2008) [149] present a method for motion segmentation based on trajectory clustering. They employ Agglomerative Lossy Compression (ALC) [150] to cluster the trajectories, which is a lossy compression-based agglomerative clustering algorithm. The goal is to find clusters (i.e. subspaces spanned by some of the data) with minimal dimensions. They find the clusters by minimizing an objective function using ALC. This method can handle rigid motion and camera motion. However, ALC employs a greedy optimization algorithm to obtain the segmentation, which does not always find the global minimum. There is also a parameter that is tuned for each sequence based on the number of clusters. The tuning process is automatic but is very time-consuming [33]. Segmentation performance may degrade if there are more than three clusters [151].

Elhamifar and Vidal (2009) [152] present a method for subspace clustering based on sparse representation, Sparse Subspace Clustering (SSC). Their method is based on the fact that each point in the union of subspaces (i.e. global subspace) is a linear combination of other

points. Thus, each point has a sparse representation (i.e. coefficients of the combination) in the dictionary of other points. They find the sparse representation of the data points using the Basis Pursuit algorithm by solving an L1 optimization problem [153]. They then find the clusters by applying spectral clustering [154] to a similarity matrix built from the sparse representation of the data points. For motion segmentation, they apply SSC to cluster point trajectories to different affine subspaces. This method can handle multiple objects, rigid / non-rigid motion, articulated motion, and camera motion. However, there are some parameters of SSC that need to be tuned for each sequence [142].

Tomasi and Kanade (1992) [147] introduced the first factorization method, which recovers the camera motion and 3D structure of an object based on features tracked throughout a video sequence (i.e. point trajectories). They factor the trajectory matrix W into 2 matrices, motion M and structure S using Singular Value Decomposition (SVD) [155], a factorization of the matrix. This method works for a static object under camera motion [142]. It cannot segment multiple objects, as it assumes the features belong to one object. Although this method cannot be used directly for motion segmentation, it started a new area of research, Structure from Motion (SfM) based on Factorization. SfM is a technique which reconstructs 3D structures from 2D image sequences. The new field eventually led to factorization methods for motion segmentation.

Factorization methods can also be employed for motion segmentation. They find clusters of trajectories with similar motion by factoring the trajectory matrix [142]. They are sensitive to noise and cannot handle occlusions. They may require some prior knowledge (e.g. number of clusters) to segment the moving objects. A detailed review of factorization methods for motion segmentation can be found in Chapter 2 of Zappella (2011) [142].

Costeira and Kanade (1998) [157] present an extension to [147] that can segment multiple independently moving objects given a set of trajectories tracked in a video sequence. They factor the trajectory matrix W using SVD [155] and compute the shape interaction matrix of size P x P, where P is the number of trajectories. This matrix has non-zero entries if and only if the

indices correspond to trajectories of the same object. They then sort the rows and columns of the matrix to obtain a block diagonal matrix, where each block contains data for the trajectories of an object. They find the trajectories that belong to each object using the sorted matrix. This method can handle multiple objects, rigid motion, and static camera. It does not require the number of objects as prior knowledge. However, the sorting process is very time consuming [142]. If the trajectory data are noisy, the shape interaction matrix may also have non-zero entries for entries that should be zero.

Subspace estimation methods employ algebraic tools to estimate the basis of the subspace (i.e. cluster) for each motion [142]. They do not require any initialization. However, they suffer from the curse of dimensionality, which can be solved using different techniques, such as projecting the data to smaller subspaces. They cannot handle occlusions and may require some prior knowledge (e.g. number of clusters, subspace dimensions) to segment the moving objects. A detailed review of subspace estimation methods can be found in Chapter 2 of Zappella (2011) [142].

Yan and Pollefeys (2006) [160] present a subspace estimation method for motion segmentation, Local Subspace Affinity (LSA). They first project trajectories to a new subspace using SVD [155] to reduce noise. They then apply local sampling to estimate local subspaces in the transformed subspace. They compute each local subspace (i.e. basis) using SVD from local samples of a point (i.e. trajectory) and its n closest neighbours. Next, they build an affinity matrix to compare the local subspaces. They measure the affinity between two local subspaces using their principal angles [156]. They obtain the final segmentation (i.e. subspaces) from the affinity matrix using spectral clustering [154]. This method can handle multiple objects, rigid / non-rigid motion, articulated motion, and camera motion. It requires the number of clusters as prior knowledge. The dimensions of the subspaces can be estimated but it is a rather difficult task. It also cannot handle incomplete trajectories from occlusions or changes in lighting.

Chen and Lerman (2009) [158] [159] present the Spectral Curvature Clustering (SCC) algorithm for segmenting affine subspaces. They first sample c subsets of d + 1 distinct points (i.e. trajectories) from the data. They then compute the affinity between each point (N points in total) and each subset using squared polar curvature [158]. This forms an affinity matrix A of size N x c. Next, they compute pairwise weights $W = AA^T$ of size N x N and apply spectral clustering [154] to find K clusters. In the process, a parameter σ of polar curvature is tuned by computing the affinity matrix and the clusters d + 1 times with different candidate values of σ. They record the clusters with the smallest total error. For each cluster, they sample c / K subsets of points (of size d + 1) and repeat the steps on the c subsets to find K newer clusters. This procedure is repeated until convergence and is referred to as iterative sampling [158]. This method can handle multiple objects, noisy trajectories, rigid / non-rigid motion, articulated motion, and camera motion. However, it cannot handle occlusions or incomplete trajectories [142]. It also requires the number of clusters (K) and maximum dimension of subspaces (d) as prior knowledge.

2.1.7 Deep Learning Methods

Deep learning methods employ neural networks, such as Convolutional Neural Networks (CNNs), for motion segmentation [131]. These methods require a training set to learn the parameters of the model. The training stage has a strong influence on the performance of deep learning methods.

Tokmakov et al. (2017) [55] present the first deep learning method for motion segmentation, MP-Net. The model is a fully convolutional encoder-decoder style network learned from ground truth optical flow and motion segmentation of synthetic video sequences from FlyingThings3D dataset [69]. The output of the model is filtered using objectiveness map computed from object proposals [70] to include only pixels of moving objects. The result is then

refined with a fully connected conditional random field (CRF) [71]. This method can handle multiple objects, noise, occlusions, rigid / non-rigid motion, articulated motion, and camera motion. The model shows excellent performance on synthetic sequences. However, the performance on real sequences is limited.

Heo et al. (2017) [45] propose a deep learning model that combines an appearance network and a motion network based on the VGG (Visual Geometry Group) architecture [46] to segment moving objects. VGG is a CNN architecture developed by the Visual Geometry Group at Oxford University. This system achieves robust performance on sequences with camera motion. This system effectively copes with background contamination by foreground and surpasses state-of-the-art methods on videos captured with freely moving cameras [45]. It can handle multiple objects, noise, rigid / non-rigid motion, articulated motion, and camera motion.

Siam et al. (2018) [72] propose a two-stream system, MODNet, for joint vehicle detection and motion segmentation using appearance and motion cues. A previous model, MPNet [55], uses dense optical flow and CNN for motion segmentation. MODNet employs a shared 2-stream VGG encoder to extract motion and appearance features given input RGB frame and optical flow. The combined motion and appearance features are then fed to two decoders for vehicle detection and motion segmentation. The results show that joint training of the two tasks improve accuracy compared to training independently. For the motion segmentation task, MODNet improves the mAP score on KITTI MOD (KITTI Moving Object Detection) dataset [72] by 21.5%, when compared to MP-Net. This method is trained on a real dataset, KITTI MOD, and has improved performance on real datasets over MP-Net. It can handle multiple objects, noise, occlusions, rigid / non-rigid motion, articulated motion, and camera motion.

Bideau et al. (2018) [73] present a self-supervised approach for motion segmentation. Deep learning methods usually a require large amount of labelled training data to achieve good performance. This paper creates a procedure to generate training data from GT binary segmentation masks, that does not depend on any colour images or other labelled training data.

For prediction, the model first removes flows due camera rotation from the optical flow field. A neural network, Motion Angle Network (MoA-Net), then predicts the motion segmentation, given the remaining optical flow angle field. MoA-Net achieves state-of-the-art performance on the synthetic dataset, Sintel [74]. This method is trained in a self-supervised way and can automatically generate labelled data for training. It can handle multiple objects, noise, occlusions, rigid / non-rigid motion, articulated motion, and camera motion. However, the model is not tested on real datasets.

Bideau et al. (2018) [75] later propose a hierarchical model for instance motion segmentation that combines piece-wise rigid motion estimation into articulated objects under the guidance of semantic segmentations from CNN and a second object level statistical model. CNN methods can identify well known structures but may not accurately characterize geometric constraints. This model combines traditional geometric knowledge with semantic segmentation from CNN for motion segmentation. Given input optical flow, the first level of this model estimates rigid motion components. The second level groups the motion components under the guidance of object proposals from SharpMask [70] to compute the motion segmentation. The segmentation is then refined using a fully connected CRF [71]. This model has excellent performance on a range of motion segmentation benchmarks, including FBMS-59 (Freiburg-Berkeley Motion Segmentation) [47] [48], Complex Background dataset [76], and Camouflaged Animals dataset [2]. FBMS-59 is an extension of the BMS-26 motion segmentation dataset [47] [96] with 33 additional video sequences. This method can handle multiple objects, noise, occlusions, rigid / non-rigid motion, articulated motion, and camera motion. However, it does not handle discontinuous motion [6] and has poor performance on sequences with dynamic backgrounds (e.g. flowing water) [161].

Xie et al. (2019) [77] propose a method for instance motion segmentation using a Recurrent Neural Network (RNN), a type of artificial neural network that can be used to process sequential data. They first employ an encoder-decoder CNN, Y-Net, to compute the foreground

mask for each frame, given the frame and the forward optical flow map. An RNN then predicts embeddings of foreground pixel trajectories linked across time given the foreground masks. The trajectories are clustered into different foreground objects with the von Mises-Fisher mean shift algorithm [162]. This model achieves state of the art performance on several motion segmentation datasets [2] [47] [48] [76]. It can handle multiple objects, noise, occlusions, rigid / non-rigid motion, articulated motion, and camera motion. It does not use a CRF model to refine the results and can segment some novel objects in the test videos. However, the performance deteriorates if the foreground masks are poor. Cluster collapse can also cause multiple objects to be segmented as a single object.

Dave et al. (2019) [6] present a simple learning-based method for instance motion segmentation and tracking. The model combines motion cues from optical flow and appearance cues from RGB frame to segment moving objects. The motion cues help to separate the moving objects from each other, while the appearance cues provide objectness to capture the full extent of the objects. The model is based on the Mask R-CNN architecture [78] and has a motion stream and an appearance stream to extract spatiotemporal features using the two backbone networks. The features from the two streams are then fused by a region proposal network to predict proposals. The proposals are then fed to box and mask regression heads to predict the bounding boxes and the segmentation masks. This method surpasses all prior works on the FBMS-59 dataset [47] [48]. It can handle multiple objects, noise, occlusions, rigid / non-rigid motion, articulated motion, and camera motion. It can generalize to some never-before-seen objects. However, it may segment some static objects, such as cars and traffic lights.

Yang et al. (2019) [81] present an unsupervised method for motion segmentation, which does not require annotated data during training or test time. The model is an adversarial contextual model for segmenting moving objects in videos. It consists of a generator and an in-painter network which are deep CNNs with encoder-decoder design and skip connections. The generator predicts a mask of the moving object given the image and optical flow, while the in-

painter tries to in-paint back the optical flow masked out by the predicted mask from the generator. The in-painter and generator are trained jointly in an adversarial manner. This model achieves better or similar performance as unsupervised methods on DAVIS2016 (Densely Annotated Video Segmentation) [79], FBMS-59 [47] [48], and SegTrackV2 [80] dataset. DAVIS is a video segmentation dataset with pixel-level segmentation annotation. It also surpasses several methods which are pretrained on labelled datasets. It can handle multiple objects, occlusions, noise, rigid / non-rigid motion, articulated motion, and camera motion. It can generalize to new data without pretraining on large image segmentation datasets. However, it may segment other objects that are moved by primary objects (e.g. water moved by a surfer). The generator also requires optical flow between the current frame and some previous or subsequent frames as inputs.

Rashed et al. (2019) [82] propose a real-time CNN model, FuseMODNet, for motion segmentation under low-light conditions using motion information from camera and LiDAR (Light Detection and Ranging) sensors. LiDAR is a method that uses laser to measure distances. Moving object segmentation can be estimated from temporal information, such as optical flow. However, optical flow computation from camera sensor may fail under low illumination conditions. LiDAR sensors can work under low illumination. This model has a three-stream mid-fusion design that predicts the motion mask given RGB image, RGB optical flow, and LiDAR optical flow. This model is benchmarked on a novel dataset, Dark-KITTI, and standard KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) [82] and achieves an improved performance over models with other input combination and fusion methods. KITTI [92] is a real-world dataset of traffic scenarios for various tasks (e.g. optical flow estimation, depth estimation, segmentation). This method can handle multiple objects, low-lighting conditions, noise, rigid / non-rigid motion, and camera motion. For high illumination examples, a model without LIDAR optical flow as input is slightly better than a model using LIDAR optical flow. For far objects, the segmentation is less accurate as LIDAR flow becomes less accurate.

Yang et al. (2021) [83] present a modular network for instance motion segmentation from two frames with known camera intrinsic. This model utilizes geometric reasoning to decompose the scene into rigid background and moving rigid bodies, parametrized by 3D rigid transformations. It utilizes a two-stream CNN architecture to segment the rigid background and an arbitrary number of rigid bodies from 3D scene flow which is initially estimated from the two frames. This method achieves state of the art performance on rigid motion segmentation on KITTI and Sintel benchmarks. It can generalize to novel scenes and some never-before-seen objects. It can handle multiple objects, noise, occlusions, rigid motion, and camera motion. It can also handle various degenerate cases in motion segmentation when dynamic motion (i.e. motion of objects) is indistinguishable from camera motion given 2D optical flow (e.g. a car moving parallel to camera translation direction). However, it may segment static objects, such as traffic lights.

Neoral et al. (2021) [91] present a method for instance motion segmentation from three monocular video frames. They propose an end-to-end CNN model, Raptor, that combines features from a semantic branch and a motion branch to predict instance motion segmentation. The semantic backbone is built upon the ResNet-50 DetectorRS design [94], using a Recursive Feature Pyramid backbone with Switchable Atrous Convolutions. It is trained with a temporary Hybrid Task Cascade (HTC) head [95] on an object segmentation task to generalize better to unknown classes. The motion branch first produces motion cost volumes (MCVs) from depth, optical flow, optical expansion, and ego-motion estimate (inspired by [83]). The MCVs contain various features (e.g. reconstructed 3D scene points from optical flow). The MCVs are then fed to the motion backbone, which is built using the ResNet-18 DetectorRS design. It is also trained with a temporary HTC head on the motion segmentation task. The outputs from the semantic and motion branch are then concatenated and fed to an HTC head to predict instance motion segmentation. For the final training, both branches are fixed, while the final HTC head is trained. This method achieves state-of-the-art results on KITTI [92], DAVIS-Moving [6] [223], and

YTVOS-Moving (YouTube Video Object Segmentation) [6] [93] dataset. It generalizes well to a range of environments and datasets. It can handle multiple objects, noise, some occlusions, rigid / non-rigid motion, articulated motion, and camera motion. However, it may miss slow moving objects; objects far away from the camera; or objects close to a point of expansion. Ambiguous object instances (e.g. person with a bag) may be merged and static objects may be segmented. Errors may appear on significantly occluded moving objects (e.g. behind bars).

2.2 Motion Processing Hierarchy

Early works present numerous models for motion processing. Some models focus on a single cortical area [85]. Zemel and Sejnowksi (1998) [29], Simoncelli and Heeger (1998) [23], and Beardsley and Vaina (1998) [1] present models that explain some functions of MT and MST neurons, but they do not form a motion hierarchy.

There are also hierarchical models that can detect motion [85]. Meese and Anderson (2002) [20] argue that at least 8 spiral detecting mechanisms with direction bandwidths around 45° are needed to compute generalized spiral motions but does not provide a computationally plausible motion processing hierarchy that includes these. Giese and Poggio (2003) [8] present a biologically motivated complex hierarchy for processing human movement patterns. However, it does not include attention mechanisms and cannot process real image sequences as inputs. Lu and Sperling (1995) [18] describe a motion system with attentive processes, but the computation details are not included.

There are other models with attentive processes [85]. Nowlan and Sejnowski (1995) [21] and Daniilidis (1994) [5] are the earliest ones. In Nowlan and Sejnowski [21], motion energy is computed to model MT neurons. Local velocity selection uses Softmax activation for the neurons. The model does not include a full motion processing hierarchy, top-down component,

or binding of complex motion patterns. Experiments on attentional modulation in motion neurons are described in Treue and Maunsell (1996) [25], after their model is presented.

Daniilidis (1994) [5] computes 3D motion and structure from optical flow. An object is fixated to estimate ego-motion in the presence of translation and rotation of the observer from the flow in the log polar periphery. The goal is to compute the time to collision, rather than defining an attentive motion hierarchy. Attentive fixations are used, but the motion processing is based on log-polar representations and the connection to affine motion, implicated by the need to fixate, is not recognized.

Grossberg, Mingolla, and Viswanathan (2001) [10] present an integration and segmentation model for motion capture, the Formotion BCS model. The goal is to integrate motion data across the image and segment motion cues into a unified global percept. Models that process translation motion are employed in Area V1, V2, MT, and MST. The Secondary Visual Cortex (V2) receives data from V1 for further processing [177]. Neurons in V2 respond to various visual features, such as colour, shape, and motion. Motion patterns are not considered in the Foremotion BCS model. Competition determines local winners among neural responses and the MST cells with the winning direction can influence the MT cells tuned to the same direction. Various motion illusions are illustrated, but real image sequences are not attempted.

Tsotsos et al. (2005) [26] present a feed-forward motion processing pyramid, on which ST mechanisms are implemented to attend to motion and localize and label motion patterns. The model is the first to include a multi-level decomposition with local spatial derivatives of velocity. The input is a temporal sequence of grayscale images. The model includes multiple neural populations in the cortical areas V1, MT, MST, and 7a with neurons of three distinct speed selectivity types. A new solution to the feature binding problem is also proposed to group motion features into coherent object motion. Binding is achieved using a top-down selection mechanism. The model employs a realistic feature pyramid to extract motion features, but the filters are not trained.

Vintila and Tsotsos (2007) [26] later propose a motion estimation model, which extracts the velocity field from spatio-temporal input using quadrature filters, based on the method of orientation tensor formalism. This is the first stage of a two-stage process for attending to complex motion patterns. The second stage involves detection of complex motion patterns using the topology of the flow field. However, it is difficult to implement ST mechanisms within this model to attend to motion.

Steinmetz et al. (2022) [84] recently introduce a self-tuning mechanism that allows neurons to adapt to changing visual stimuli. This mechanism is built upon the principles of efficient sensory encoding [86]. The neural tuning curve parameters can be continually updated to encode the distribution of recently detected stimuli values. They implemented this mechanism in a neural model for self-motion heading estimation based on optical flow. The preferred speed of each speed cell is dynamically tuned, based on optical flow speeds of recent frames. This allows the network to remain sensitive under varying optic flow speeds. They found that the model with dynamic tuning has more accurate, shorter latency heading estimates than the model with static tuning.

2.3 Selective Tuning

Selective Tuning (ST) is a model of visual attention that is first described in Tsotsos (1990) [163]. ST is a computational model which includes a description of how attention is computed and can be tested with image inputs [85]. ST is a complex model, which has been under development for decades. A comprehensive review of ST can be found in Chapter 4 - 9 of Tsotsos (2011) [85].

The visual processing architecture has a hierarchical pyramid structure and consists of layers of neurons with decreasing spatial resolution [85]. The layers have increasing complexity

and abstraction of selectivity as one moves higher in the pyramid. Each neuron has both feed-forward and feedback connections within the network. Each neuron sends diverging output to neurons in the next layer and receives converging input from neurons in the previous layer. Each neuron also sends diverging feedback signals to lower layers and receives converging feedback signals from higher layers. Each neuron can have lateral connections to its neighbours.

In ST, when a stimulus is presented as input to the pyramid, the units within the pyramid are activated in a feedforward manner [85]. Once the feedforward process reaches the topmost layer, ST employs a hierarchy of winner-take-all (WTA) processes for TD attentional selection [85]. WTA is a parallel algorithm that finds the max value in a set [85]. WTA first runs across the entire visual field of the top layer to find the global winners (i.e. units with the strongest response). A global winner then activates a WTA process over its direct inputs to select the strongest responding region within its receptive field. Next, the feedforward connections from the input that do not contribute to the winner are pruned (i.e. inhibited). This helps to improve the signal to noise ratio of the input to the higher-level unit. This strategy is applied recursively within the pyramid to select winners in successively smaller receptive fields and prune away irrelevant neural connections. In the end, the cause of the strongest response (i.e. the global winner) is localized in the sensory field at earlier levels of the pyramid (e.g. input layer). The paths that remain form the pass zone while the pruned paths form the inhibitory zone of the attentional beam. ST represents a visual item (e.g. object, scene, event) using the entire network pathway of winners rather than some high level subset of neurons in the pyramid [164]. Thus, the representation can easily switch between high level abstractions (e.g. object category) and low level details (e.g. colour or texture).

2.4 Selective Tuning of Convolutional Neural Networks

Recent works include a TD attention model for CNNs based on a probabilistic WTA process and a library for Selective Tuning of CNNs with applications to various tasks, including object localization, object segmentation, and network compression.

Zhang et al. (2016) [165] present a TD attention model for CNN based on a probabilistic WTA process. It is realized by a novel Excitation Backprop scheme. TD signals are passed downwards in the network via the probabilistic WTA process, which utilizes TD and BU information to compute the winning probability of each neuron. Attention maps can then be generated for each layer of the network. They also introduce a contrastive TD attention formulation to improve the discriminativeness of the attention maps. It supresses irrelevant signals by cancelling out common winner neurons between two attention maps (e.g. elephant and non-elephant classifier). They implement Excitation Backprop for some common layers in CNNs, including ReLU (Rectified Linear Unit), Max Pooling, Average Pooling, Convolutional, and Full-connected layers. ReLU is a non-linear activation function that is used in artificial neural networks. Their methods are tested on some weakly-supervised localization tasks on MS COCO (Microsoft Common Objects in Context) [58], PASCAL VOC 2007 (PASCAL Visual Object Classes) [60], and ImageNet [167] dataset and achieve state of the art performance. MS COCO is a large-scale dataset for object detection, segmentation, and captioning. PASCAL VOC is an image dataset with pixel-level segmentation, bounding box, and object class annotations. Their methods produce accurate localization and demonstrate generalizability. They further evaluate their methods on the text-to-region association task on Flickr30k Entities dataset [166] by training a CNN image tag classifier. They achieve similar performance as fully supervised methods without any localization supervision or language model. They can localize both noun and verb phrases on the input image.

Biparva et al. (2017) [3] propose the Selective Tuning of Convolutional Neural Networks (STNet) which combines BU processing and attentive TD processing in a unified model. The BU process supports various basic operations including convolution, pooling, activation, and normalization functions. The attentive TD process propagates gating activities downward to compute gating volumes for each layer until the task requirements are fulfilled. STNet is then evaluated on the object localization task to predict bounding boxes of objects. STNet surpasses state of the art results on the weakly supervised localization task on the ImageNet benchmark dataset [7]. However, over-selection (e.g. mistakenly localize other accompanying objects besides the ground truth) or under-selection may occur for the bounding box predictions.

Biparva et al. (2020) [4] later apply ST-Net to the task of object segmentation. They present the Selective Segmentation Network (SSN). The model utilizes gating activities from the TD selection process and the BU hidden activities to train a network for object segmentation. The model is evaluated on benchmark datasets for semantic segmentation (PASCAL VOC 2012 [60], CamVid [172], and Horse-Cow dataset [173]) and outperforms the baseline model. It can also handle low levels of noise on the input image, including uniform noise, salt-pepper noise, and box-occlusion noise.

Biparva et al. (2020) [168] present an attentive pruning method based on STNet [3] to reduce the number of parameters of multi-layer neural networks using TD selection mechanisms. The TD selection process is added to a BU feedforward network to select important connections and prune away irrelevant ones for each layer. They evaluate the performance of this method on MNIST (Modified National Institute of Standards and Technology) [170] and CIFAR-10 (Canadian Institute for Advanced Research) [171] dataset using various CNN models. MNIST is a dataset of handwritten digits and CIFAR-10 is a dataset of colour images in 10 classes. The performance of this method on MNIST with LeNet models [169] are similar to previous baseline methods. They also test this method on 3 CNN

architectures on the CIFAR-10 dataset. They achieve negligible loss of accuracy under high compression ratios.

Chapter 3

ST-Motion-Net and Motion Pattern Detection

In this chapter, we present a fully convolutional motion processing hierarchy, ST-Motion-Net. We evaluate each area of the motion hierarchy on the motion pattern detection task using 2 datasets, Blender-MP and Blender-Complex.

3.1 ST-Motion-Net

ST-Motion-Net is a neurally-inspired model of the primate visual motion system (similar to [26] [27]). The motion processing goes through a set of stages with neural representations in areas V1, MT, MST, and 7a. It is inspired by what is known about monkey visual motion analysis [26] [129]. Each area provides input for the next area. Each area contains populations of neurons that respond to different motion features. Neurons in higher areas generally respond to more complex motion patterns and have larger receptive fields. ST-Motion-Net is a hierarchical feedforward neural network with neural populations in the 4 cortical areas for detecting different kinds of motion patterns. The model also supports learning from visual input, such as a temporal sequence of frames.

ST-Motion-Net has 4 areas, and each area feeds its outputs to the next area. Area V1, MT, MST, and 7a are modelled using fully convolutional networks [19] [22]. Thus, the model operates on an input of any size and outputs downsampled feature maps of the corresponding spatial dimensions. Smaller filters are used to reduce computation cost for training and the TD selection process (See 4.4). A stack of convolution layers with smaller filters is also more expressive than a convolution layer with large filters, as there are additional non-linearities

between the layers. Each area is trained using input and output feature maps prepared by the Data Preparation model (See 3.2) for the corresponding area. A summary of each area follows.

Neurons in area V1 are selective for a particular local direction and speed of motion in 5 main speed ranges (α and v). The input to this area is a temporal sequence of RGB frames. There are two kinds of neurons in area MT. The first kind responds to a particular local direction and speed of motion (similar to the neurons in V1). The second kind responds to a particular angle δ between local motion and speed gradient. The neurons in MT have larger receptive fields. The inputs to this area are the outputs from area V1. Neurons in area MST respond to translation (as in V1) and spiral motion patterns. Spiral motion patterns include expansion, contraction, clockwise rotation, and counterclockwise rotation, and combinations of these. These neurons have even larger receptive fields. The inputs to this area are the outputs from area MT. Neurons in area 7a respond to translation and spiral motion patterns (as in MST), and rotation (regardless of direction), radial motion (expansion or contraction), and deformations (normal and shear strain). Neurons that respond to rotation, radial motion, and deformations are denoted as AP. The neurons in 7a have the largest receptive fields. The inputs to this area are the outputs from area MST.

Figure 3.1 shows the full motion hierarchy. This shows the neural selectivities within each area of ST-Motion-Net. The remaining subsections of 3.1 will describe each area of ST-Motion-Net in turn.

3.1.1 Area V1

Area V1 receives visual input as a temporal sequence of RGB frames. The output models V1 neurons of 12 different directions (0˚, 30˚, 60˚, ..., 330˚) and 5 different speed selectivity types (s_1=0.0 pixels / frame, s_2=0.5 pixels / frame, s_3= 1.0 pixels / frames, …, s_5=2.0 pixels / frame). Thus, the outputs consist of 60 feature maps arranged by direction and speed. V1 neurons are

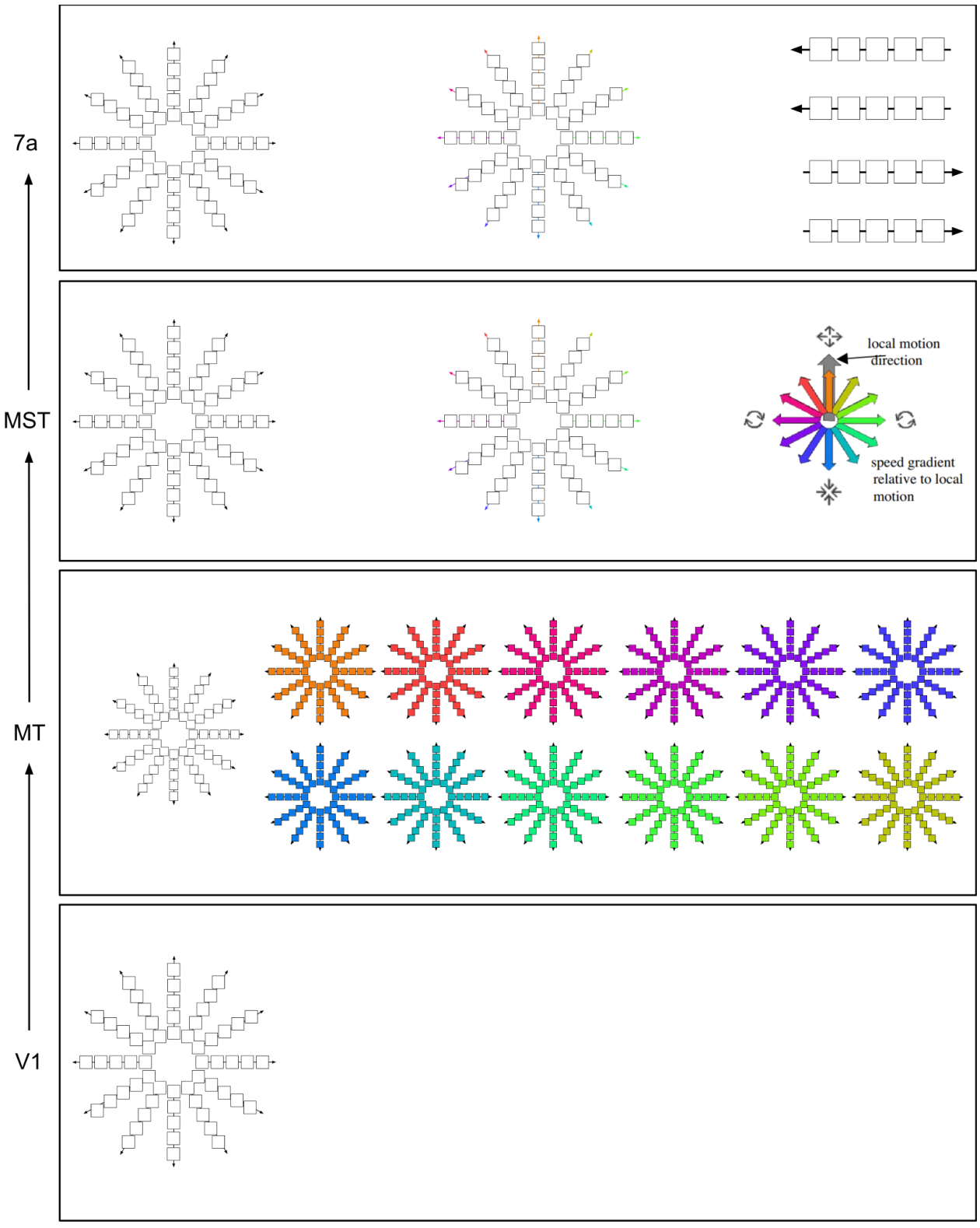

Figure 3.1: The full motion hierarchy. This shows the set of neural selectivities of ST-Motion-Net covering visual areas V1, MT, MST, and 7a. (Continued on the following page.)

Figure 3.1 (continued): Each rectangle indicates a single type of selectivity applied over the full image at the corresponding level of the hierarchy. Other than AP of 7a (top right), dark arrows indicate the local motion direction. The direction 0˚ is indicated by an upward pointing arrow. The five rectangles in each direction represent the 5 speed selectivity ranges in the model. Coloured rectangles in MT and arrows in MST / 7a indicate angles between motion and speed gradient. In area V1, the neurons that are selective for different local motion directions and speeds are represented by a single sheet of white rectangles (bottom left). In area MT, there are 13 sheets. The left one represents direction and speed selectivity, while the remaining 12 represent the 12 directions of speed gradient relative to the 12 motion directions. The wheel of coloured arrows (i.e. the legend) represents the colour coding for speed gradient relative to local motion. This codes the angle between local motion and speed gradient. MST units respond to translation motion (left sheet) and spiral motion (middle sheet). Spiral motion patterns include contraction, expansion, clockwise rotation, counterclockwise rotation, and combination of them. These are indicated by the wheel of coloured arrows in the corresponding directions of speed gradient relative to local motion direction. 7a units respond to translation motion (left sheet), spiral motion (middle sheet), plus radial motion, rotation, and deformations (shear and normal strain) on the right sheets without direction.

referred to as $V1_{\alpha, v, i}$.

As mentioned earlier, the input to Area V1 is a temporal sequence of frames. We explore 2 approaches from [16] to process a sequence of contiguous input frames in time. These approaches can be employed in CNNs to fuse information over the temporal dimension and learn spatio-temporal features.

Early Fusion. In Early Fusion, the input frames are merged in the first layer. This will collapse all temporal information in the first layer.

Slow Fusion. In Slow Fusion, the first layers slowly merge frames in groups smaller than the number of input frames. The fusion process continues until the temporal depth of the network reduces to 1.

We first experiment with a 2-frame model for Area V1 using early fusion. Each layer uses 2D convolution with filters that span over space and the channels of the input. The 2 input frames are merged in the first layer. The architecture of the 2-Frame-V1 model is shown below.

conv1_1	k3-n64-s1-p1 / ReLU
conv1_2	k3-n64-s2-p1 / ReLU
conv2_1	k3-n128-s1-p1 / ReLU
conv2_2	k3-n128-s2 / ReLU
conv3_1	k3-n256-s1-p1 / ReLU
conv3_2	k3-n256-s1-p1 / ReLU
conv4_1	k3-n512-s1-p1 / ReLU
conv4_2	k3-n512-s1-p1 / ReLU
fc5	k3-n1024-s1 / ReLU
fc6	k3-n1024-s1-p1 / ReLU
fc7	k3-n60-s1-p1 / Sigmoid

Layers are described by k (kernel size), n (number of channels), s (stride), and p (padding).

We then experiment with a 6-frame model for Area V1 using slow fusion. The slow fusion V1 model uses 3D Convolutions [14] [24] which convolve in both space and time and output tensor containing a temporal dimension. 2D convolutions are used to further process the feature maps once the temporal depth reduces to 1. The architecture of the multi-frame V1 model is shown below.

conv1_1	k(3,3)-n64-s(1,1)-p(0,1) / ReLU
conv1_2	k(1,3)-n64-s(1,2)-p(0,1) / ReLU
conv2_1	k(3,3)-n128-s(1,1)-p(0,1) / ReLU
conv2_2	k(1,3)-n128-s(1,2)-ReLU
conv3_1	k(2,3)-n256-s(1,1)-p(0,1) / ReLU
conv3_2	k(1,3)-n256-s(1,1)-p(0,1) / ReLU
conv1	k3-n512-s1-p1 / ReLU
conv2	k3-n512-s1-p1 / ReLU
fc1	k3-n1024-s1 / ReLU

fc2 k3-n1024-s1-p1 / ReLU
fc3 k3-n60-s1-p1 / Sigmoid

Layers are described by k (kernel size), n (number of channels), s (stride), and p (padding).

For 3D convolution layers, the kernel size, stride, and padding are described in the format: k (depth, height/width), s (depth, height/width), and p (depth, height/width).

3.1.2 Area MT

The input to Area MT consists of 60 output feature maps from Area V1 for translation motion. The output of Area MT consists of two groups of neurons. One group is tuned for a particular local speed and direction of movement (similar to V1 neurons). Thus, a translation neuron i with preferred direction of motion α and speed selectivity type v is referred as $MT_{\alpha, v, i}$. The other group of neurons respond to different angles between the direction of motion and local speed gradient. A speed gradient neuron i with preferred direction of motion α, preferred angle δ between direction of motion and speed gradient, and preferred speed v is referred as $MG_{\alpha, \delta, v, i}$. If in a region we constantly find the same angle δ between the direction of motion and speed gradient for all directions of motion α, then this indicates a particular motion pattern. The common motion patterns indicated by the angle are listed in Table 3.1.

δ (Angle between Direction of Motion and Speed Gradient)	**Motion Pattern**
0	Expansion
180	Contraction
90	Clockwise Rotation
270	Counterclockwise Rotation
Other	Combinations of Patterns

Table 3.1: Motion patterns indicated by angle δ between direction of motion and speed gradient in MG.

The output of Area MT consists of 780 feature maps (60 for MT and 720 for MG). The MT model uses 2D convolutions which convolve in space and outputs the translation and speed gradient feature maps. The architecture of Area MT is shown below.

conv1_1	k5-n128-s1-p2 / ReLU
conv1_2	k5-n128-s2-p2 / ReLU
conv2_1	k3-n256-s1-p1 / ReLU
conv2_2	k3-n256-s1-p1 / ReLU
conv3_1	k3-n512-s1-p1 / ReLU
conv3_2	k3-n512-s1-p1 / ReLU
fc4	k3-n1024-s1 / ReLU
fc5	k3-n780-s1-p1 / Sigmoid

Layers are described by k (kernel size), n (number of channels), s (stride), and p (padding).

3.1.3 Area MST

The input to area MST consists of 780 output feature maps from Area MT. Two groups of neurons are modelled: translation (similar to V1) and spiral motion neurons (clockwise and counterclockwise rotation, expansion, contraction, and combinations).

An MST translation neuron i with preferred direction of motion α, preferred speed selectivity of type v is referred to as $ST_{\alpha, v, i.}$ The directions of motion and speed selectivity types are the same as V1 neurons. An MST spiral motion neuron i selective for a pattern with angle δ (between the direction of motion and speed gradient of motion) and preferred speed v is referred as $SS_{\delta, v, i}$. The common motion patterns indicated by the angle are listed in Table 3.2 (See next page).

δ (Angle between Direction of Motion and Speed Gradient)	Motion Pattern
0	Expansion
180	Contraction
90	Clockwise Rotation
270	Counterclockwise Rotation
Other	Combinations of Patterns

Table 3.2: Motion patterns indicated by angle δ between direction of motion and speed gradient in SS.

The output of Area MST consists of 120 feature maps (60 for ST and 60 for SS). The MST model uses 2D convolutions and outputs the translation and spiral motion feature maps. The architecture of Area MST is shown below.

conv1_1 k3-n128-s1-p1 / ReLU
conv1_2 k3-n128-s2-p2 / ReLU

conv2_1 k3-n256-s1-p1 / ReLU
conv2_2 k3-n256-s1 / ReLU

conv3_1 k3-n512-s1-p1 / ReLU
conv3_2 k3-n512-s1 / ReLU

fc4 k3-n1024-s1 / ReLU
fc5 k3-n120-s1-p1 / Sigmoid

Layers are described by k (kernel size), n (number of channels), s (stride), and p (padding).

3.1.4 Area 7a

The input to Area 7a consists of 120 output feature maps from MST. Area 7a involves at least 6 types of computations: translation and spiral motion (similar to MST), rotation (clockwise or counterclockwise irrespective of direction), radial motion (expansion or contraction regardless of direction), and deformations (normal strain ρ and shear strain σ). Additional neurons are modelled for deformations, which are not implemented in Tsotsos et al. (2005) [26].

Translation and spiral motion neurons are referred to as $AT_{\alpha, v, i}$ and $AS_{\delta, v, i}$. Neuron ART responds to rotation, regardless of direction of rotation (clockwise or counterclockwise). An ART neuron i with speed selectivity type v is referred to as $ART_{v, i}$. Neuron ARD responds to radial motion, regardless of whether it is expansion or contraction. An ARD neuron i with speed selectivity type v is defined as $ARD_{v, i}$. Neuron ADR responds to deformation from normal strain regardless of direction. An ADR neuron i with speed selectivity type v is referred to as $ADR_{v, i}$. Neuron ADS responds to deformation from shear strain regardless of direction. An ADS neuron i with speed selectivity type v is referred to as $ADS_{v, i}$. We denote the feature maps for ART, ARD, ADR, and ADS as AP.

The output of Area 7a consists of 140 feature maps (60 for AT, 60 for AS, 5 for ART, 5 for ARD, 5 for ADR, and 5 for ADS). The 7a model uses 2D convolutions and outputs the translation, spiral motion, and AP feature maps. The architecture of Area 7a is shown below.

conv1_1	k3-n128-s1-p1 / ReLU
conv1_2	k3-n128-s2 / ReLU
conv2_1	k3-n256-s1-p1 / ReLU
conv2_2	k3-n256-s1-p1 / ReLU
conv3_1	k3-n512-s1-p1 / ReLU
conv3_2	k3-n512-s1-p1 / ReLU
fc4	k3-n1024-s1-p1 / ReLU
fc5	k3-n140-s1-p1 / Sigmoid

Layers are described by k (kernel size), n (number of channels), s (stride), and p (padding).

3.2 Data Preparation

In this section, we describe a method to prepare the GT feature maps for training and testing for each area of ST-Motion-Net. We prepare the GT feature maps for each area from GT optical flow maps. We use Gaussian filters, oriented gradient filters, and spatial derivatives of velocity

to generate GT feature maps for the neuron responses of each area. The preparation of the ground truth is similar to the computations of the motion hierarchy in [26]. Since GT optical flow is used to generate GT feature maps for Area V1, the ground truth for V1 is more accurate than V1 outputs from the spatiotemporal filters in [26], which can be quite noisy. Speed gradients or spatial derivatives of velocity are also computed from GT optical flow. Thus, the GT feature maps for each area are more accurate than the outputs using [26]. The feature maps prepared from GT optical flow can be chosen as ground truth.

For the training data, we prepare GT feature maps for the ChairsSDHom extended dataset [12] which consists of images of chairs moving with random affine motion and very small displacements on random backgrounds. We also prepare GT feature maps for sequences from the HighSpeedSintel dataset [13] to finetune the 6-Frame Area V1 model. For benchmark datasets, GT feature maps for each area are also be prepared with this method. GT optical flow is available for each pair of image frames. The details of the data preparation method are described below.

Given a sequence of n input RGB frames of even length, $\mathrm{I}_1, \mathrm{I}_2, \ldots, \mathrm{I}_\mathrm{n}$, we denote the corresponding ground-truth optical flow maps by $\mathrm{F}_{1\to2}, \mathrm{F}_{2\to3}, \ldots, \mathrm{F}_{\mathrm{n}-1\to\mathrm{n}}$. We prepare the ground truth feature maps for V1 from $\mathrm{F}_{\mathrm{c}\to\mathrm{c}+1}$. Here, c is the index of the first frame of the 2 centre frames. $\mathrm{F}_{\mathrm{c}\to\mathrm{c}+1}$ is first downsampled using Gaussian filters to compute flow map for Area V1, denoted by $\mathrm{F}^{\mathrm{V1}}_{\mathrm{c}\to\mathrm{c}+1}$. We use a 11 x 11 Gaussian filter with $\sigma = 1.0$ to compute $\mathrm{F}^{\mathrm{V1}}_{\mathrm{c}\to\mathrm{c}+1}$.

We then compute the speed and angle feature maps, denoted by $\mathrm{V}^{\mathrm{V1}}_{\mathrm{c}\to\mathrm{c}+1}$ and $\mathrm{A}^{\mathrm{V1}}_{\mathrm{c}\to\mathrm{c}+1}$ from the corresponding flow map $\mathrm{F}^{\mathrm{V1}}_{\mathrm{c}\to\mathrm{c}+1}$. We use the speed and angle feature maps, $\mathrm{V}^{\mathrm{V1}}_{\mathrm{c}\to\mathrm{c}+1}$ and $\mathrm{A}^{\mathrm{V1}}_{\mathrm{c}\to\mathrm{c}+1}$ to prepare the GT Translation motion feature maps for Area V1.

$$\mathrm{V1}_{\alpha,\,\mathrm{v},\,\mathrm{i}} = \begin{cases} 1, & \left(\left(\left|\alpha - \mathrm{A}_\mathrm{i}'^{\mathrm{V1}}\right| \le \theta_1\right) \vee \left(\mathrm{V}_\mathrm{i}'^{\mathrm{V1}} < \theta_2\right)\right) \wedge \left(\left|\mathrm{v} - \mathrm{V}_\mathrm{i}'^{\mathrm{V1}}\right| \le \theta_3\right) \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

θ_1, θ_2, and θ_3 in Equation 3.1 are thresholds. We set $\theta_1 = 15^\circ$, $\theta_2 = 0.1$, and $\theta_3 = 0.25$.

$$A_i'^{V1} = \begin{cases} 0^\circ, & A_i^{V1} > 360^\circ - \theta_1 \\ A_i^{V1}, & \text{otherwise} \end{cases} \tag{3.2}$$

$$V_i'^{V1} = \begin{cases} 2.0, & V_i^{V1} > 2.0 + \theta_3 \\ V_i^{V1}, & \text{otherwise} \end{cases} \tag{3.3}$$

We then prepare feature maps for MT, ST, and AT by downsampling using Gaussian filters.

$$MT_{\alpha, v, i} = \sum_{j \in R_i} W_j V1_{\alpha, v, j} \tag{3.4}$$

Here, W_j is a 5 x 5 Gaussian filter with $\sigma = 1.0$.

$$ST_{\alpha, v, i} = \sum_{j \in R_i} W_j MT_{\alpha, v, j} \tag{3.5}$$

Here, W_j is an 11 x 11 Gaussian filter with $\sigma = 1.0$.

$$AT_{\alpha, v, i} = \sum_{j \in R_i} W_j ST_{\alpha, v, j} \tag{3.6}$$

Here, W_j is a 3 x 3 Gaussian filter with $\sigma = 1.0$.

For ground-truth MG, SS, and AS feature maps, we first compute speed gradient feature maps in Area MT using gradient filters oriented in 12 gradient directions.

$$G_{\beta, i}^{MT} = \sum_{j \in R_i} O_{\beta, j} V_j^{V1} \tag{3.7}$$

$O_{\beta,j}$ in Equation 3.7 is the gradient filter in direction β that leads to the maximum gradient if speed increases in direction β (See Figure 3.2). If $\beta \neq 0^{\circ}, 90^{\circ}, 180^{\circ}, 270^{\circ}$, we normalize the gradient filter by a factor of 12. Otherwise, we normalize the gradient filter by a factor of 10.

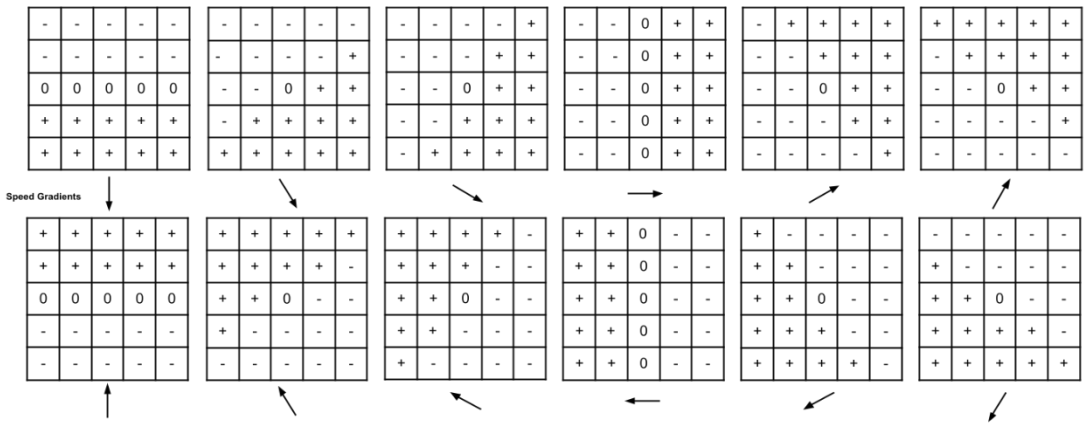

Figure 3.2: Oriented Gradient Filters corresponding with $O_{\beta,j}$ in the equation.

The feature maps for MG are then generated using MT and speed gradient feature maps.

$$MG_{\alpha,\delta,v,i} = MT_{\alpha,v,i} * C_0 * \text{clip}\left(G^{MT}_{\alpha+\delta,i},\ 0.0,\ \infty\right) \tag{3.8}$$

Here, $C_0 = 4.5$ is a scaling constant.

We then downsample from MG feature maps using Gaussian filters to generate spiral motion feature maps for MST and 7a.

$$SS_{\delta,v,i} = \sum_{j \in R_i} \sum_{\alpha} W_j \, MG_{\alpha,\delta,v,j} \tag{3.9}$$

Here, W_j is an 11 x 11 Gaussian filter with $\sigma = 1.0$.

$$AS_{\delta,v,i} = \sum_{j \in R_i} W_j \, SS_{\delta,v,j} \tag{3.10}$$

Here, W_j is a 3 x 3 Gaussian filter with $\sigma = 1.0$.

To generate ground truth AP feature maps, we first compute the spatial derivatives of velocity feature maps for Area MT from x and y component of V1 flowmap using gradient filters, denoted by S_x and S_y.

$$F_{xx}^{MT}{}_i = \sum_{j \in R_i} S_{x_j} F_x^{V1}{}_j \tag{3.11}$$

$$F_{xy}^{MT}{}_i = \sum_{j \in R_i} S_{y_j} F_x^{V1}{}_j \tag{3.12}$$

$$F_{yx}^{MT}{}_i = \sum_{j \in R_i} S_{x_j} F_y^{V1}{}_j \tag{3.13}$$

$$F_{yy}^{MT}{}_i = \sum_{j \in R_i} S_{y_j} F_y^{V1}{}_j \tag{3.14}$$

$$S_x = \frac{1}{20}\begin{bmatrix} -1 & -1 & 0 & +1 & +1 \\ -1 & -1 & 0 & +1 & +1 \\ -1 & -1 & 0 & +1 & +1 \\ -1 & -1 & 0 & +1 & +1 \\ -1 & -1 & 0 & +1 & +1 \end{bmatrix} \tag{3.15}$$

$$S_y = \frac{1}{20}\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ +1 & +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 & +1 \end{bmatrix} \tag{3.16}$$

We then compute the motion pattern feature maps at MT scale using Area MT translation motion feature maps and spatial derivative of velocity feature maps.

$$ART_{v,i}^{MT} = \left| F_{xy}^{MT}{}_i - F_{yx}^{MT}{}_i \right| * C_1 * VF_{v,i} \tag{3.17}$$

$$\mathrm{ARD}_{v,i}^{MT} = \left|\mathrm{F}_{xx}^{MT}{}_{i} + \mathrm{F}_{yy}^{MT}{}_{i}\right| * C_1 * \mathrm{VF}_{v,i} \tag{3.18}$$

$$\mathrm{ADR}_{v,i}^{MT} = \left|\mathrm{F}_{xx}^{MT}{}_{i} - \mathrm{F}_{yy}^{MT}{}_{i}\right| * C_1 * \mathrm{VF}_{v,i} \tag{3.19}$$

$$\mathrm{ADS}_{v,i}^{MT} = \left|\mathrm{F}_{xy}^{MT}{}_{i} + \mathrm{F}_{yx}^{MT}{}_{i}\right| * C_1 * \mathrm{VF}_{v,i} \tag{3.20}$$

Here, $C_1 = 4.5$ is a scaling constant.

$\mathrm{VF}_{v,i}$ in Equation 3.17 - 3.20 is the speed detection feature map which detects motion of speed v regardless of direction of motion.

$$\mathrm{VF}_{v,i} = \sum_{\alpha} \mathrm{MT}_{\alpha,v,i} \tag{3.21}$$

We then downsample the motion pattern feature maps at MT scale using Gaussian filters to compute motion pattern feature maps at MST scale.

$$\mathrm{ART}_{v,i}^{MST} = \sum_{j \in R_i} W_j \, \mathrm{ART}_{v,j}^{MT} \tag{3.22}$$

$$\mathrm{ARD}_{v,i}^{MST} = \sum_{j \in R_i} W_j \, \mathrm{ARD}_{v,j}^{MT} \tag{3.23}$$

$$\mathrm{ADR}_{v,i}^{MST} = \sum_{j \in R_i} W_j \, \mathrm{ADR}_{v,j}^{MT} \tag{3.24}$$

$$\mathrm{ADS}_{v,i}^{MST} = \sum_{j \in R_i} W_j \, \mathrm{ADS}_{v,j}^{MT} \tag{3.25}$$

Here, W_j is an 11 x 11 Gaussian filter with $\sigma = 1.0$.

We finally downsample the motion pattern feature maps at MST scale to compute AP feature maps.

$$\mathrm{ART}_{v,i} = \sum_{j \in R_i} W_j \, \mathrm{ART}_{v,j}^{MST} \tag{3.26}$$

$$\mathrm{ARD}_{v,i} = \sum_{j \in R_i} W_j \, \mathrm{ARD}_{v,j}^{MST} \tag{3.27}$$

$$\mathrm{ADR}_{v,i} = \sum_{j \in R_i} W_j \, \mathrm{ADR}_{v,j}^{MST} \tag{3.28}$$

$$\mathrm{ADS}_{v,i} = \sum_{j \in R_i} W_j \, \mathrm{ADS}_{v,j}^{MST} \tag{3.29}$$

Here, W_j is a 3 x 3 Gaussian filter with $\sigma = 1.0$.

3.3. Training

As mentioned earlier, the GT feature maps are prepared from GT optical flow maps. They are more accurate than the computations from the hierarchy in [26]. Thus, the ground truth can be used to train ST-Motion-Net to produce an improved hierarchy. We train each area of ST-Motion-Net using Adam [17] with default parameters. For each convolution layer in an area, the learnable parameters include the filters and a set of biases. Other than finetuning of 6-Frame-V1, each area is trained from scratch. We use Xavier Normal initialization [9] for the output layer of each area. We use He Normal initialization [11] for other layers. Since the output feature maps of each area contain strengths of detecting motion patterns between 0 and 1, we train each area using binary cross-entropy loss. We apply 2D Dropout regularization [90] on the output layers (name starts with fc in the architecture) of each model, except for the final layer.

Training of the 2-Frame-V1 model is done for 16 epochs with a batch size of 16 on sequences prepared from ChairsSDHom extended dataset [12] with an initial learning rate of 1e-4. Learning rate is decreased by a factor of 0.4 after each epoch. The 6-Frame-V1 model is first pretrained on sequences from ChairsSDHom extended dataset [12] for 16 epochs with a

learning rate of 1e-4. Since each sequence in this dataset has only 2 frames, we repeat the 2 centre frames to extend each input sequence to 6 frames. We pretrain the 6-Frame-V1 model with ground truth to predict translation motion on the 2 center frames. This helps the model to learn some motion features which can be finetuned on a 6-frame dataset. On the 6-frame dataset, the pretrained model converges faster than a 6-frame model trained from scratch, as the model has already learned some relevant motion features which are useful for the task. The 6-frame model trained from scratch does not converge. We finetune the pretrained model on 6-frame input sequences from the HighSpeedSintel dataset [13] for 32 epochs with a learning rate of 1e-5. Both pretraining and finetuning of the model are done with a batch size of 16.

Area MT, MST, and 7a are trained with a batch size of 16 and a learning rate of 1e-4 on the corresponding GT feature maps prepared from ChairsSDHom extended dataset [12]. Training of Area MT is done for 16 epochs and training of Area MST is done for 120 epochs. Area 7a is trained for 60 epochs.

We perform data augmentation by random cropping on the ChairsSDHom [12] sequences. For the HighSpeedSintel [13] sequences, the orientations of the objects are more limited and there are less motions with zero speed. Therefore, we perform augmentation by random cropping, horizontal flip, vertical flip, and still Gaussian noise patches (0-3 patches with size up to 250 x 250) on the HighSpeedSintel [13] sequences. Table 3.3 shows the motion pattern detection performance on the validation set of each benchmark dataset (See 3.5.1), after each augmentation option is chosen in order. This prepares 250 x 250 input patches for training Area V1; 60 x 60 input feature maps for training Area MT; 28 x 28 input feature maps for training Area MST; and 9 x 9 input feature maps for training Area 7a.

During data preparation, we repeat the data generation process 5 times over the ChairsSDHom [12] sequences and 20 times over the HighSpeedSintel [13] sequences. This prepares around 100000 sequences for each training set. For each training set, we use 10% of the data for the train-validation set.

Data Augmentation Option	Blender-MP Val	Blender-Complex Val
Random Crops (250 x 250)	0.053512	0.063403
Horizontal and Vertical Flips	0.048682	0.059754
Gaussian Noise Patches (0-3)	**0.048606**	**0.059747**

Table 3.3: Performance of 6-Frame-V1 on the motion pattern detection task (See 3.4) on the Validation set of Blender-MP and Blender-Complex dataset (See 3.5.1) after each data augmentation option is chosen in order, during finetuning on HighSpeedSintel dataset. We report the overall performance of 6-Frame-V1 on each validation set using the root mean squared error (RMSE) metric.

3.4 Motion Pattern Detection Task

In this section, we describe the task in terms of input, output, and evaluation metric.

Input. Assume a temporal sequence of images is given. Input to the motion processing hierarchy is a continuous subsequence of T images.

Output. For each input subsequence to the model, output feature maps for each area of the motion processing hierarchy should be produced after the feedforward process (See example for area V1 in Figure 3.3). These feature maps contain the responses of the neurons for each area and indicate the motion categories that are present on the input subsequence to the model. The detected motion categories can be determined using WTA during attention initialization (See 4.2).

3.4.1 Evaluation Metric

For the motion pattern detection task, it is important to have neuron responses that are close to the GT neuron responses for each area of the motion hierarchy as the detected motion categories are determined from the computed responses. For performance evaluation, we

measure the root mean square error (RMSE) between computed and GT responses of the neurons, $\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(r_{t,i} - r_{o,i})^2}{N}}$, for each area of the motion processing hierarchy.

Here, $r_{t,i}$ and $r_{o,i}$ are the computed and GT responses of a neuron.

N is the total number of neuron responses for the area in the dataset.

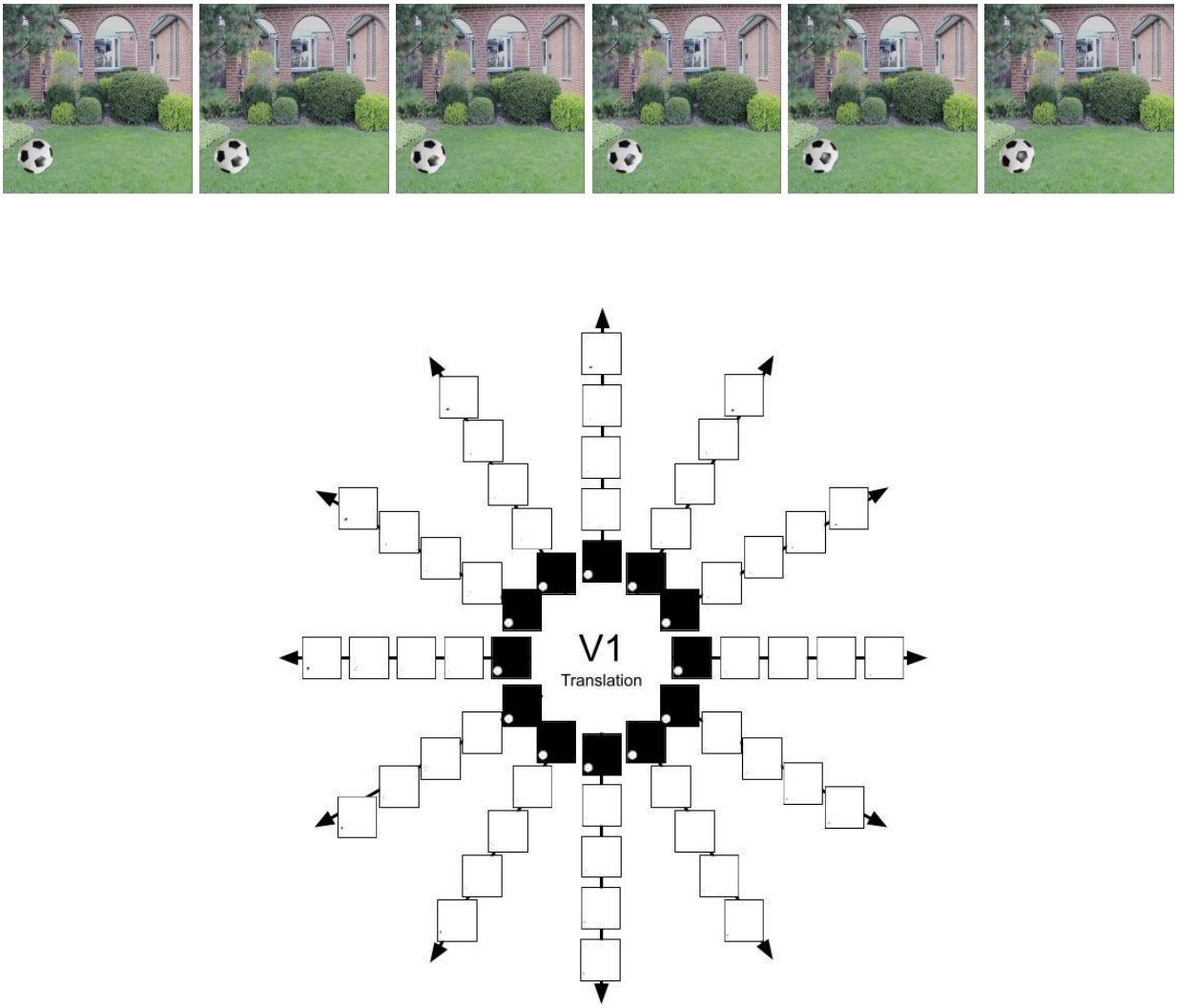

Figure 3.3: Sequence of a textured soccer ball rotating counterclockwise and the corresponding GT output feature maps for Area V1. The innermost ring of feature maps shows the detections of zero speed (v = 0.0) motion patterns. Darker pixel values on the feature maps indicate stronger responses, while white means zero response.

3.5. Experiments and Results

We evaluate each area of ST-Motion-Net on the motion pattern detection task with 2 datasets, Blender-MP and Blender-Complex. The two datasets are created in Blender [37] to test the motion hierarchy on motion of geometric and real-world objects, respectively.

3.5.1 Datasets

We create two datasets in Blender [37] to evaluate the motion hierarchy. The first dataset contains sequences of textured geometric objects moving on still backgrounds with different motion patterns, including translation, radial motion, rotation, and deformation. Shadows or illumination on the objects may change. GT output feature maps for each area, motion pattern localization masks, motion segmentation, and camera intrinsics are available for each sequence. We call this dataset the Blender Motion Pattern (Blender-MP) dataset.

For the Blender-MP dataset, we create 20 sequences of length 10 for both the validation and test set. 5 overlapping input subsequences are generated from each sequence. Thus, the validation and test dataset each contains 100 input subsequences. Once the 6-frame input subsequences are created, we generate the 2-frame input subsequences by removing 2 input frames from the start and end of each 6-frame input subsequence.

The second dataset contains sequences of real-world objects [98] – [119] moving on still backgrounds with different motion patterns. More complex motion patterns, such as mixture of translation, radial motion, and rotation may also be present. Some objects may overlap during motion and may exit or re-enter the scene. Shadows or illumination on the moving objects may also change. Ground truth data are also available for each sequence. We call this dataset the Blender-Complex dataset.

The Blender-Complex dataset contains 25 sequences of length 1000. We use 4 sequences for validation and 21 sequences for testing. 995 overlapping input subsequences are generated from each sequence. After the 6-frame input subsequences are created, we also generate the 2-frame input subsequences by removing 2 input frames from the start and end of each 6-frame input subsequence.

GT optical flow maps are generated using the VisionBlender tool [15]. For the motion pattern detection task, GT feature maps for each area of the motion hierarchy are prepared from GT optical flow maps using the method described in 3.2.

3.5.2 Motion Pattern Detection Benchmark

We evaluate each area of the motion hierarchy on the motion pattern detection task given input feature maps and GT output feature maps for the area from each sequence of the Blender-MP and Blender-Complex dataset.

Each area is evaluated using the root mean squared error (RMSE) metric between the output feature maps of the area and the GT output feature maps (See 3.4.1). We also show the mean squared error (MSE) for each sequence. The performance results for 2-Frame Area V1, 6-Frame Area V1, Area MT, Area MST, and Area 7a are given in Table 3.4 using the evaluation metrics.

Blender-MP Val Sequence		2-Frame-V1	6-Frame-V1	MT	MST	7a
0	Wooden Sphere	0.00216	0.00249	0.00021	0.00102	0.00575
1	Small Stony Cube 1	0.00114	0.00092	0.00008	0.00021	0.00090
2	Leafy Cylinder	0.00211	0.00297	0.00012	0.00032	0.00098
3	Rock	0.00114	0.00130	0.00014	0.00040	0.00133
4	Grassy Ring	0.00266	0.00293	0.00023	0.00045	0.00118
5	Leafy Rectangle	0.00171	0.00200	0.00022	0.00017	0.00029
6	Grassy Cone	0.00170	0.00182	0.00032	0.00009	0.00018
7	Grassy Cylinder	0.00212	0.00282	0.00040	0.00118	0.00290
8	Wooden Polyhedron	0.00204	0.00253	0.00019	0.00071	0.00318

9	Gray Diamond	0.00134	0.00127	0.00029	0.00025	0.00037
10	Grassy Rectangle 1	0.00096	0.00167	0.00008	0.00029	0.00092
11	Stony Ellipsoid 1	0.00130	0.00142	0.00010	0.00046	0.00185
12	Wooden Rectangle 1	0.00363	0.00321	0.00023	0.00076	0.00235
13	Grassy Ellipsoid	0.00146	0.00158	0.00020	0.00071	0.00119
14	Leafy Parallelepiped	0.00276	0.00279	0.00025	0.00102	0.00226
15	Wooden Rectangle 2	0.00281	0.00255	0.00010	0.00044	0.00213
16	Stony Sphere	0.00202	0.00364	0.00023	0.00048	0.00157
17	2 Gray Stars	0.00296	0.00361	0.00027	0.00034	0.00040
18	Stony Polyhedron 1	0.00135	0.00262	0.00020	0.00015	0.00045
19	Stony Ring	0.00206	0.00311	0.00014	0.00013	0.00036
MSE		0.00197	0.00236	0.00020	0.00048	0.00153
RMSE		0.04441	0.04861	0.01417	0.02192	0.03908

Blender-MP Test Sequence		**2-Frame-V1**	**6-Frame-V1**	**MT**	**MST**	**7a**
0	Large Stony Sphere	0.00199	0.00234	0.00007	0.00029	0.00301
1	Grassy Cube	0.00179	0.00222	0.00008	0.00037	0.00212
2	Large Rock	0.00214	0.00253	0.00031	0.00121	0.00540
3	Stony Cone and Sphere	0.00272	0.00309	0.00017	0.00052	0.00090
4	Stony Polyhedron 2	0.00226	0.00286	0.00018	0.00055	0.00145
5	Wooden Cube	0.00330	0.00336	0.00016	0.00079	0.00147
6	Stony Cone	0.00218	0.00236	0.00048	0.00047	0.00105
7	Small Stony Cube 2	0.00296	0.00339	0.00063	0.00155	0.00144
8	Wooden Cylinder	0.00247	0.00258	0.00023	0.00072	0.00243
9	Clay Teapot	0.00218	0.00241	0.00044	0.00097	0.00205
10	Grassy Rectangle 2	0.00270	0.00318	0.00037	0.00151	0.00558
11	Stony Ellipsoid 2	0.00219	0.00332	0.00023	0.00063	0.00351
12	Grassy Parallelepiped	0.00211	0.00233	0.00015	0.00106	0.00189
13	Reflective Ellipsoid	0.00331	0.00345	0.00023	0.00175	0.00340
14	Large Gassy Parallelepiped	0.00512	0.00356	0.00063	0.00134	0.00285
15	Wooden Star	0.00101	0.00135	0.00008	0.00025	0.00057
16	Stony Square	0.00154	0.00151	0.00023	0.00068	0.00205
17	Leafy Star and Stony Star	0.00390	0.00620	0.00027	0.00047	0.00061
18	Small Stony Sphere	0.00160	0.00098	0.00004	0.00013	0.00085
19	Stony Cube	0.00110	0.00110	0.00019	0.00016	0.00124
MSE		0.00243	0.00270	0.00026	0.00077	0.00219
RMSE		0.04929	0.05201	0.01611	0.02777	0.04683

Blender-Complex Val Sequence		**2-Frame-V1**	**6-Frame-V1**	**MT**	**MST**	**7a**
1	Airplane	0.00049	0.00045	0.00003	0.00001	0.00004
11	Wooden Toy Car	0.00191	0.00225	0.00012	0.00014	0.00030
19	Cow Balloon	0.00266	0.00343	0.00025	0.00046	0.00097
21	Two Butterflies	0.00498	0.00815	0.00062	0.00092	0.00184
MSE		0.00251	0.00357	0.00025	0.00038	0.00079
RMSE		0.05012	0.05975	0.01594	0.01959	0.02809

Blender-Complex Test Sequence		2-Frame-V1	6-Frame-V1	MT	MST	7a
2	Hover Car	0.00217	0.00230	0.00011	0.00015	0.00046
3	Rainbow Hot Air Balloon	0.00074	0.00070	0.00004	0.00004	0.00012
4	Drone	0.00086	0.00124	0.00008	0.00012	0.00025
5	Paper Airplane	0.00148	0.00169	0.00009	0.00017	0.00042
6	Eagle	0.00170	0.00199	0.00013	0.00019	0.00043
7	Butterfly	0.00220	0.00293	0.00020	0.00029	0.00057
8	Seagull	0.00254	0.00398	0.00029	0.00042	0.00068
9	Crab Balloon	0.00223	0.00208	0.00018	0.00029	0.00055
10	Soccer Ball	0.00124	0.00221	0.00016	0.00017	0.00039
12	Toy Airplane	0.00136	0.00246	0.00012	0.00022	0.00047
13	Volleyball	0.00299	0.00283	0.00014	0.00014	0.00069
14	Shopping Cart	0.00277	0.00448	0.00025	0.00039	0.00065
15	Sword	0.00221	0.00322	0.00015	0.00026	0.00058
16	Rusted Tire	0.00231	0.00297	0.00021	0.00033	0.00072
17	Fighter Jet	0.00067	0.00069	0.00003	0.00002	0.00007
18	Wooden Wheel	0.00396	0.00535	0.00057	0.00061	0.00078
20	Flying Saucer	0.00172	0.00304	0.00013	0.00019	0.00049
22	Soccer Ball and Volleyball	0.00242	0.00298	0.00020	0.00037	0.00096
23	Two Drones	0.00419	0.00596	0.00034	0.00052	0.00110
24	Two Hot Air Balloons	0.00283	0.00346	0.00020	0.00025	0.00070
25	Cow Balloon and Penguin Balloon	0.00373	0.00396	0.00029	0.00053	0.00089
MSE		0.00221	0.00288	0.00019	0.00027	0.00057
RMSE		0.04696	0.05367	0.01365	0.01640	0.02388

Table 3.4: Performance of different areas of ST-Motion-Net on the Validation and Test set of Blender-MP and Blender-Complex dataset on the motion pattern detection task. We show the mean-squared error (MSE) for each sequence. We measure the overall performance of an area using the root mean squared error (RMSE) metric.

The 2-Frame Early Fusion Area V1 model achieves a better performance than the 6-Frame Slow Fusion Area V1 model. The RMSE metric is lower on both the validation and test set after using the 2-Frame-V1 model. The data augmentation options (See Table 3.3) improve the performance of 6-Frame-V1 significantly from overfitting. 2-Frame-V1 and 6-Frame-V1 also have roughly the same number of parameters. Thus, the remaining gap in performance is likely due to overfitting of the data on the additional input frames. The additional frames may also introduce inconsistencies in the motion patterns, which can affect the model's ability to learn the motion patterns on the 2 central frames. Since 3D Convolution has not been implemented in the

ST Library [3] at this time, we experiment with motion pattern localization and motion segmentation on ST-Motion-Net in Chapter 4 and 5 using the 2-Frame-V1 model.

3.5.3 Qualitative Results

Next, we show some qualitative results for each area on the motion pattern detection task. Figure 3.4 shows a sequence from Blender-MP Dataset containing a textured shrinking sphere on a still background. We compare the GT outputs, 2-frame model outputs, and 6-frame model outputs in Figure 3.5. Close-up views of the 2-frame and 6-frame model outputs are available in Figure A.1. Darker pixel values on the output feature maps indicate stronger responses; white means zero response.

Figure 3.4: Sequence from Blender-MP of a contracting textured sphere on a still background. Inputs to 2-Frame Area V1 are the 2 center frames and inputs to 6-Frame Area V1 are the 6 frames.

The outputs from the 2-frame model of ST-Motion-Net are slightly stronger for each area and contain less errors, as the 2-Frame-V1 model is more accurate than the 6-Frame-V1 model (See 3.5.2). For translation motion outputs, both the 2-frame and 6-frame model detect the local translation motion patterns on the shrinking sphere in all directions at different speeds. Both the 2-frame and 6-frame model successfully detect the contracting sphere on the spiral motion outputs at $\delta = 180°$. The AP outputs of the 2-frame and 6-frame models also detect the radial motion of the shrinking sphere with relatively strong response.

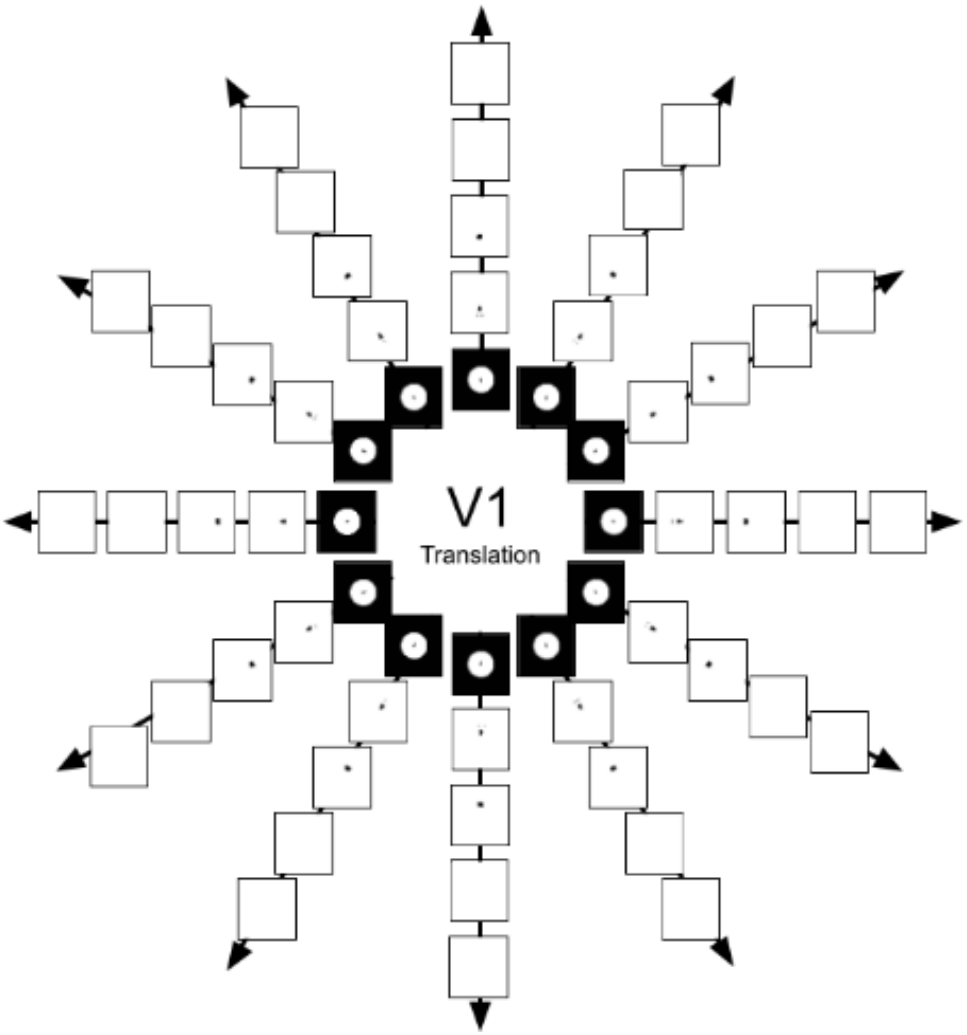

Figure 3.5: Output of area computations of ST-Motion-Net for sequence in Figure 3.4. V1 Translation shows the translation output of Area V1. This shows a close-up view of the GT output for V1 Translation.

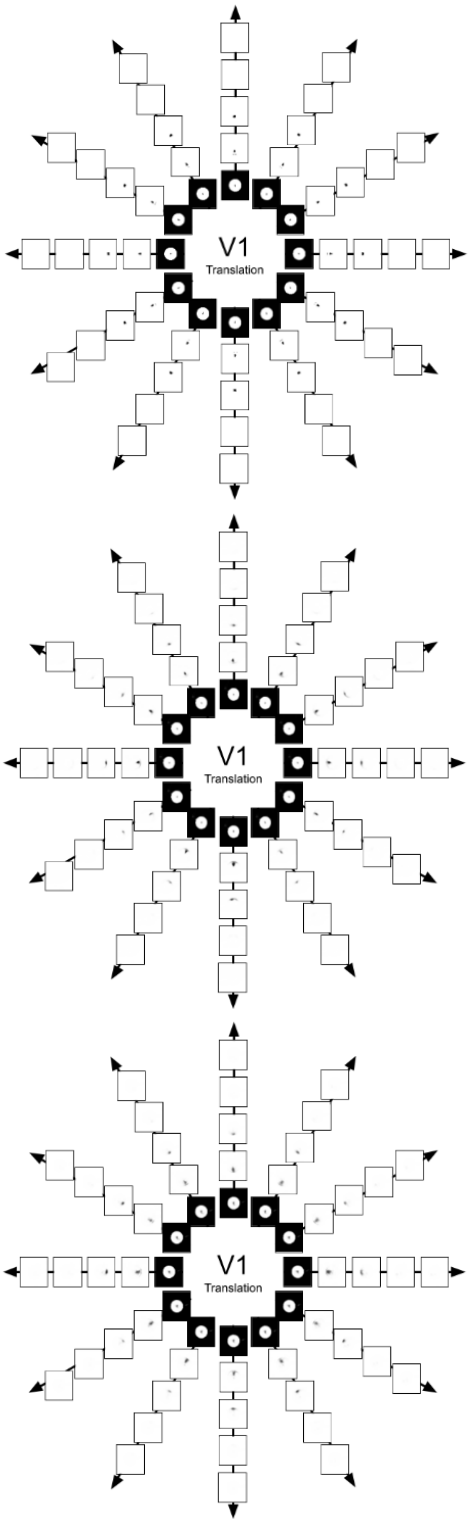

Figure 3.5: Top to bottom: GT output, 2-frame model output, and 6-frame model output. Outputs from 2-frame model are slightly stronger for each area and contain less errors.

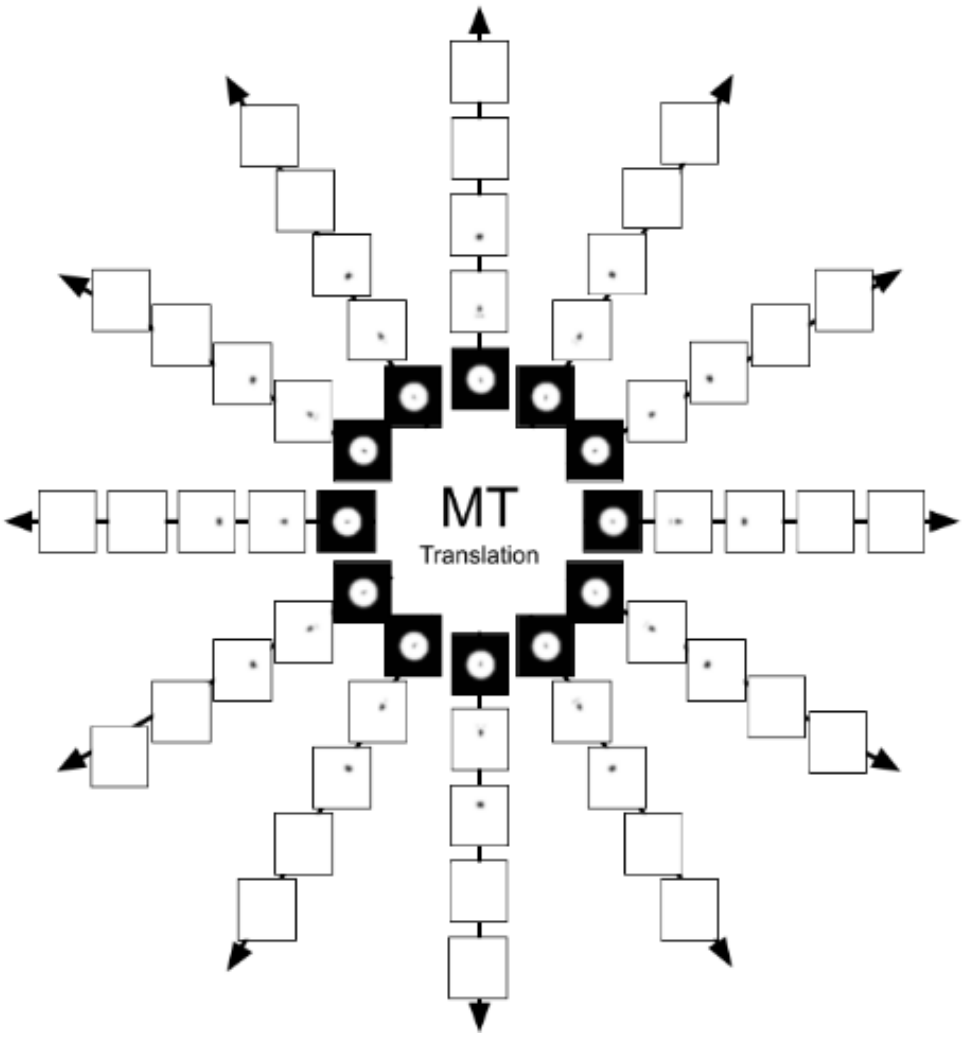

Figure 3.5 (continued): MT Translation shows the translation output of Area MT. This shows a close-up view of the GT output for MT Translation.

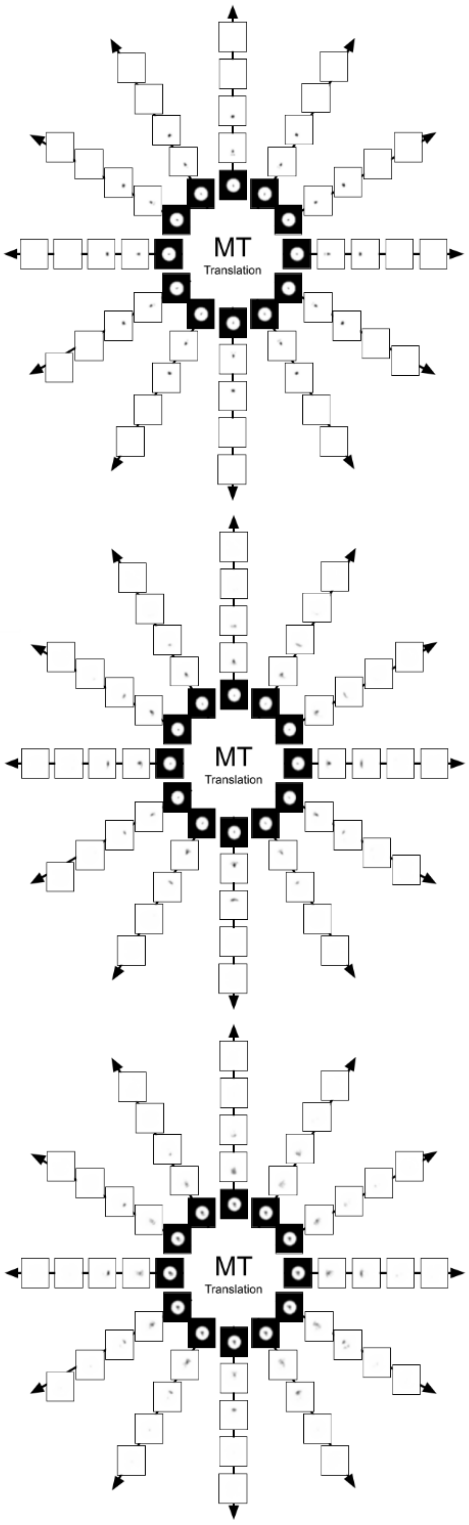

Figure 3.5 (continued): Both the 2-frame and 6-frame model detect local translation motions on the shrinking sphere in all directions at different speeds.

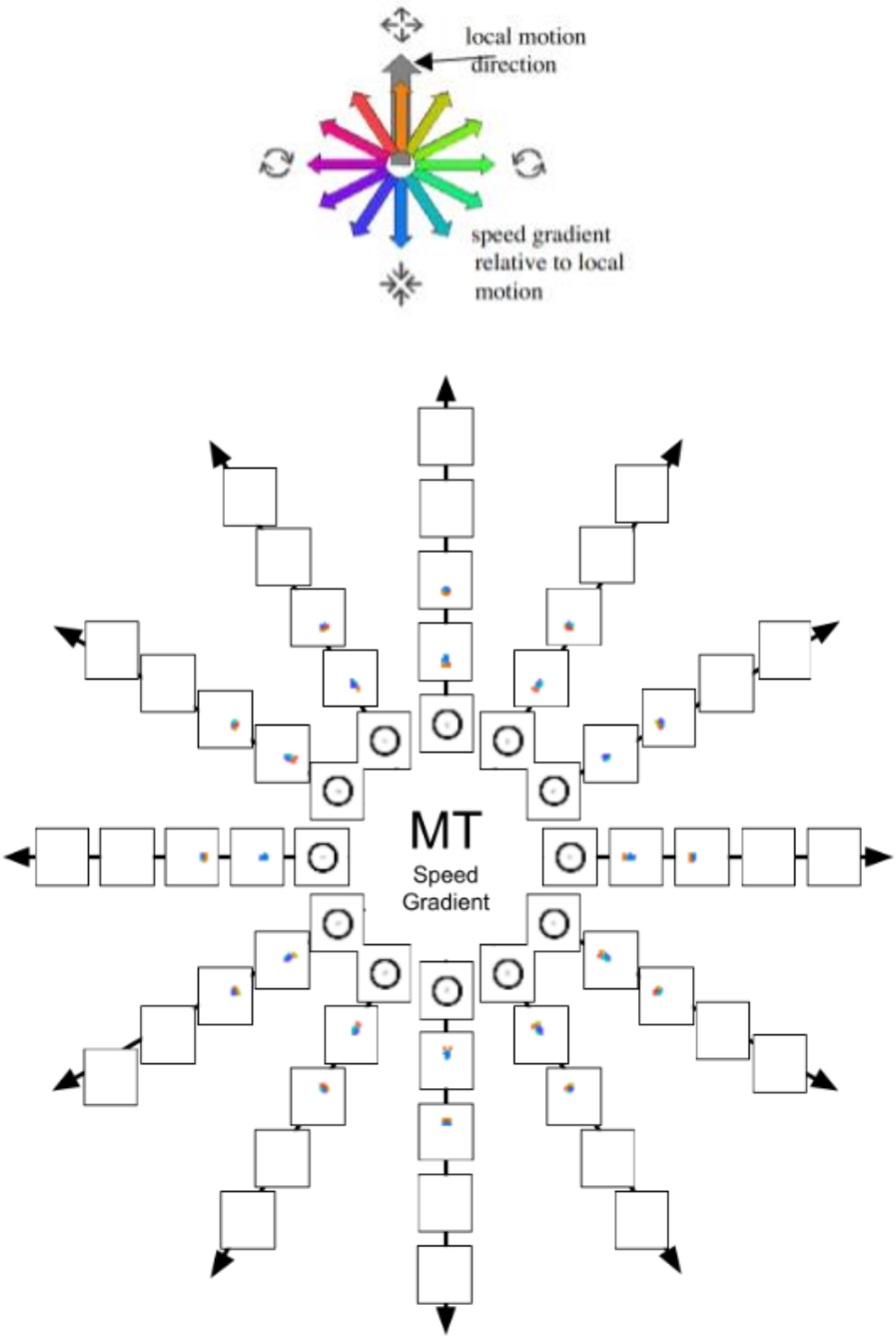

Figure 3.5 (continued): Speed gradient outputs in Area MT are shown in a summary representation of the 12 speed gradients for each local direction and speed. Each coloured dot is the maximum value across the 12 representations. This shows a close-up view of the GT output for MT Speed Gradients in the summary representation.

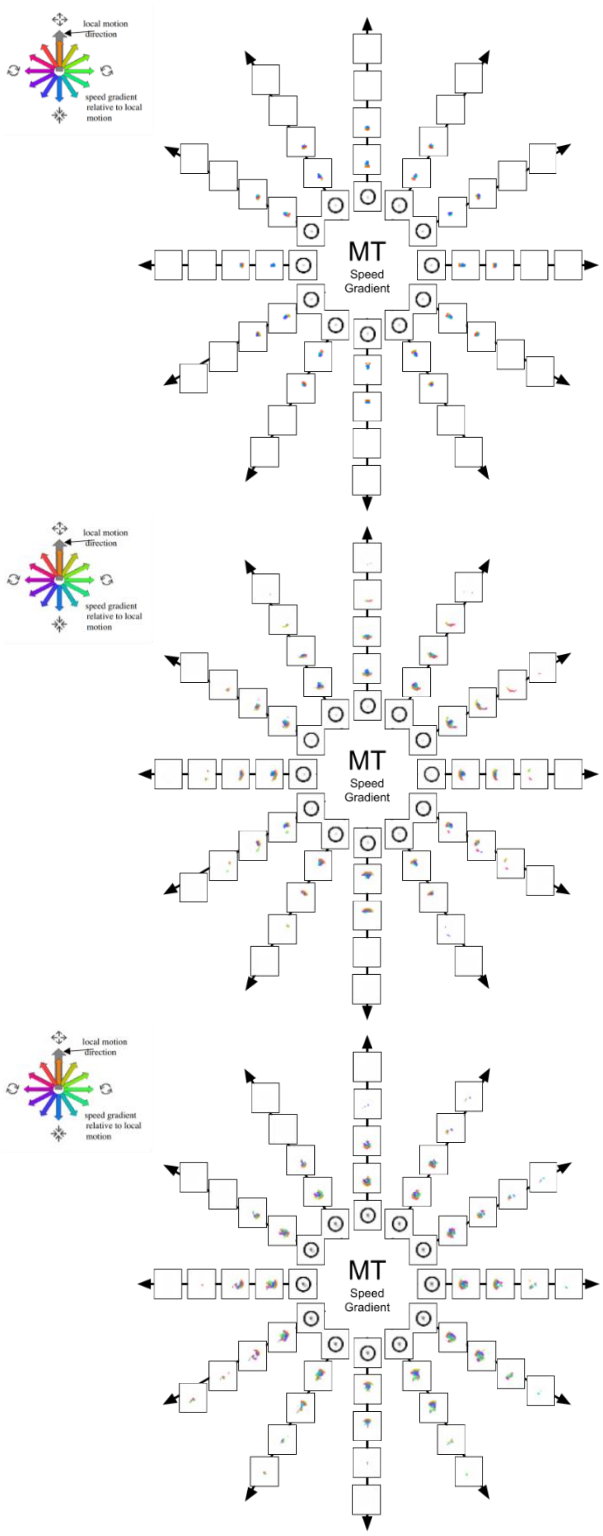

Figure 3.5 (continued): Contraction (roughly blue) is detected on the sphere by both the 2-Frame and 6-Frame models (Zoom in for details).

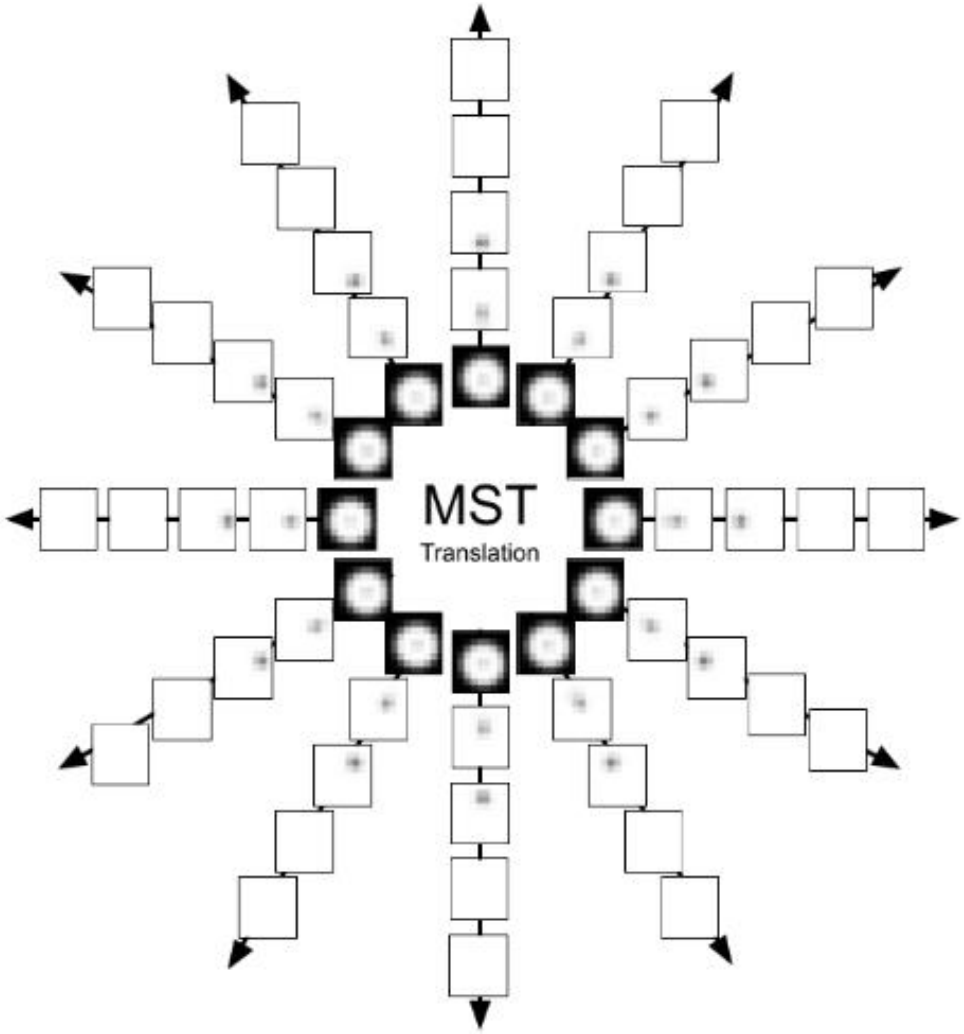

Figure 3.5 (continued): MST Translation shows the translation output in Area MST. This shows a close-up view of the GT output for MST Translation.

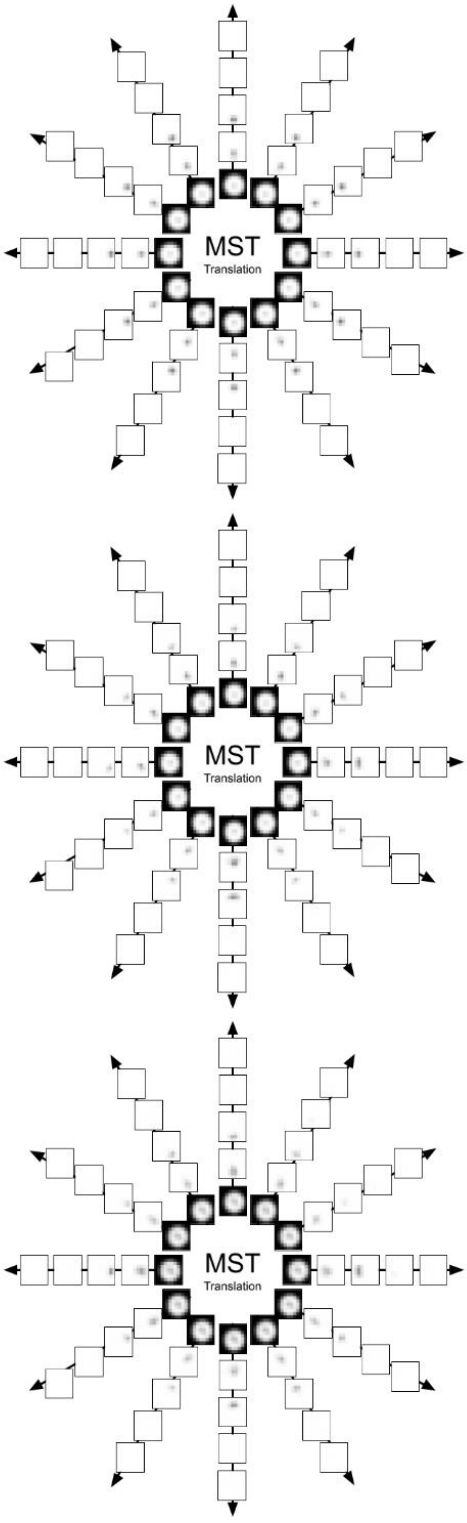

Figure 3.5 (continued)

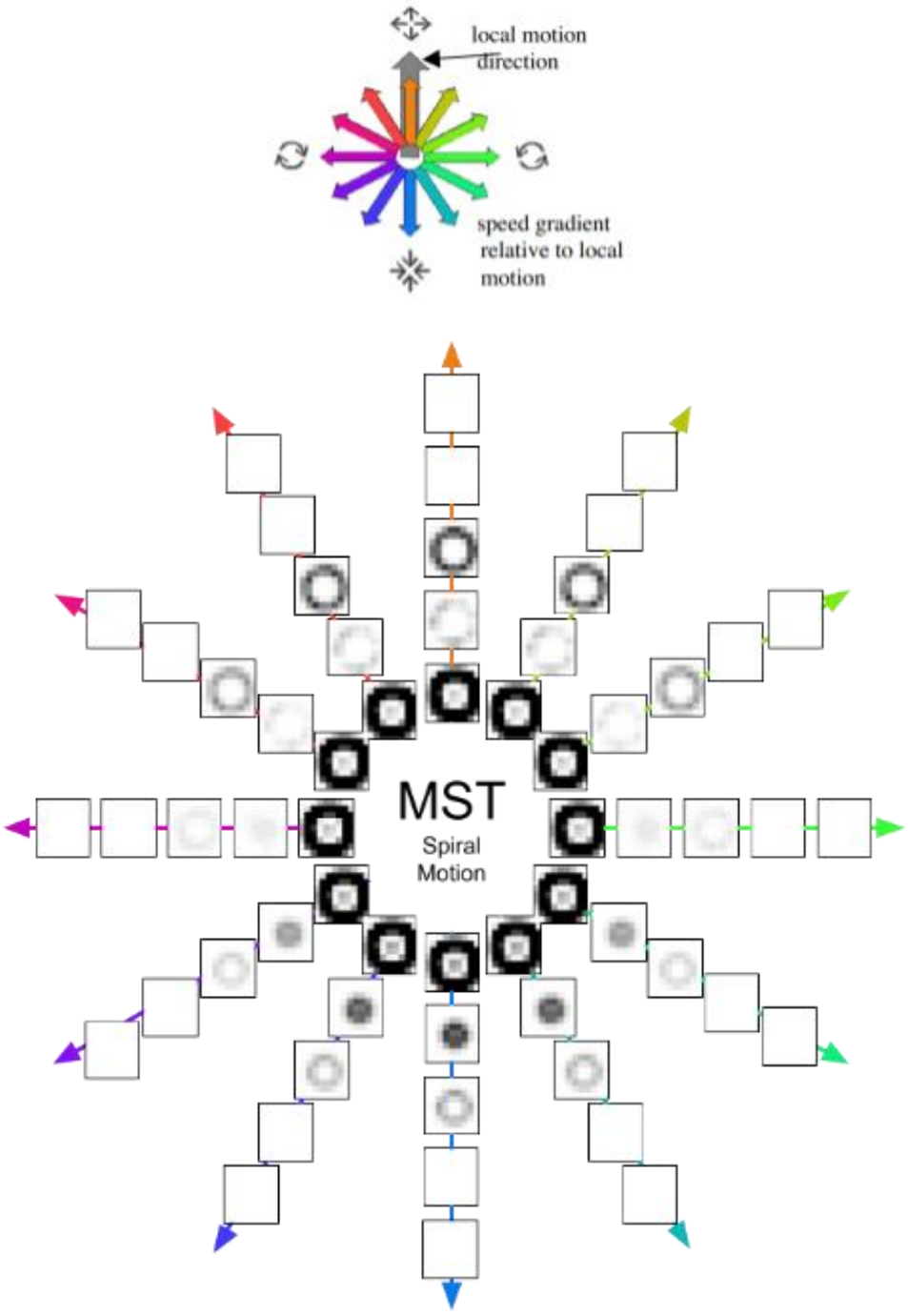

Figure 3.5 (continued): MST Spiral Motion shows the generalized spiral output in Area MST. Coloured arrows indicate the inner angle δ between motion and speed gradient (See legend). This shows a close-up view of the GT output for MST Spiral Motion.

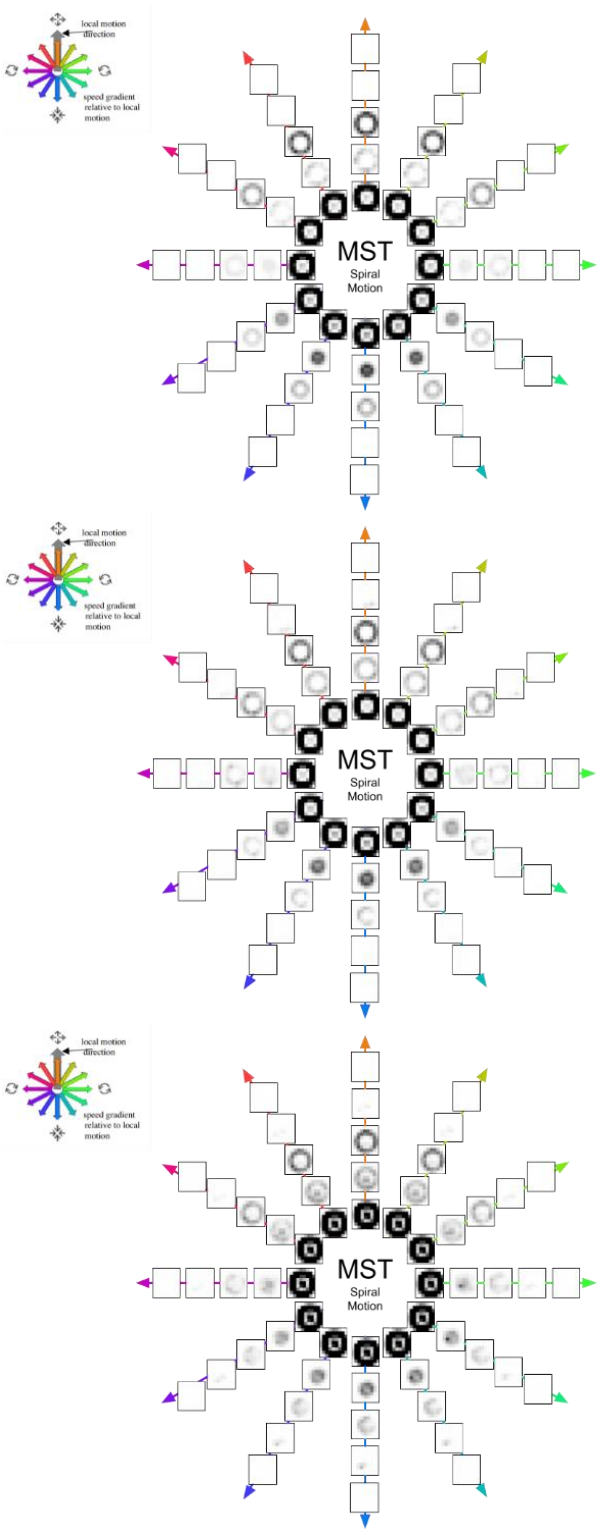

Figure 3.5 (continued): Both the 2-frame and 6-frame model successfully detect the contracting sphere on the spiral motion outputs at δ=180˚.

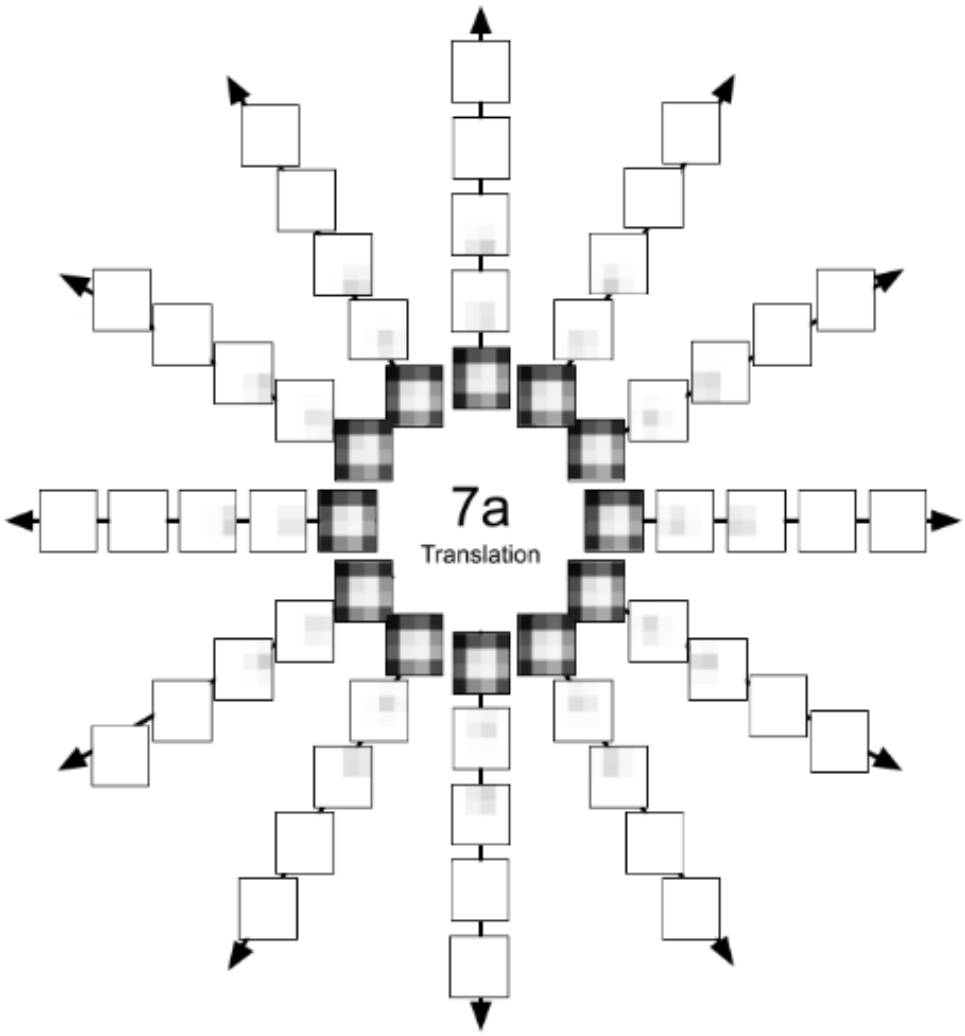

Figure 3.5 (continued): 7a Translation shows the translation output in Area 7a. This shows a close-up view of the GT output for 7a Translation.

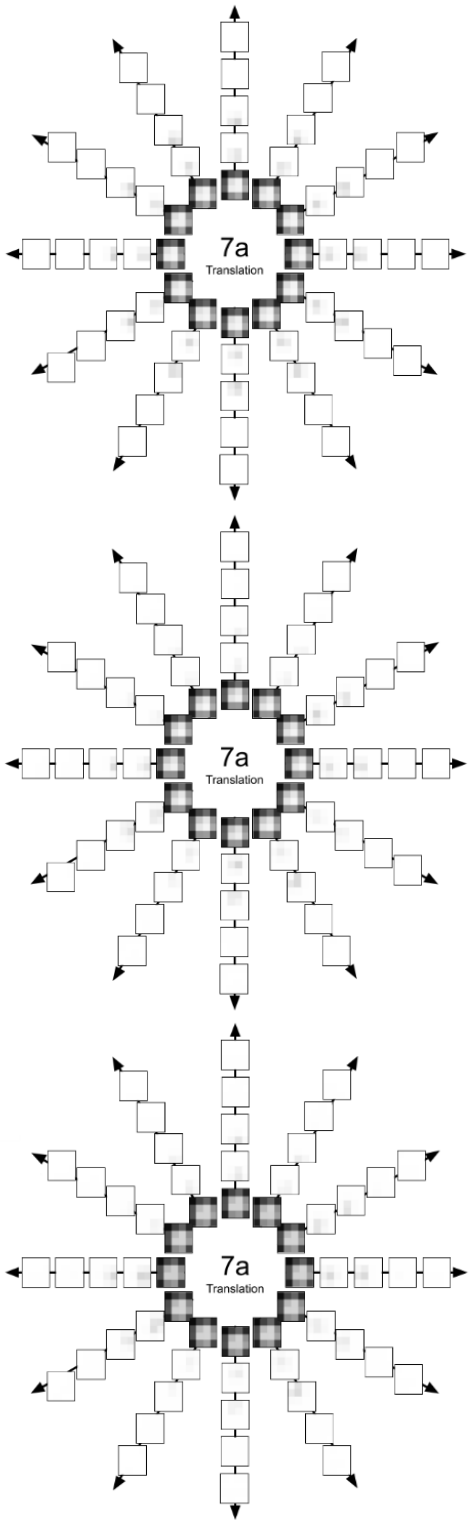

Figure 3.5 (continued)

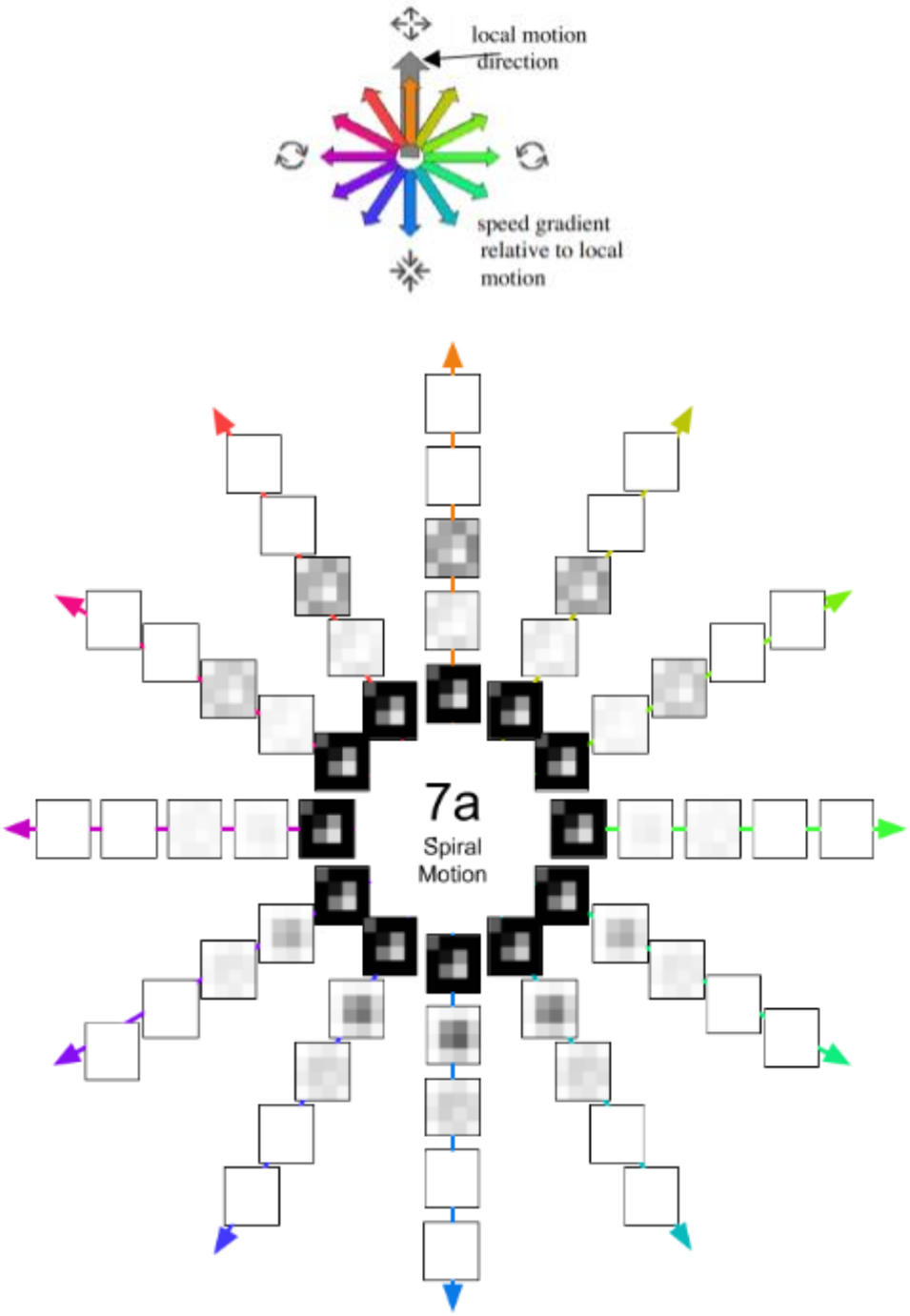

Figure 3.5 (continued): 7a Spiral Motion shows the generalized spiral output in Area 7a. Coloured arrows indicate the inner angle δ between motion and speed gradient (See legend). This shows a close-up view of the GT output for 7a Spiral Motion.

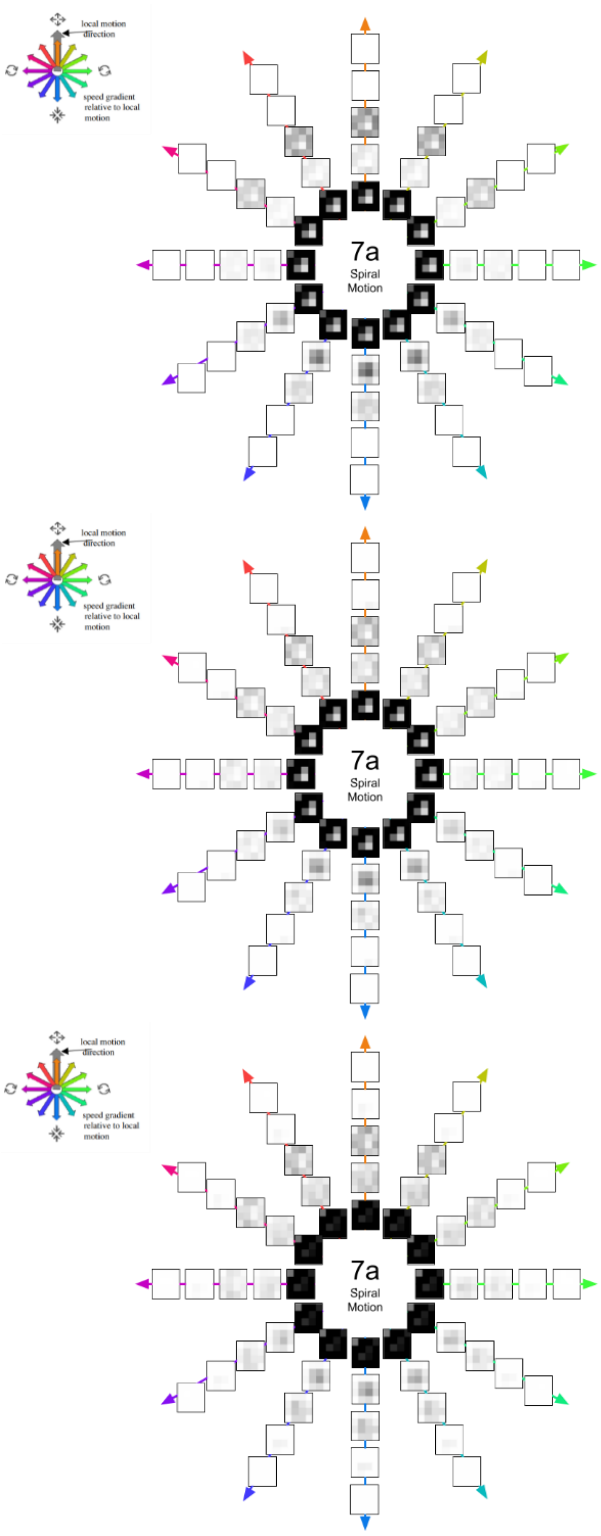

Figure 3.5 (continued): Both the 2-frame and 6-frame model successfully detect the contracting sphere on the spiral motion outputs at δ=180˚.

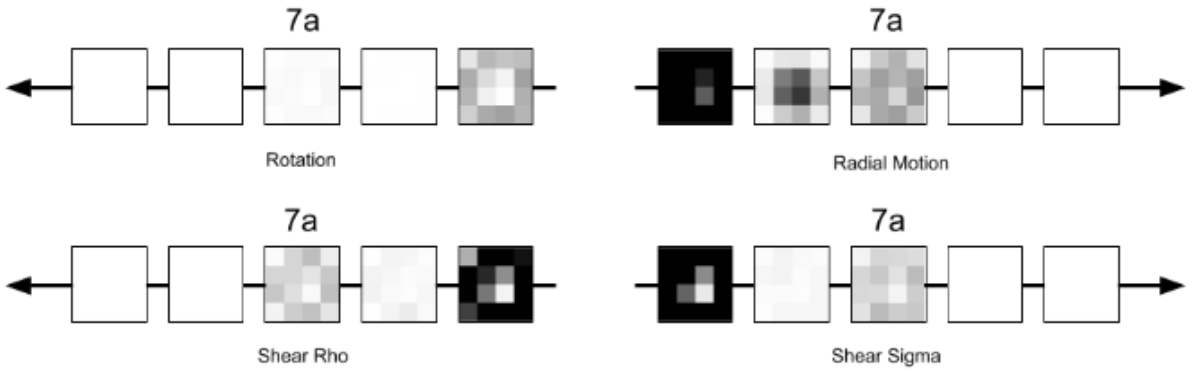

Figure 3.5 (continued): Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the GT outputs for AP.

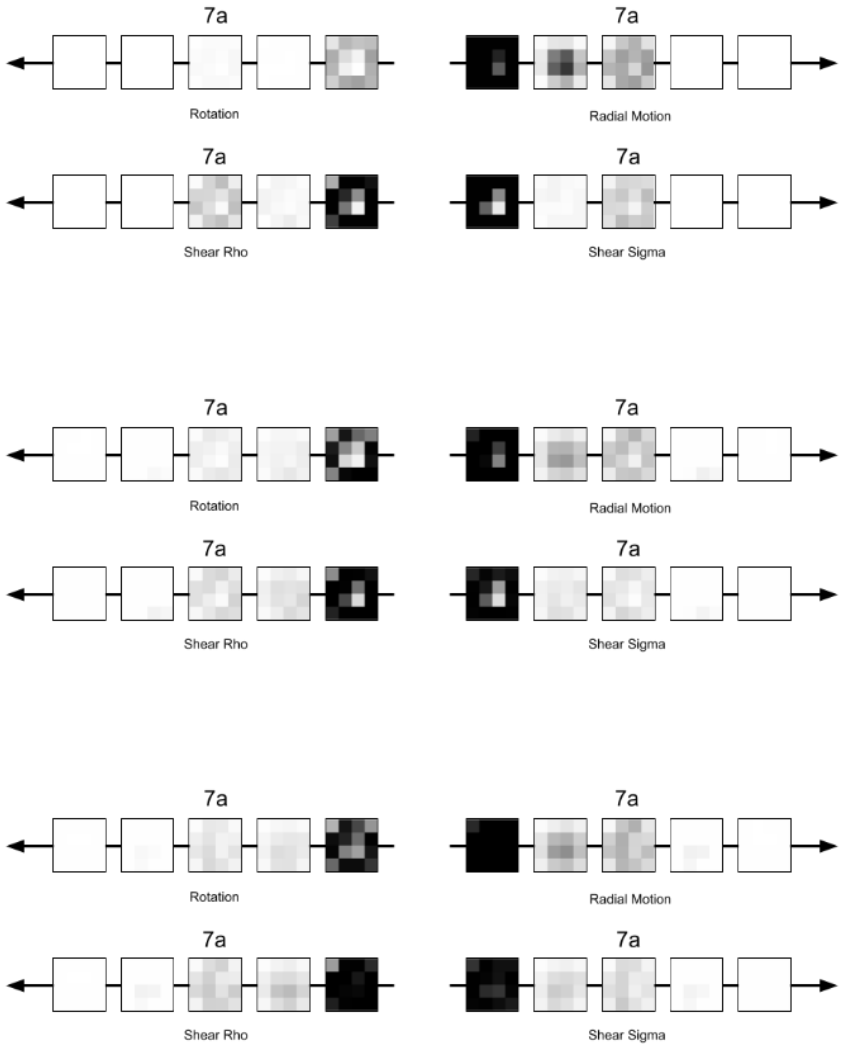

Figure 3.5 (continued): Both the 2-Frame and 6-Frame model detect radial motion of the shrinking sphere with relatively strong response.

Figure 3.6 shows a sequence from Blender-Complex Dataset containing two rotating hot air balloons on a still background. The left one is rotating clockwise and the right one is rotating counterclockwise. A shadow is expanding on the left one, while the right one has a very slight contraction. We compare the GT outputs, 2-frame model outputs, and 6-frame model outputs from AP in Figure 3.7. Close-up views of the 2-frame and 6-frame model outputs are available in Figure A.2.

Figure 3.6: Sequence from Blender-Complex of a clockwise rotating (left) and a counterclockwise rotating (right) hot air balloon. Some expansions or contractions are also present on the balloons. Inputs to 2-Frame Area V1 are the 2 center frames and inputs to 6-Frame Area V1 are the 6 frames.

Both the 2-frame and 6-frame model detect translation motions on the 2 balloons. The detections tend to become weaker in higher areas, as the output feature maps become coarser. Both the 2-frame and 6-frame model also detect the rotation (i.e. clockwise or counterclockwise) and radial (i.e. expansion or contraction) motions of the balloons on the spiral motion feature maps. Some units detect a mixture of rotation and radial motions. The inner angle δ of the peak response is close to the ground truth. Both the 2-frame and 6-frame model detect rotations of the 2 balloons in AP. Some radial motions are also detected by both models in AP. For the 2-frame model, the speed selectivities of the units with peak response are closer to the ground truth, as the 2-Frame-V1 model is more accurate than the 6-Frame-V1 model (See 3.5.2).

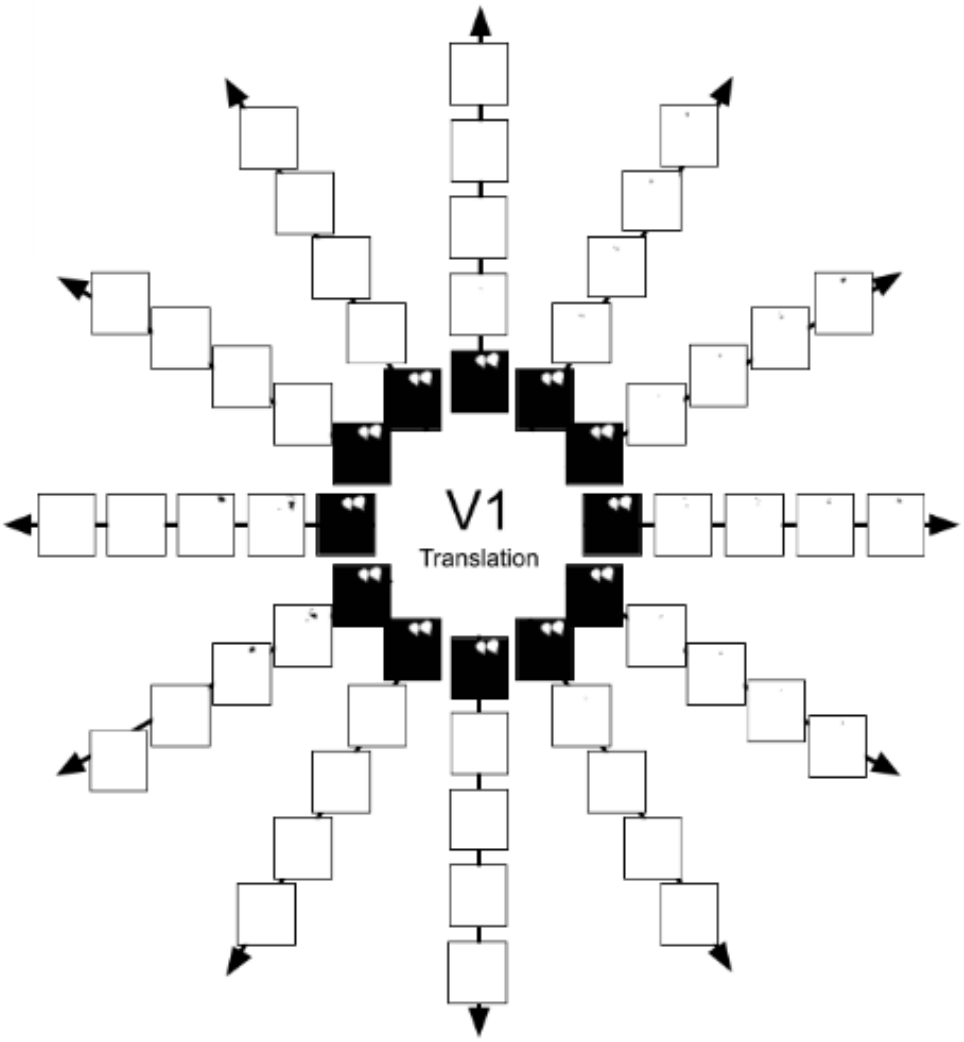

Figure 3.7: Output of area computations of ST-Motion-Net for sequence in Figure 3.6. V1 Translation shows the translation output of Area V1. This shows a close-up view of the GT output for V1 Translation.

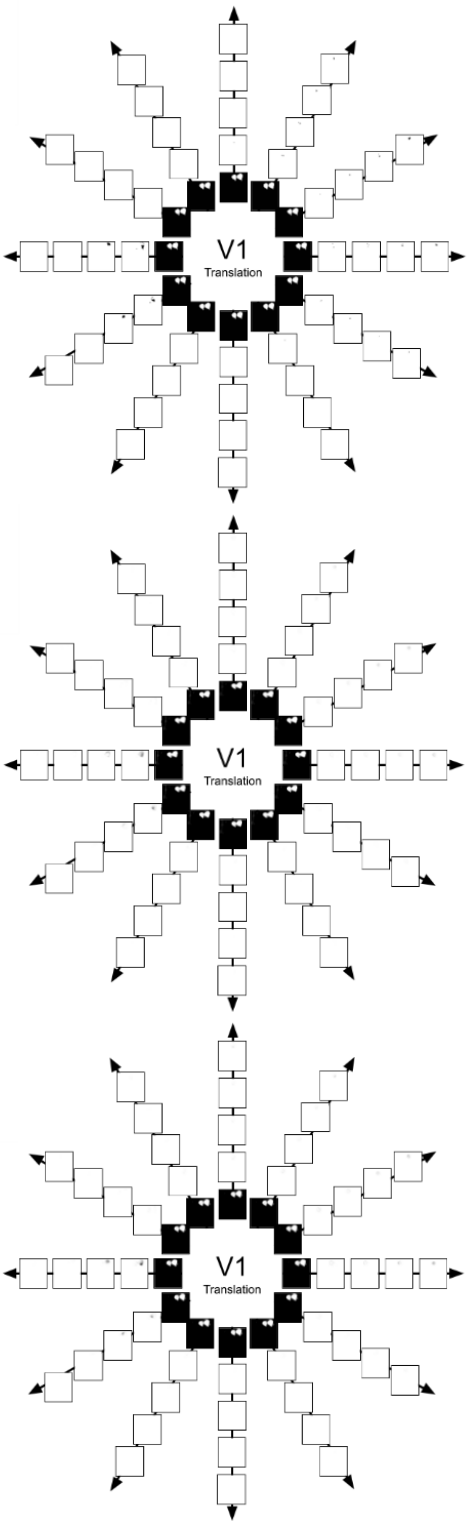

Figure 3.7: Top to bottom: GT output, 2-frame model output, and 6-frame model output. Both the 2-frame and 6-frame model detect translation motions on the 2 balloons.

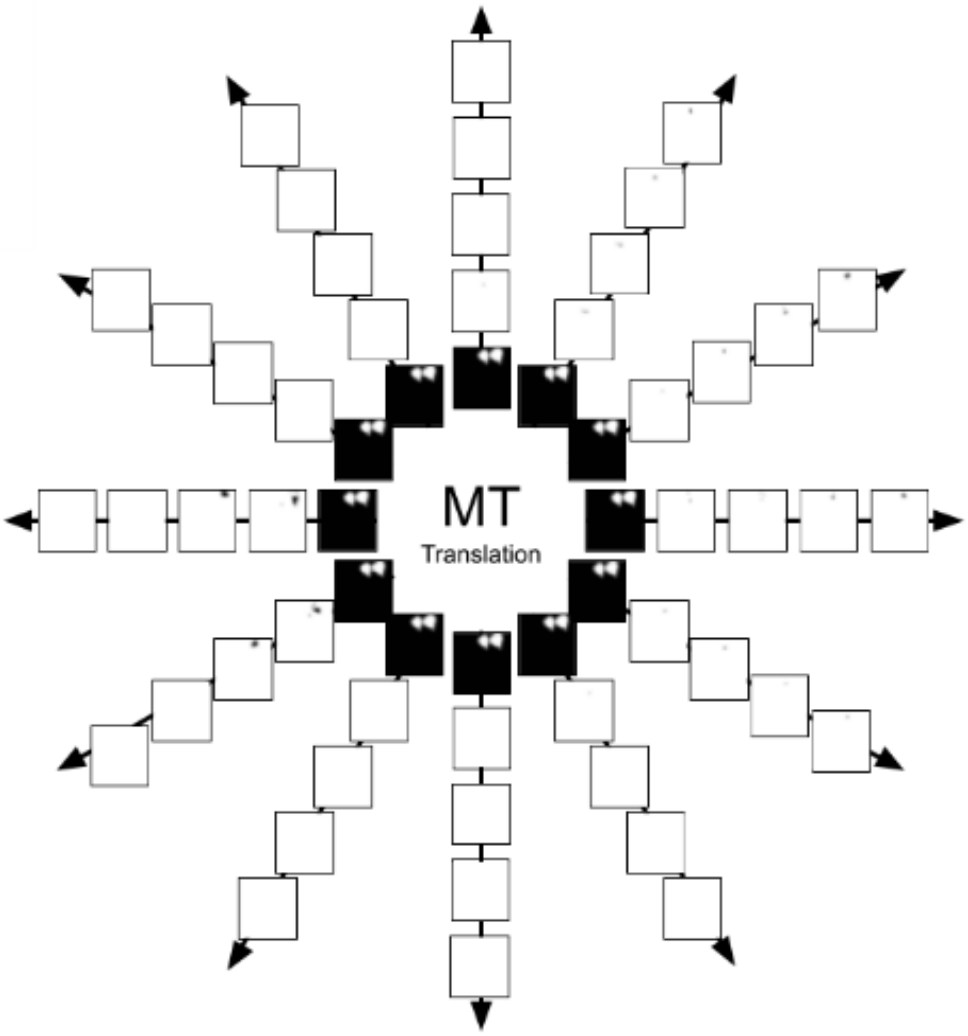

Figure 3.7 (continued): MT Translation shows the translation output of Area MT. This shows a close-up view of the GT output for MT Translation.

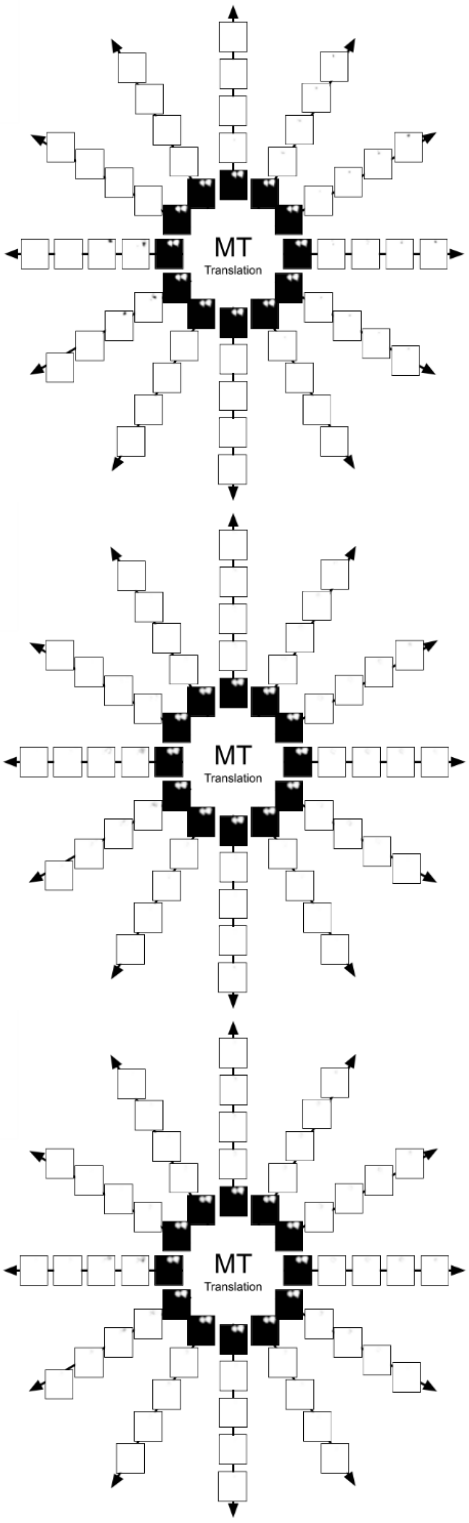

Figure 3.7 (continued): Both the 2-frame and 6-frame model detect translation motions on the 2 balloons.

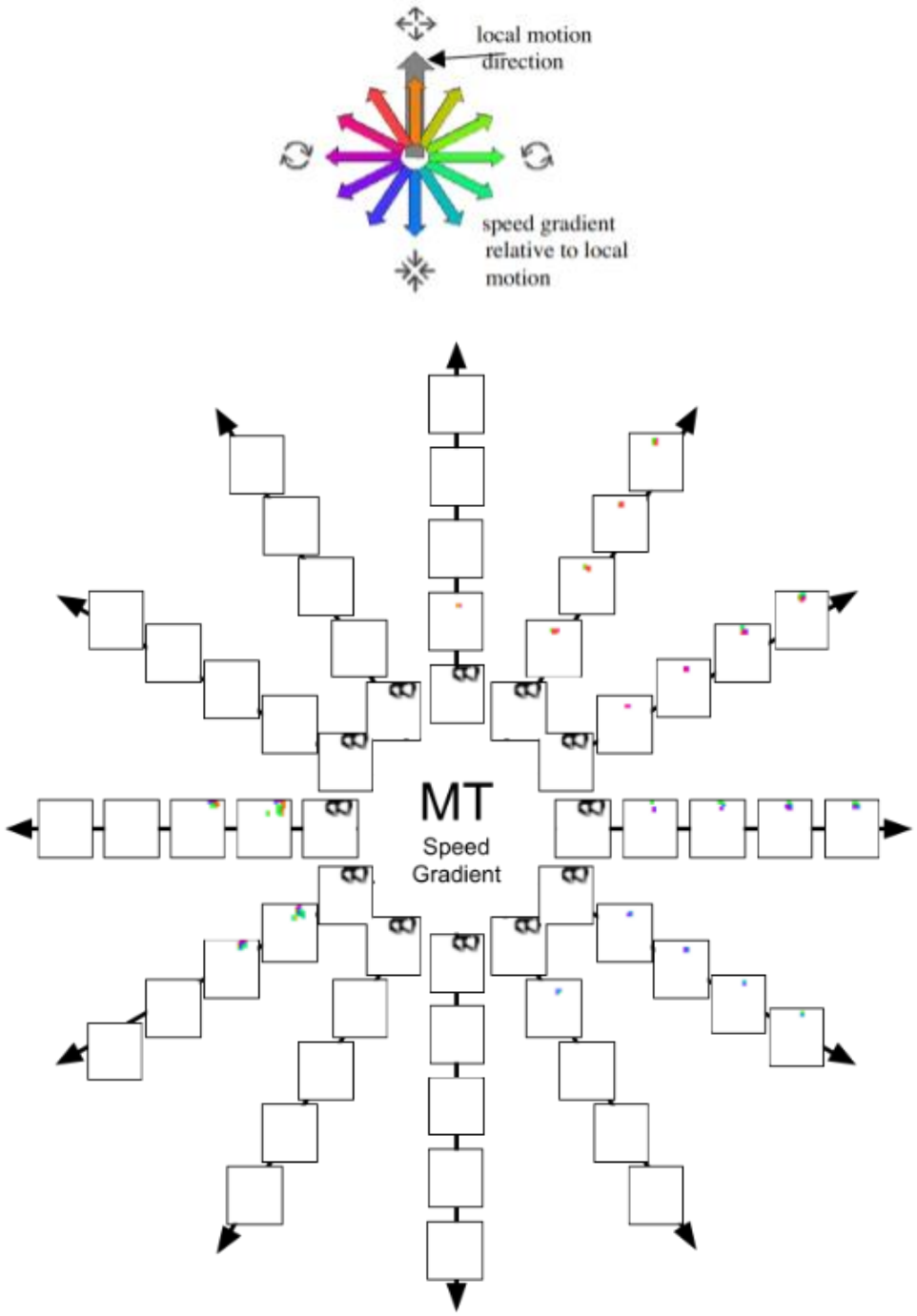

Figure 3.7 (continued): Speed gradient outputs in Area MT are shown in a summary representation of the 12 speed gradients for each local direction and speed. Each coloured dot is the maximum value across the 12 representations. This shows a close-up view of the GT output for MT Speed Gradients in the summary representation.

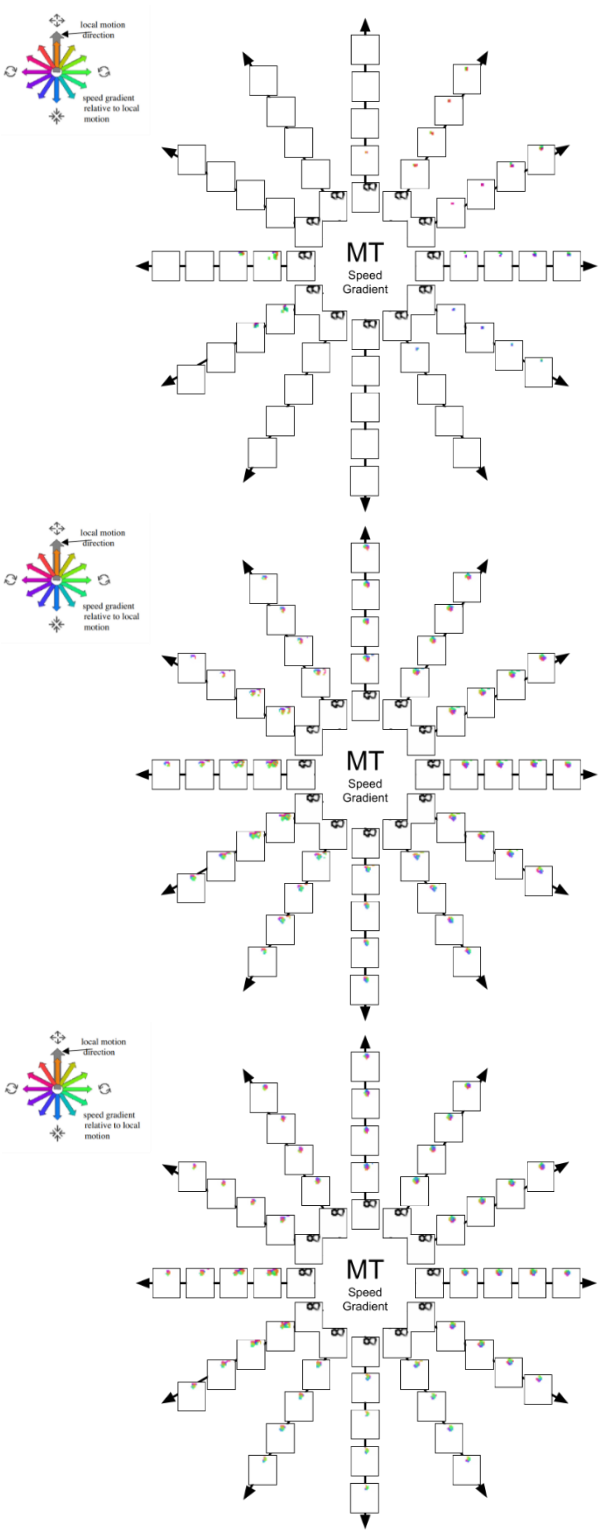

Figure 3.7 (continued): Rotations and radial motions are detected on the balloons (Zoom in).

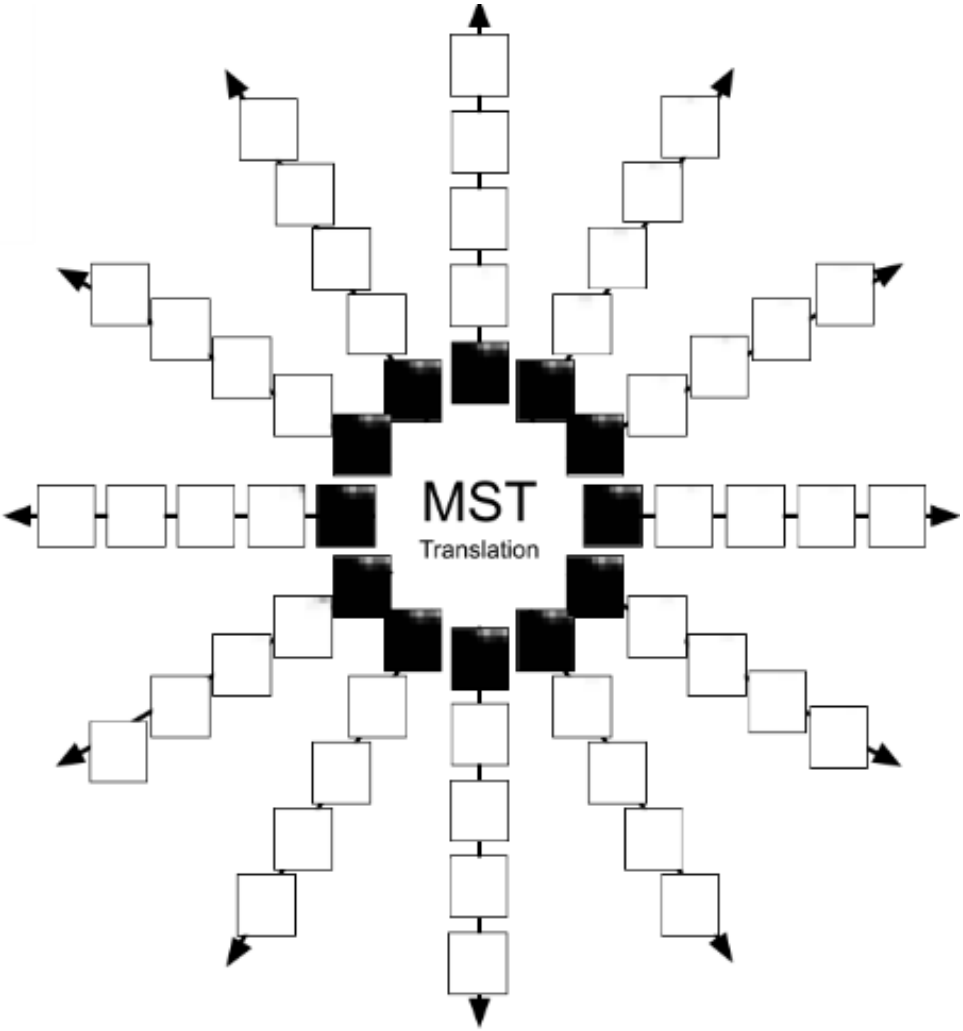

Figure 3.7 (continued): MST Translation shows the translation output in Area MST. This shows a close-up view of the GT output for MST Translation.

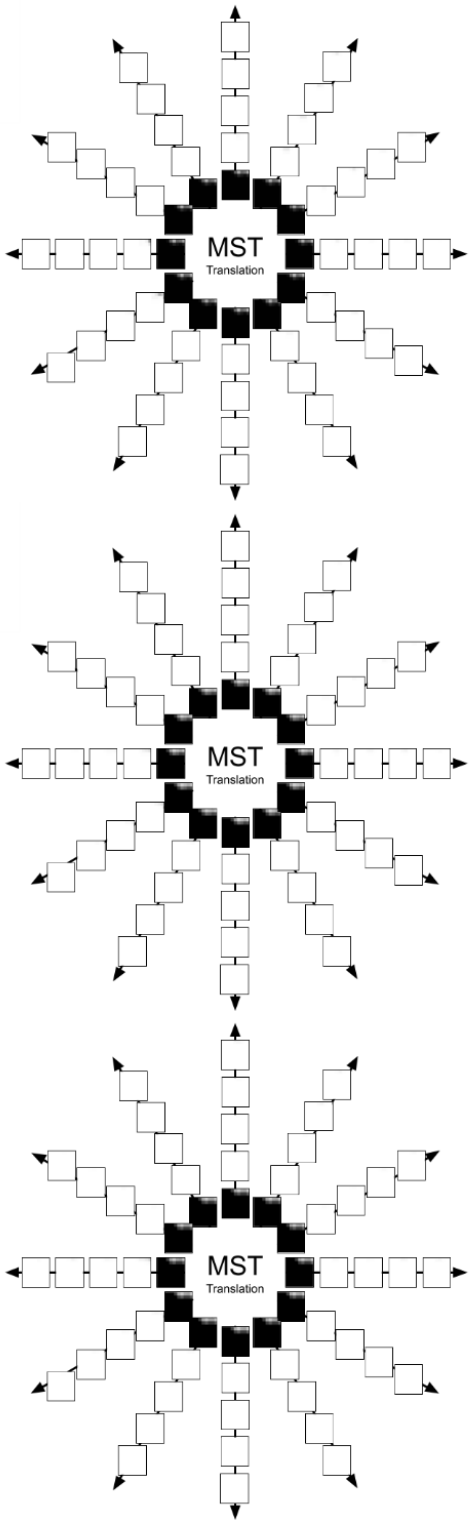

Figure 3.7 (continued): The detections tend to become weaker in higher areas, as the output feature maps become coarser (Zoom in for details).

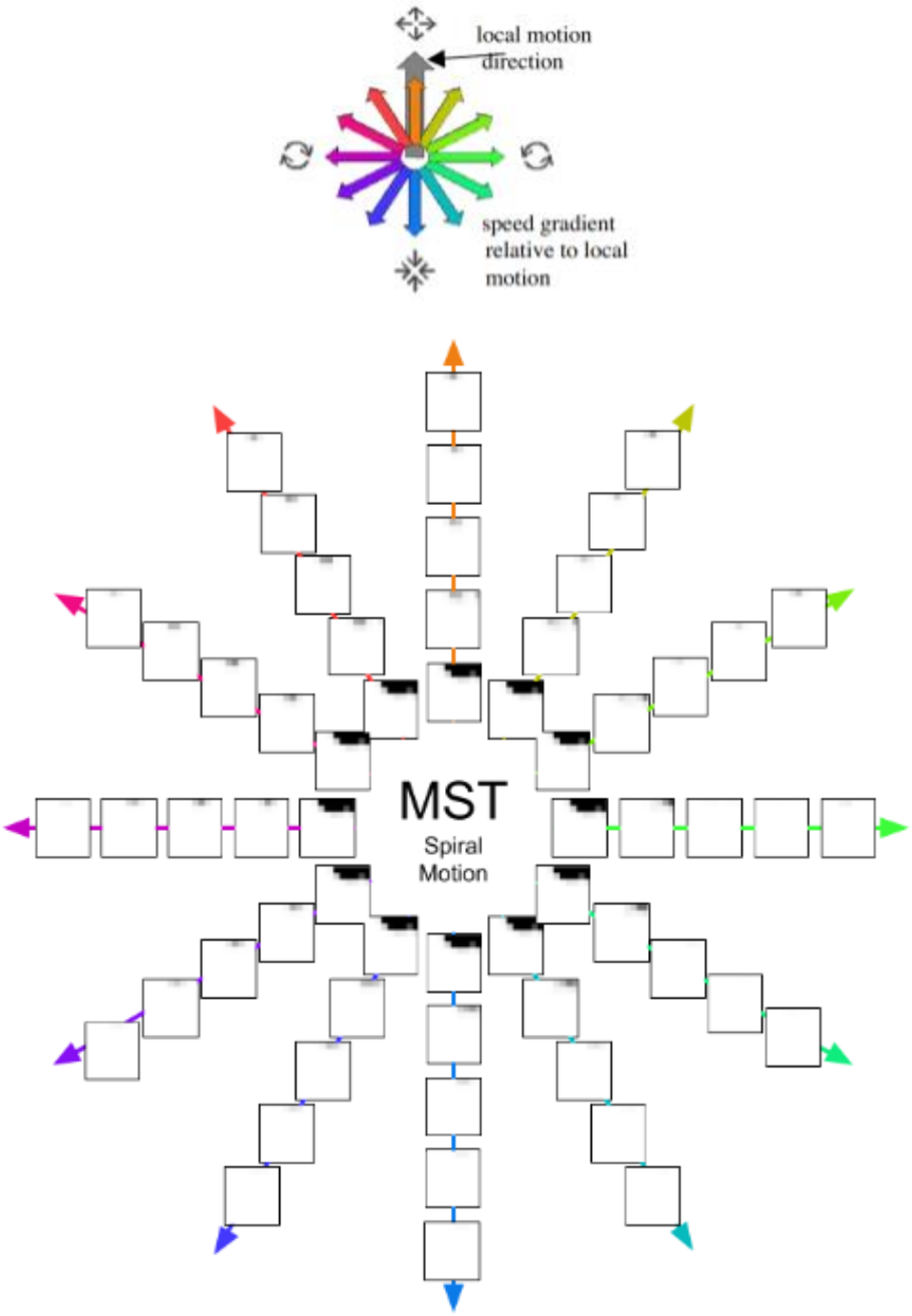

Figure 3.7 (continued): MST Spiral Motion shows the generalized spiral output in Area MST. This shows a close-up view of the GT output for MST Spiral Motion.

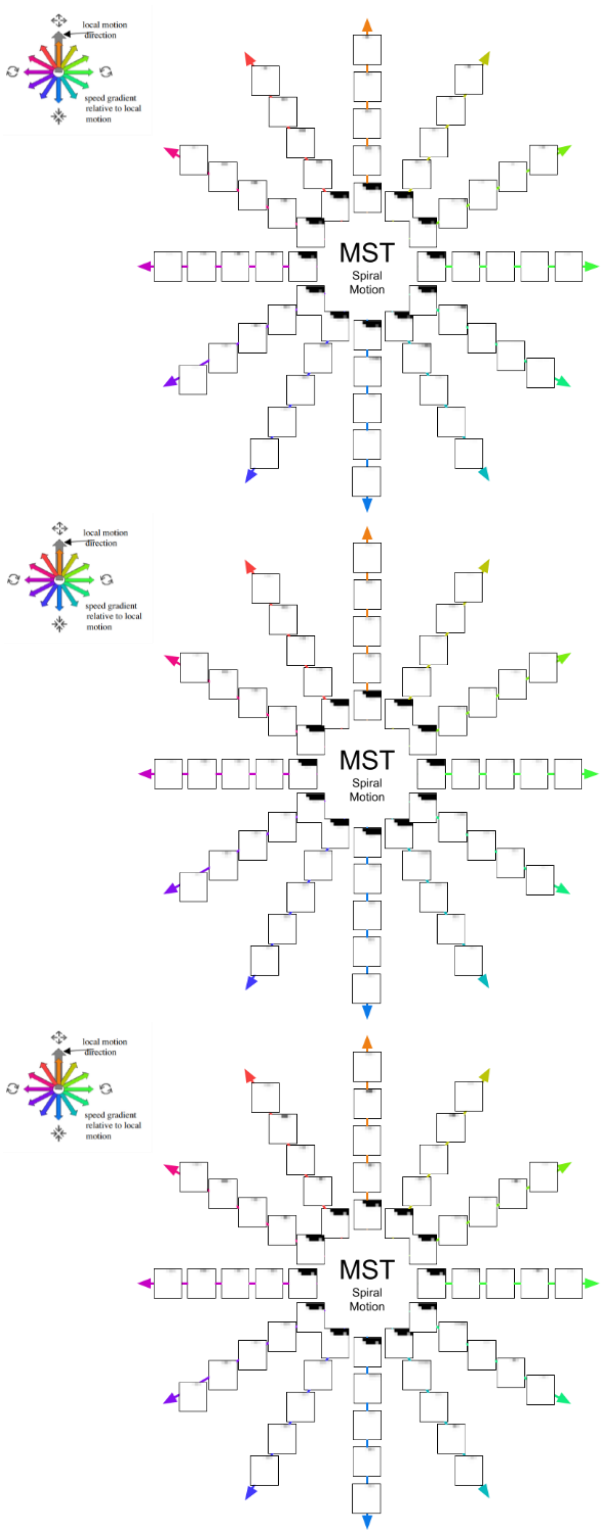

Figure 3.7 (continued): Both the 2-frame and 6-frame model detect the rotation and radial motions of the 2 balloons. Some units detect a mixed motion of the two (See legend). The inner angle δ of the peak response is close to the groundtruth.

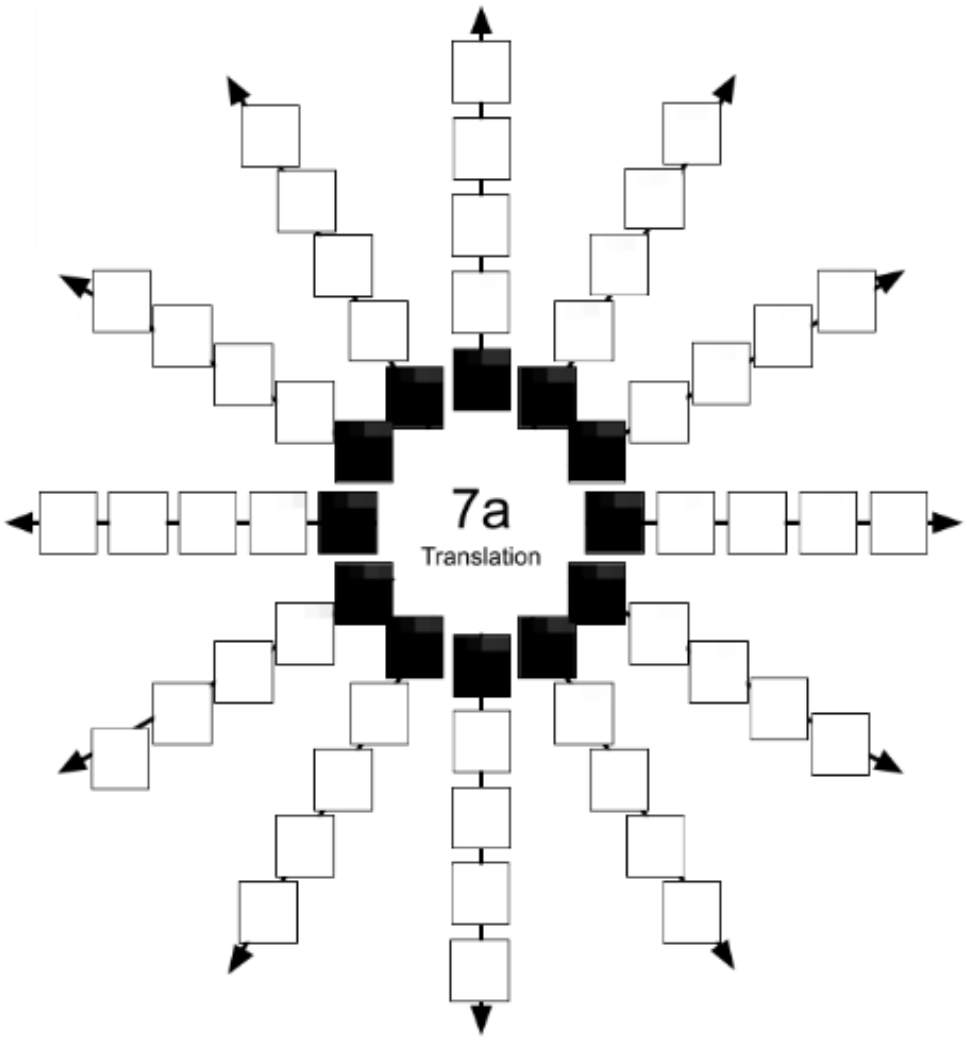

Figure 3.7 (continued): 7a Translation shows the translation output in Area 7a. This shows a close-up view of the GT output for 7a Translation.

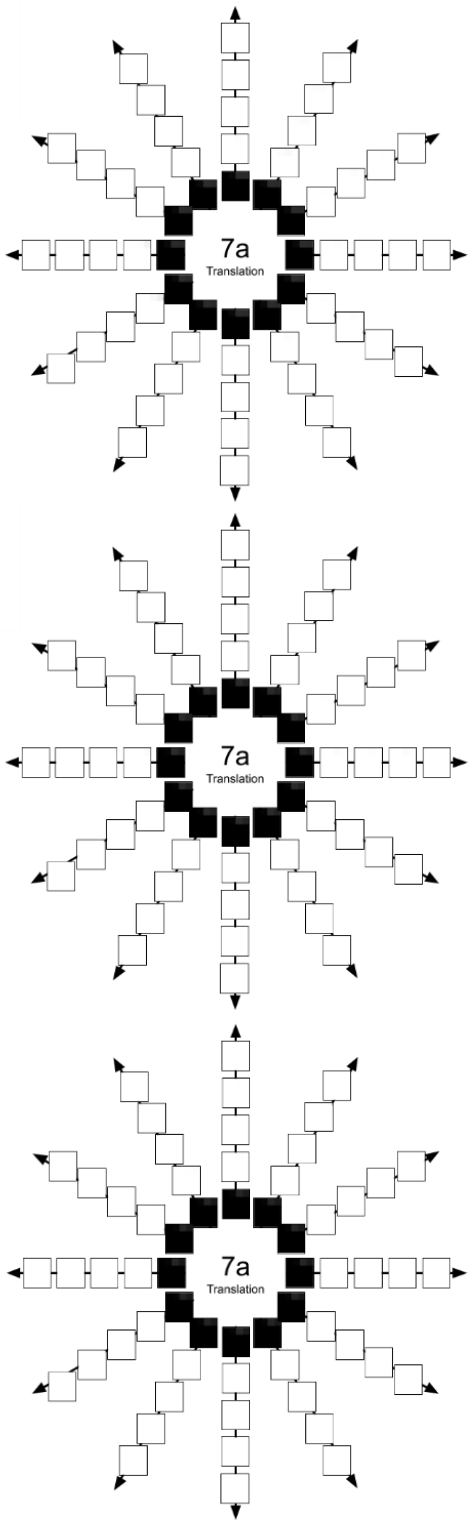

Figure 3.7 (continued): The detections tend to become weaker in higher areas, as the output feature maps become coarser.

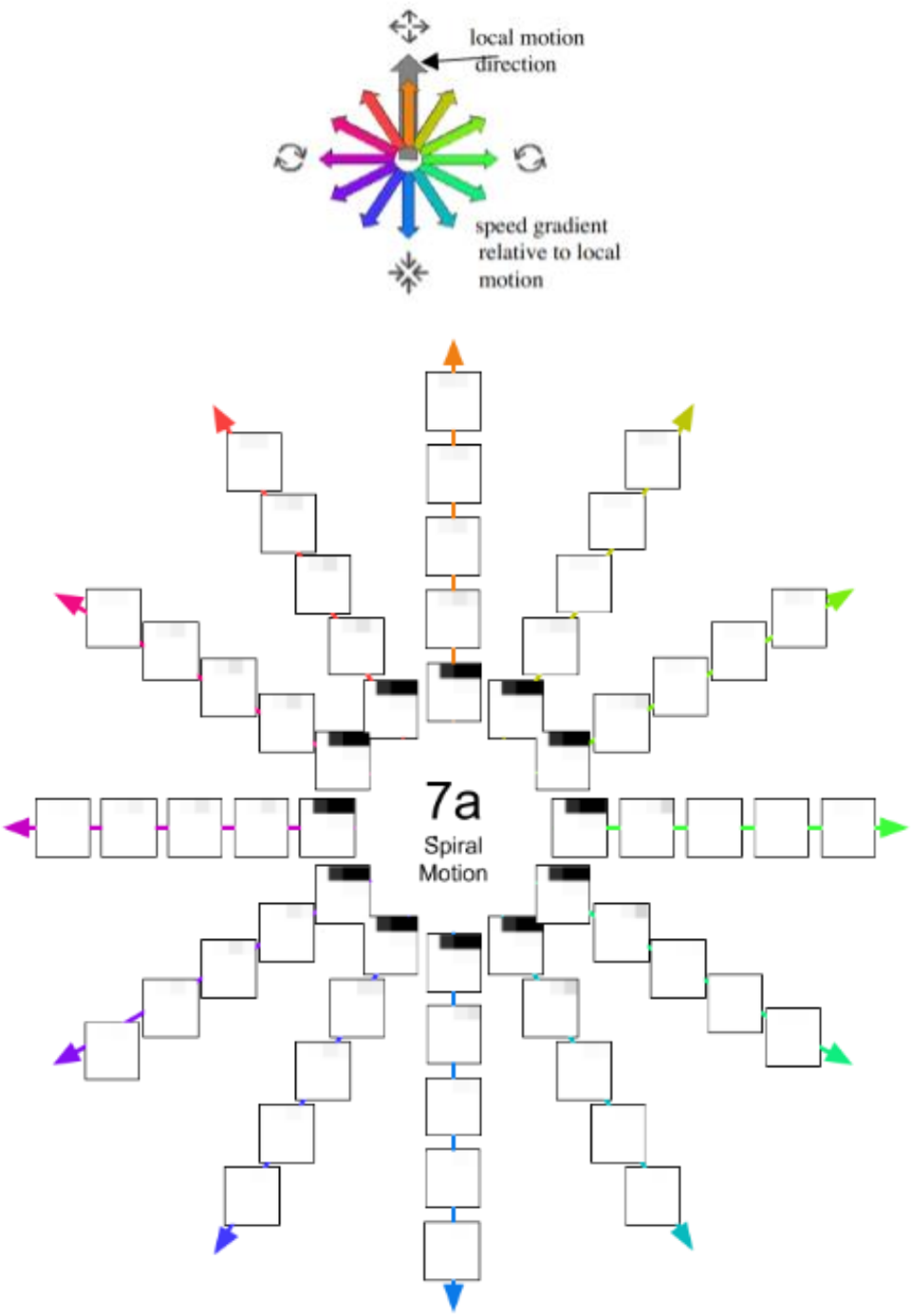

Figure 3.7 (continued): 7a Spiral Motion shows the generalized spiral output in Area 7a. This shows a close-up view of the GT output for 7a Spiral Motion.

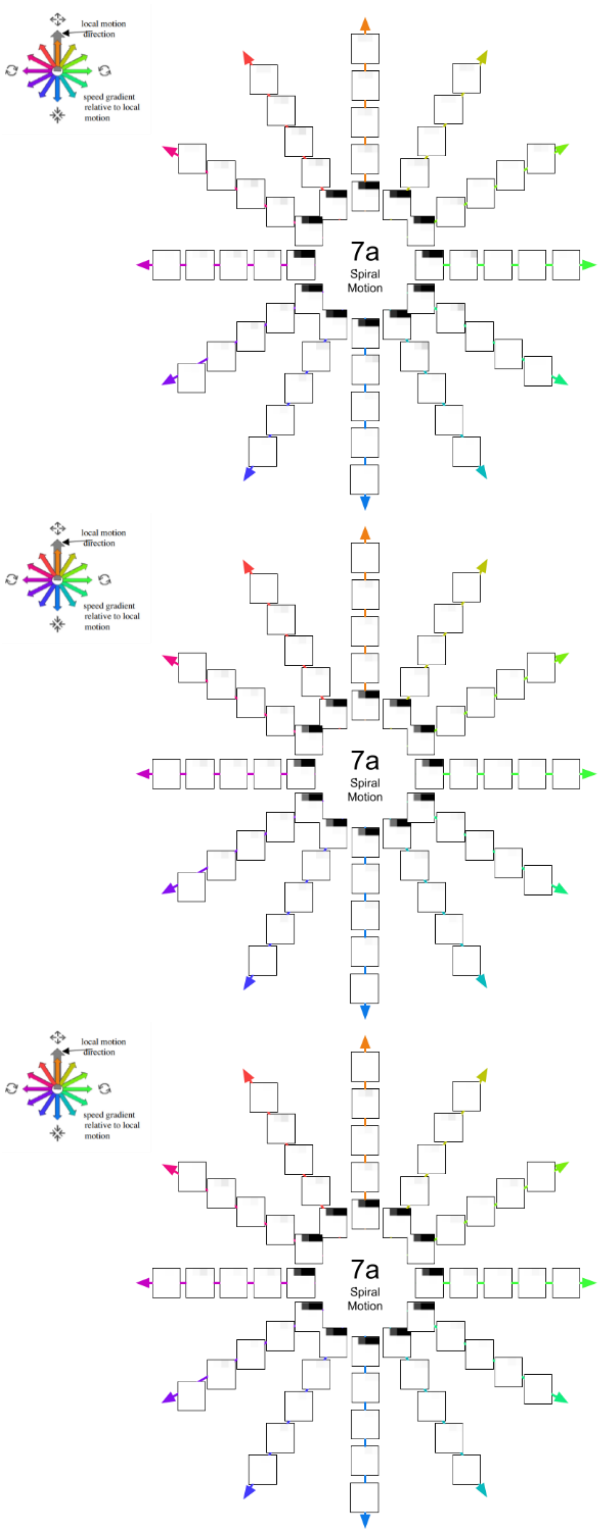

Figure 3.7 (continued): Both the 2-Frame and 6-Frame model detect the rotation and radial motions of the 2 balloons. Some units detect a mixed motion of the two (See legend). The inner angle δ of the peak response is close to the groundtruth.

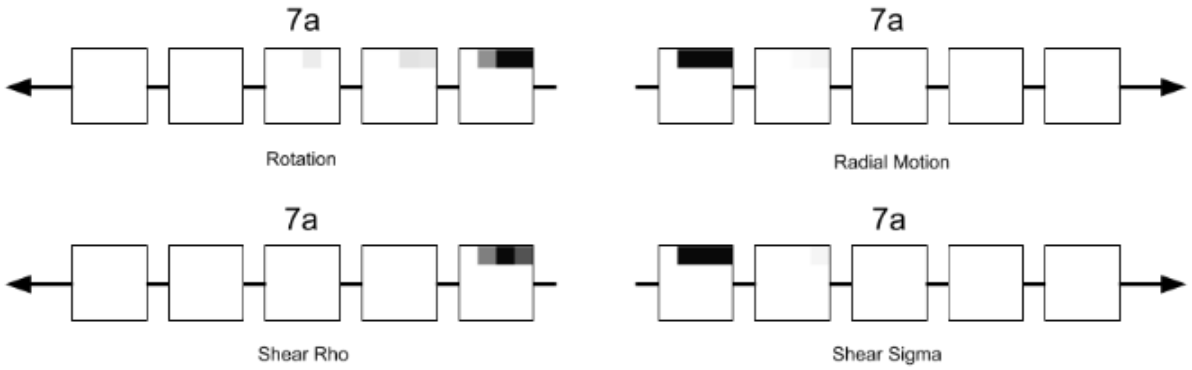

Figure 3.7 (continued): Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the GT outputs for AP.

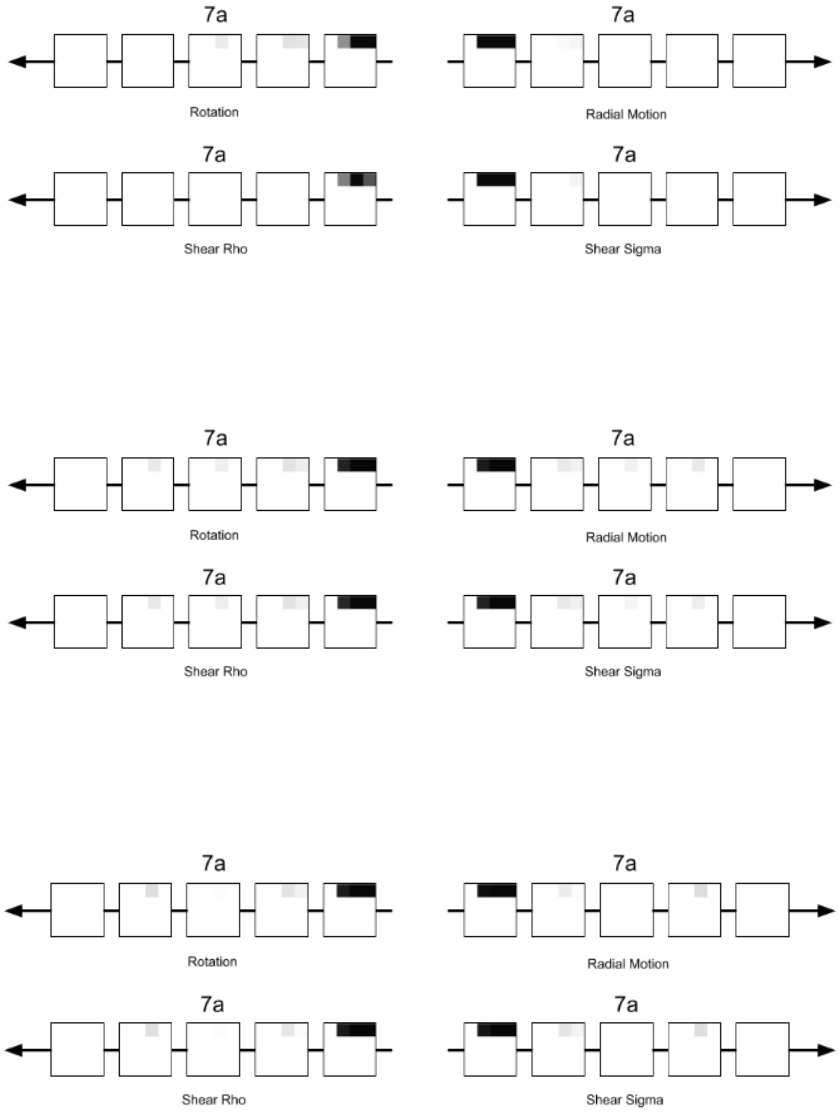

Figure 3.7 (continued): Both the 2-frame and 6-frame model detect rotations of the 2 balloons in AP. Some radial motions are also detected by both models in AP. For the 2-Frame model, the speed selectivities of the units with peak response are closer to the groundtruth.

Figure 3.8 shows a sequence from Blender-Complex Dataset containing a clockwise rotating soccer ball and a contracting volleyball on a still background. We compare the GT outputs, 2-frame model outputs, and 6-frame model outputs from AP in Figure 3.9. Close-up views of the 2-frame and 6-frame model outputs are available in Figure A.3.

Figure 3.8: Sequence from Blender-Complex of a clockwise rotating soccer ball and a contracting volleyball. Inputs to 2-Frame Area V1 are the 2 center frames and inputs to 6-Frame Area V1 are the 6 frames.

Both the 2-frame and 6-frame model detect translation motions on the 2 balls. Some detections around the occlusion boundary of the 2 balls are weaker as some moving pixels of the contracting volleyball do not have a matching pixel on the other frames. For spiral motion patterns, both the 2-frame and 6-frame model detect the clockwise rotation (around $\delta = 90°$) and contraction (around $\delta = 180°$) of the 2 balls. The inner angle δ of the peak response is close to the GT direction. For the 2-frame model, the speed selectivity of the peak response is closer to the GT. Both the 2-frame and 6-frame model detect radial motion of the volleyball in AP. The speed selectivity of the peak response from the 2-frame model is closer to the ground truth, as the 2-Frame-V1 model is more accurate than the 6-Frame-V1 model (See 3.5.2). Both models also detect Rotation of the soccer ball among the top detected motion patterns.

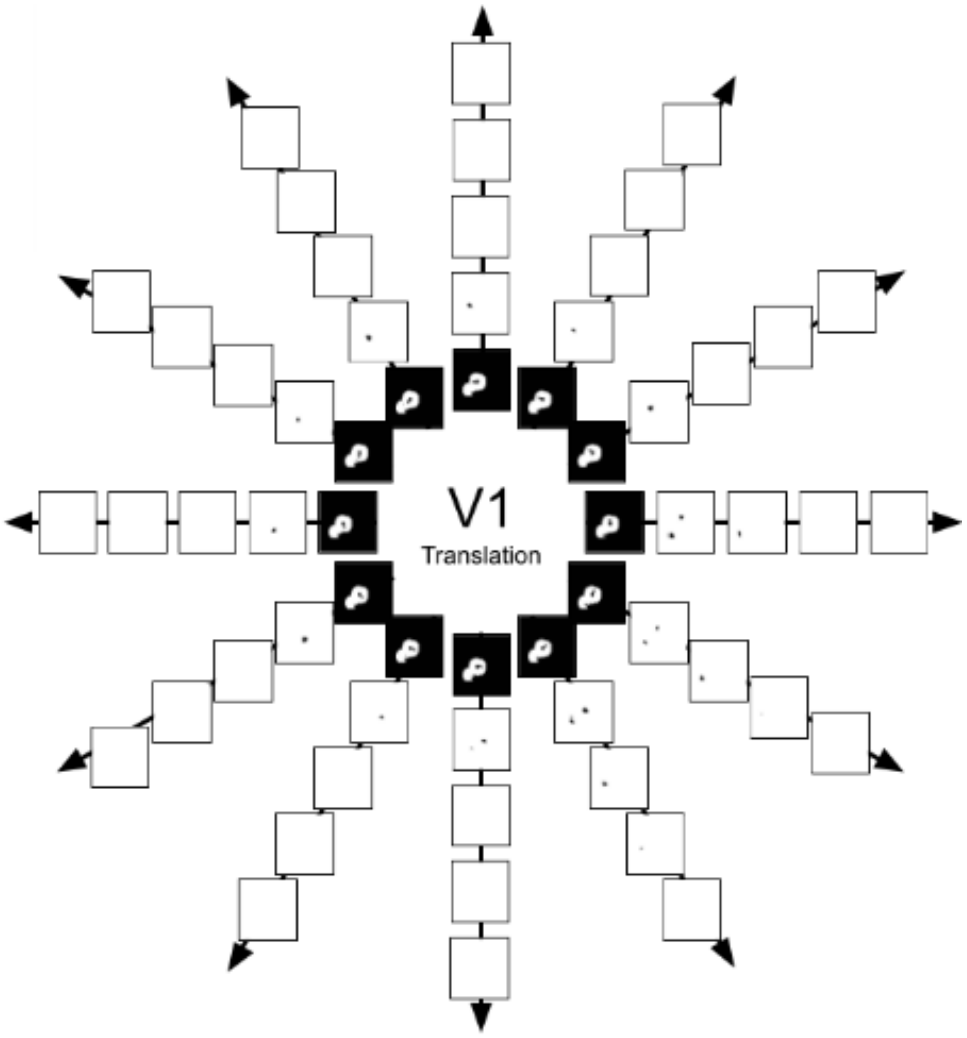

Figure 3.9: Output of area computations of ST-Motion-Net for sequence in Figure 3.8. V1 Translation shows the translation output of Area V1. This shows a close-up view of the GT output for V1 Translation.

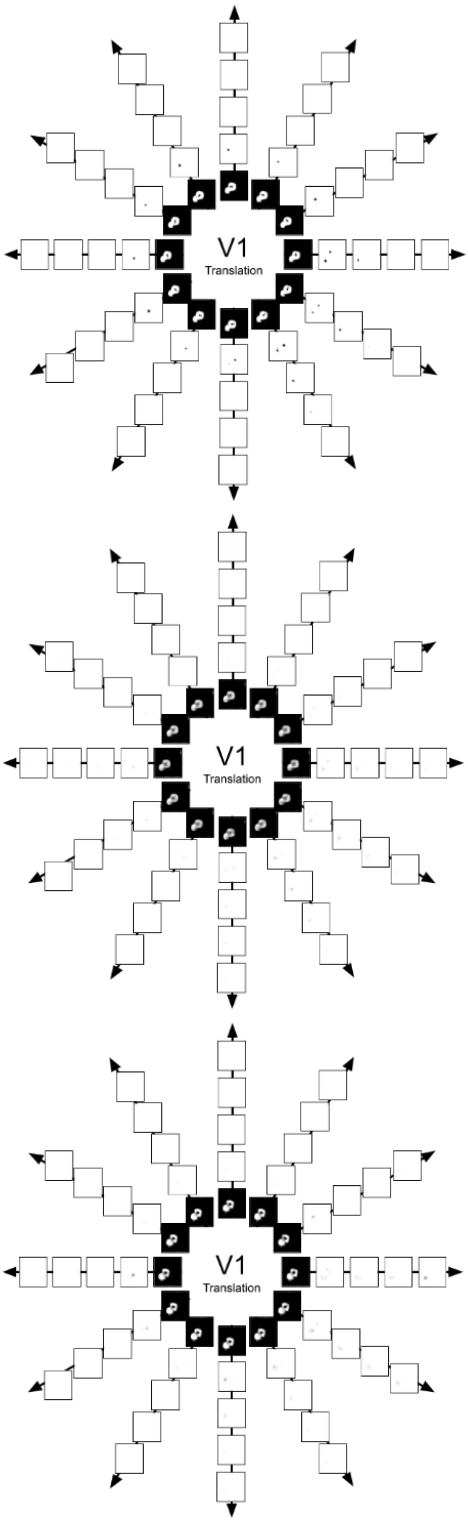

Figure 3.9: Top to bottom: GT output, 2-frame model output, and 6-frame model output. Both the 2-frame and 6-frame model detect translation motion patterns on the 2 balls. Some detections around the occlusion boundary of the 2 balls are weaker.

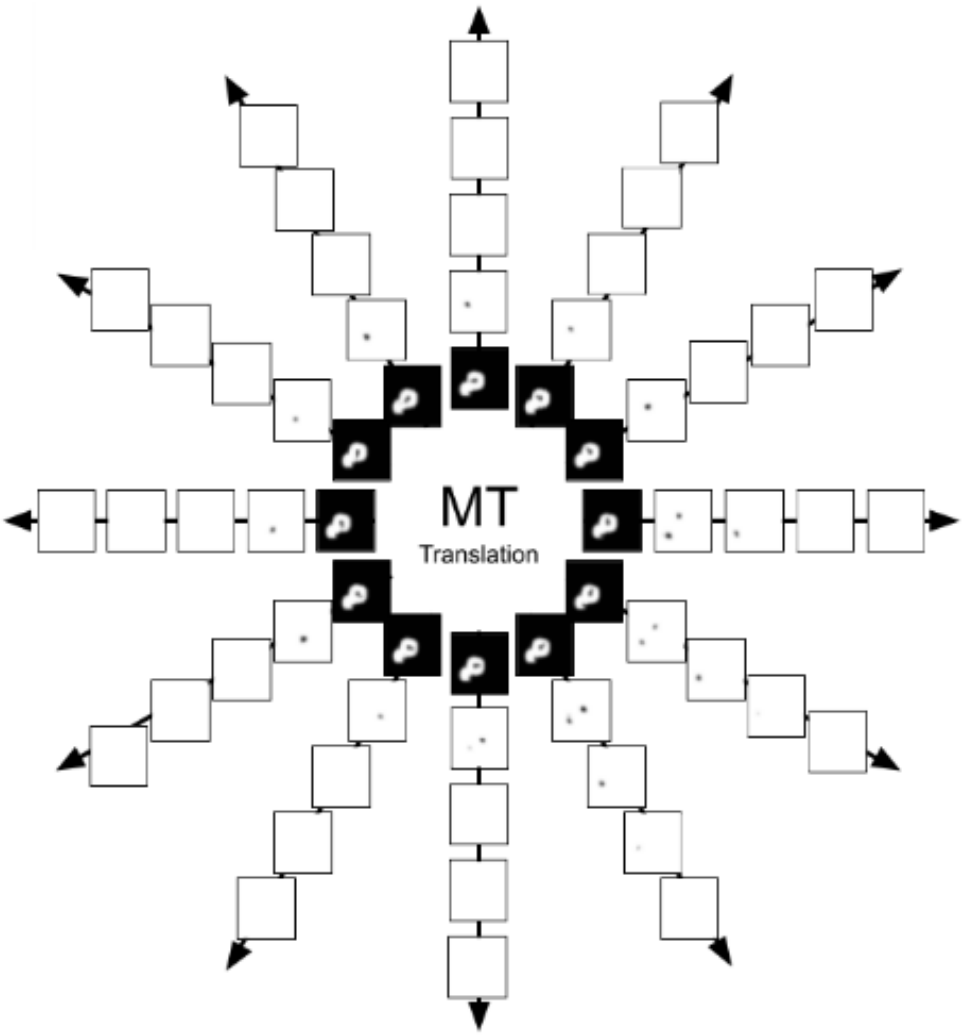

Figure 3.9 (continued): MT Translation shows the translation output of Area MT. This shows a close-up view of the GT output for MT Translation.

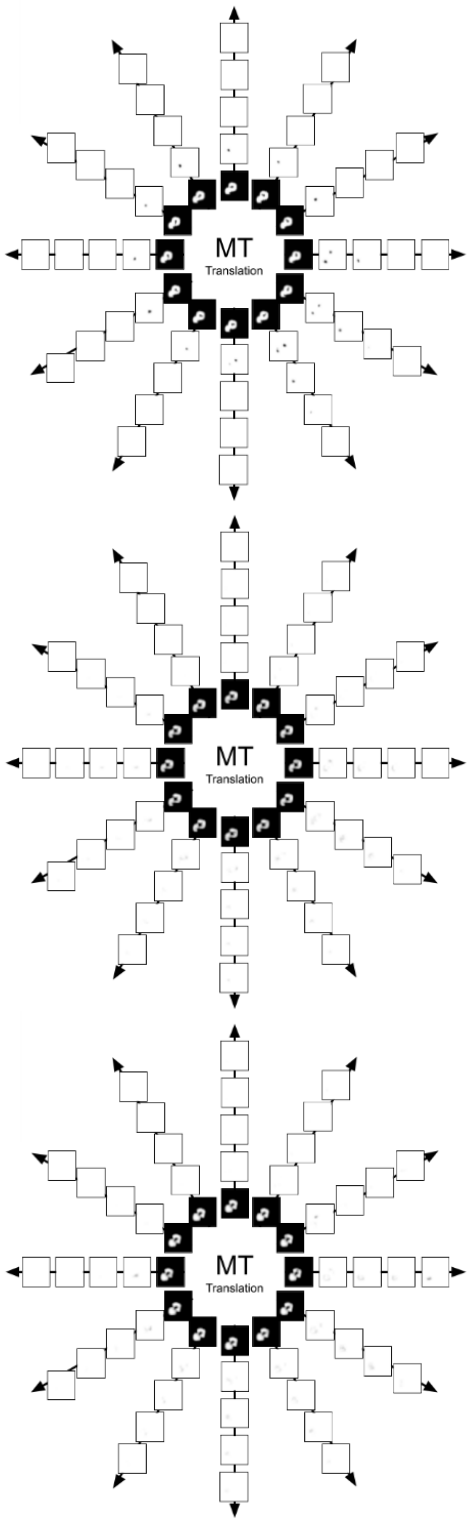

Figure 3.9 (continued)

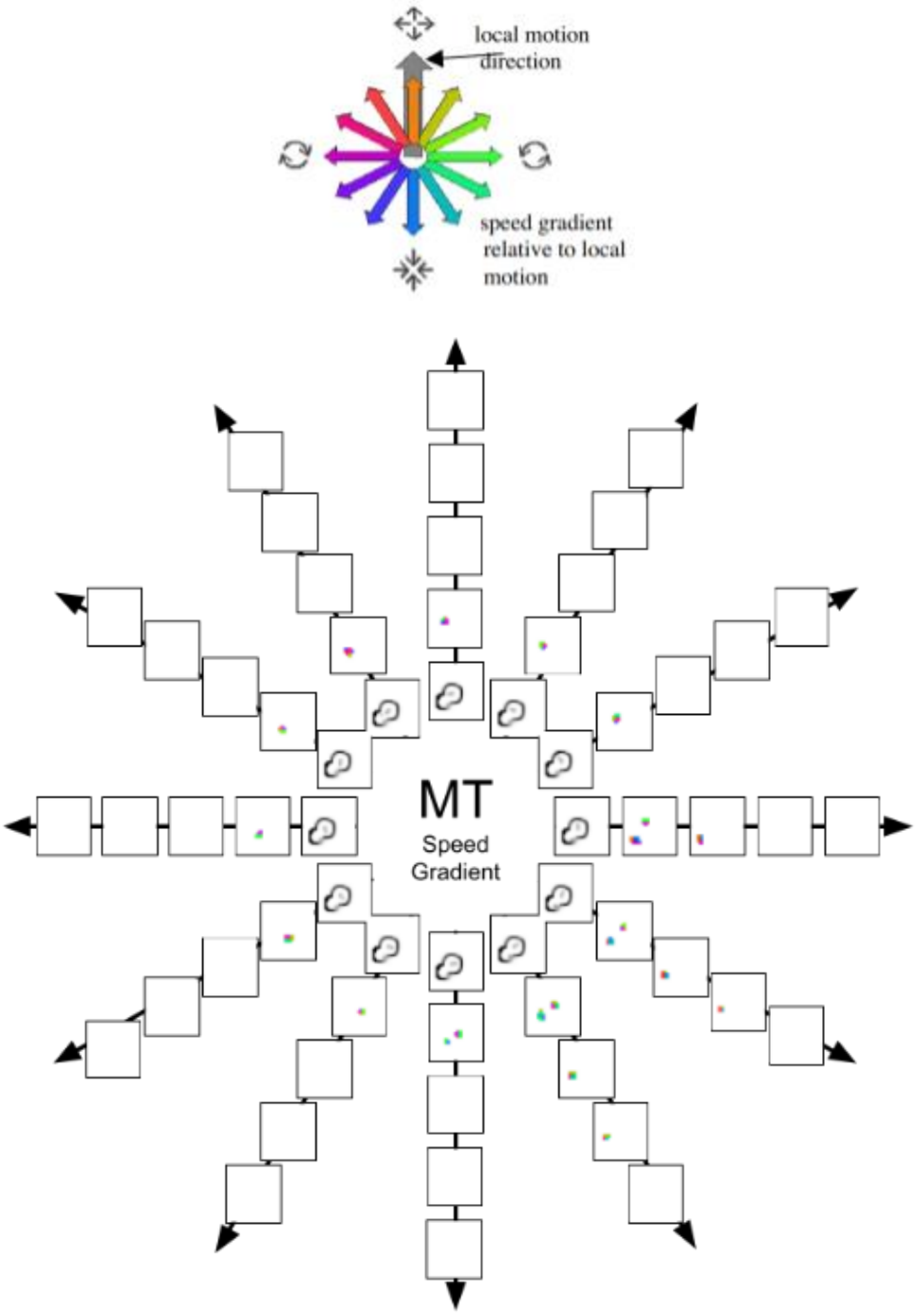

Figure 3.9 (continued): Speed gradient outputs in Area MT are shown in a summary representation of the 12 speed gradients for each local direction and speed. Each coloured dot is the maximum value across the 12 representations. This shows a close-up view of the GT output for MT Speed Gradients in the summary representation.

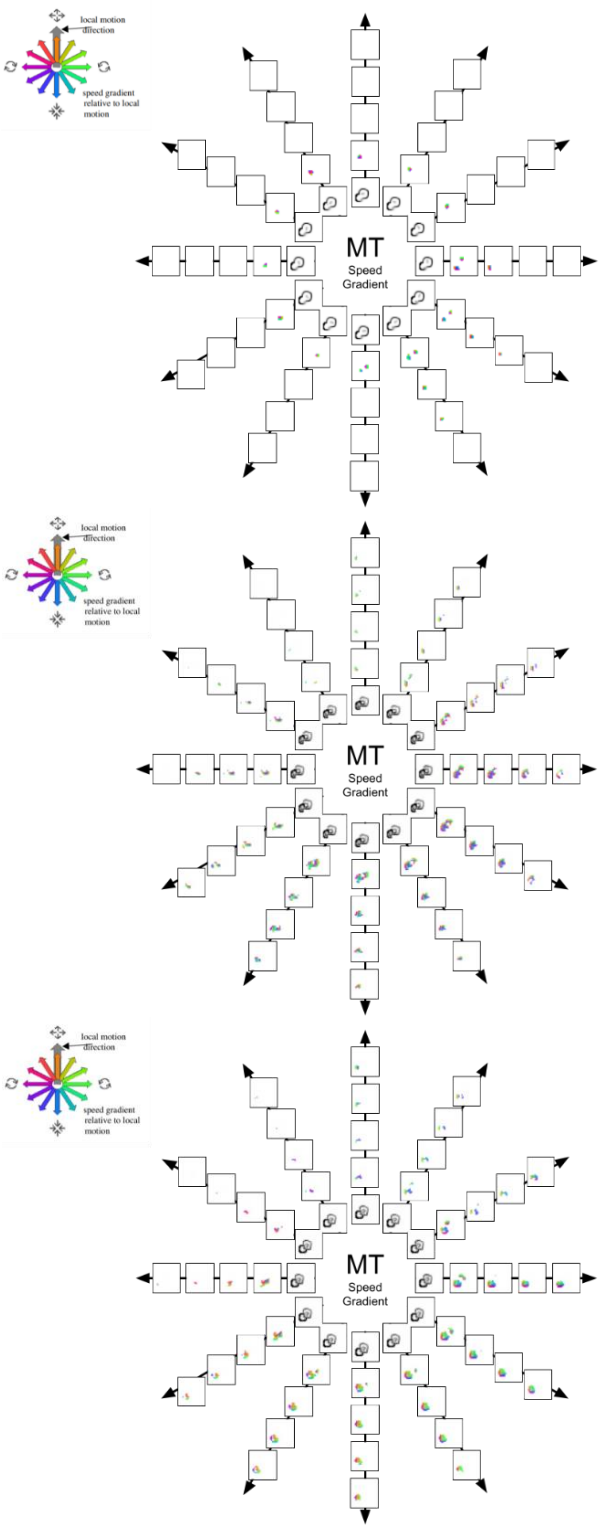

Figure 3.9 (continued): Contraction or clockwise rotation (roughly blue / purple) are detected by both the 2-Frame and 6-Frame models.

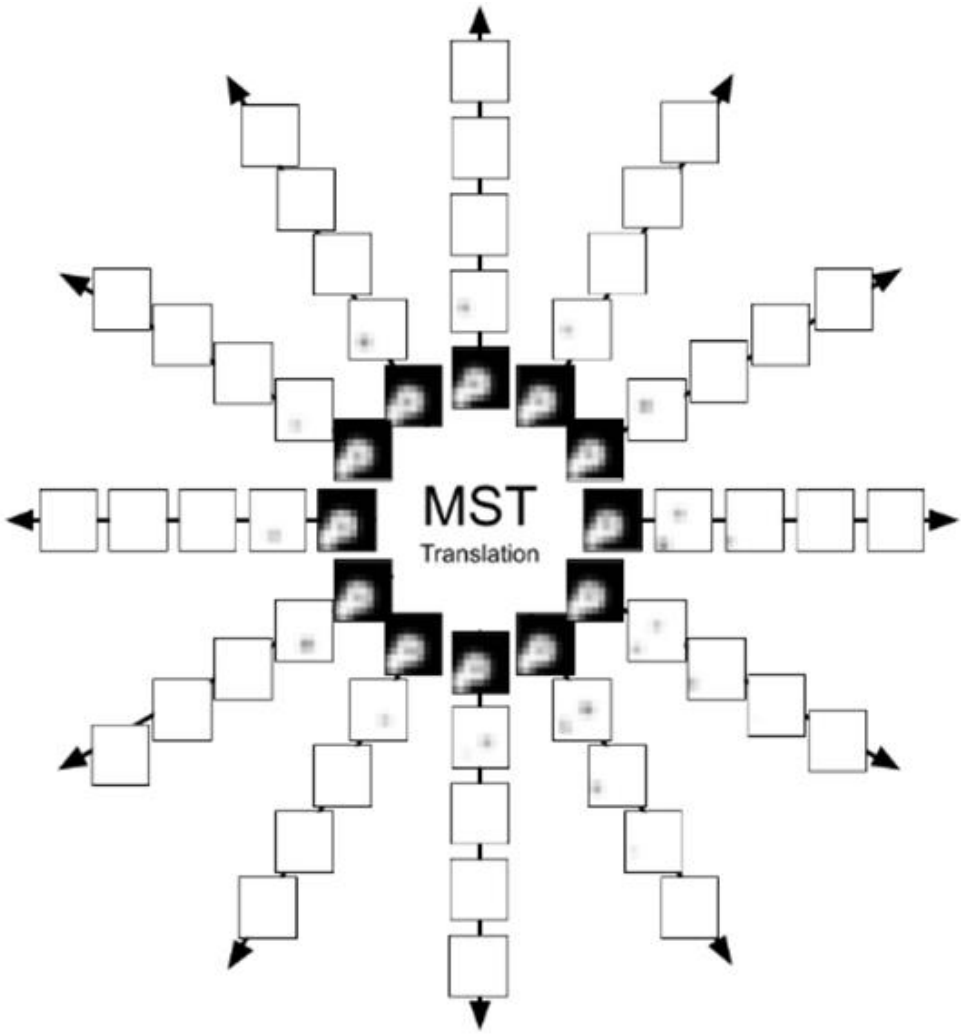

Figure 3.9 (continued): MST Translation shows the translation output in Area MST. This shows a close-up view of the GT output for MST Translation.

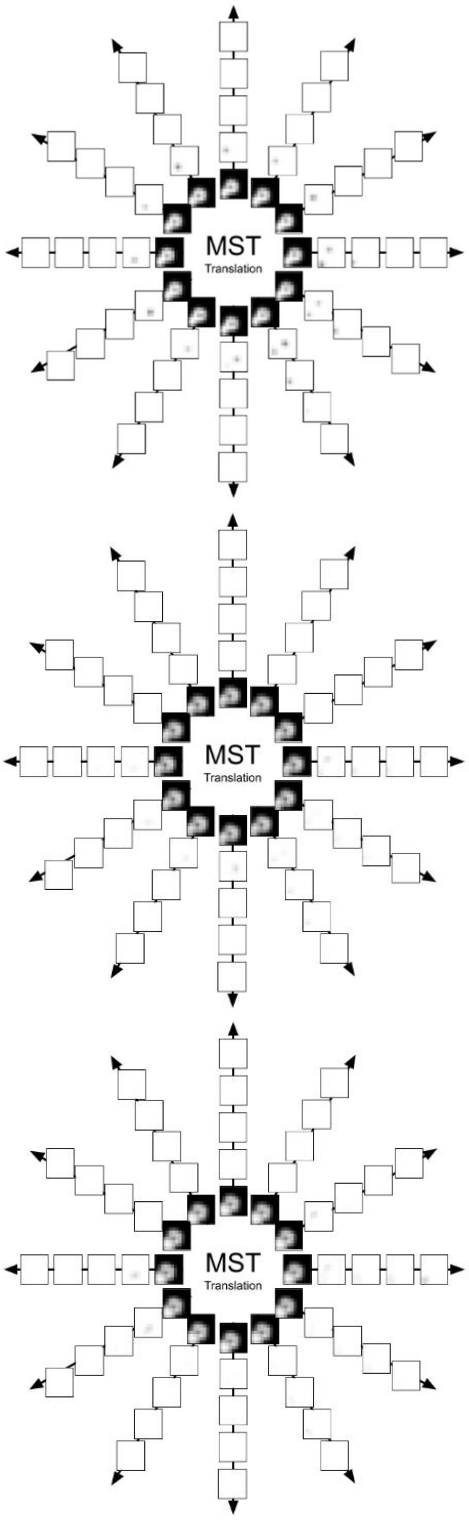

Figure 3.9 (continued)

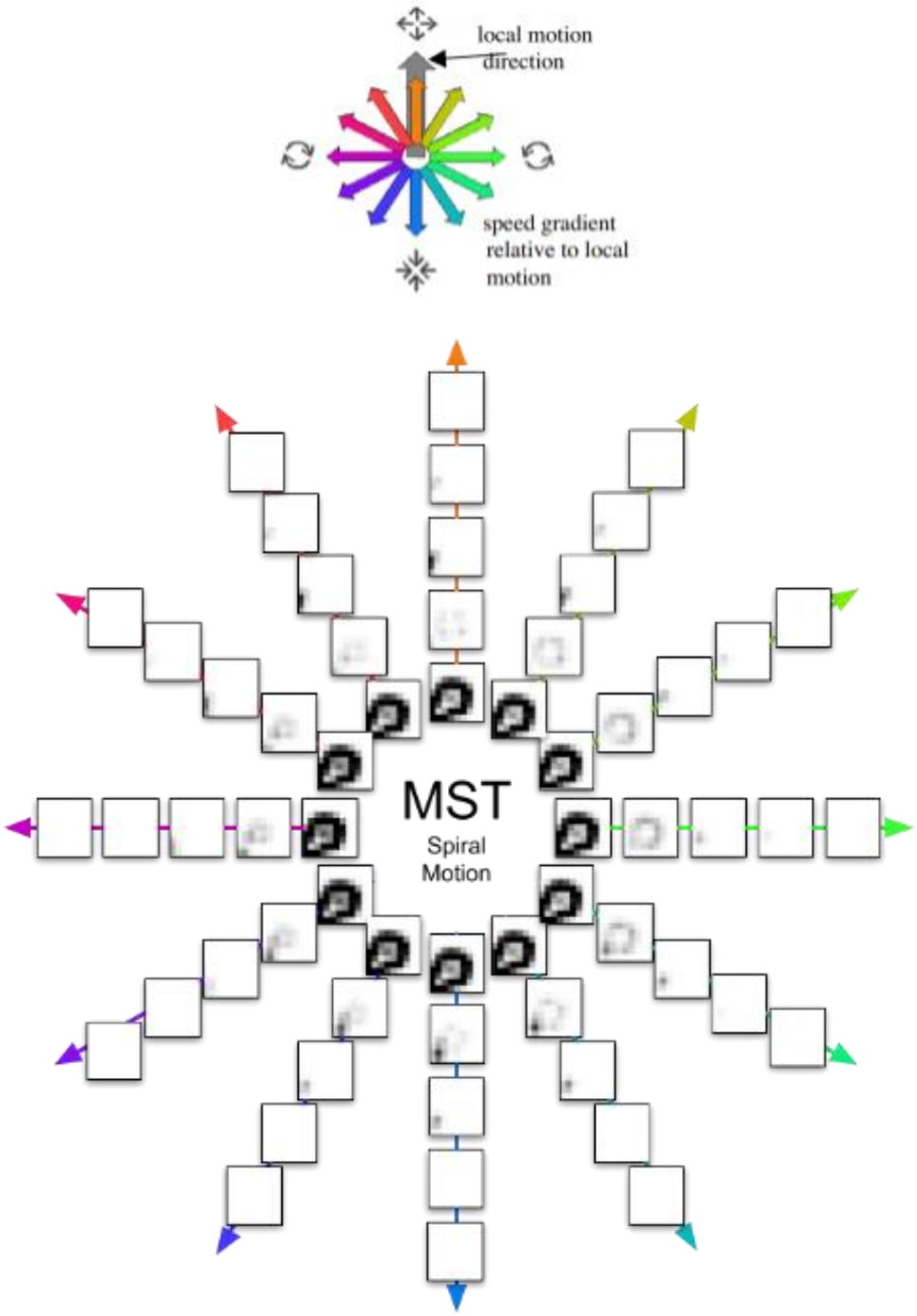

Figure 3.9 (continued): MST Spiral Motion shows the generalized spiral output in Area MST. This shows a close-up view of the GT output for MST Spiral Motion.

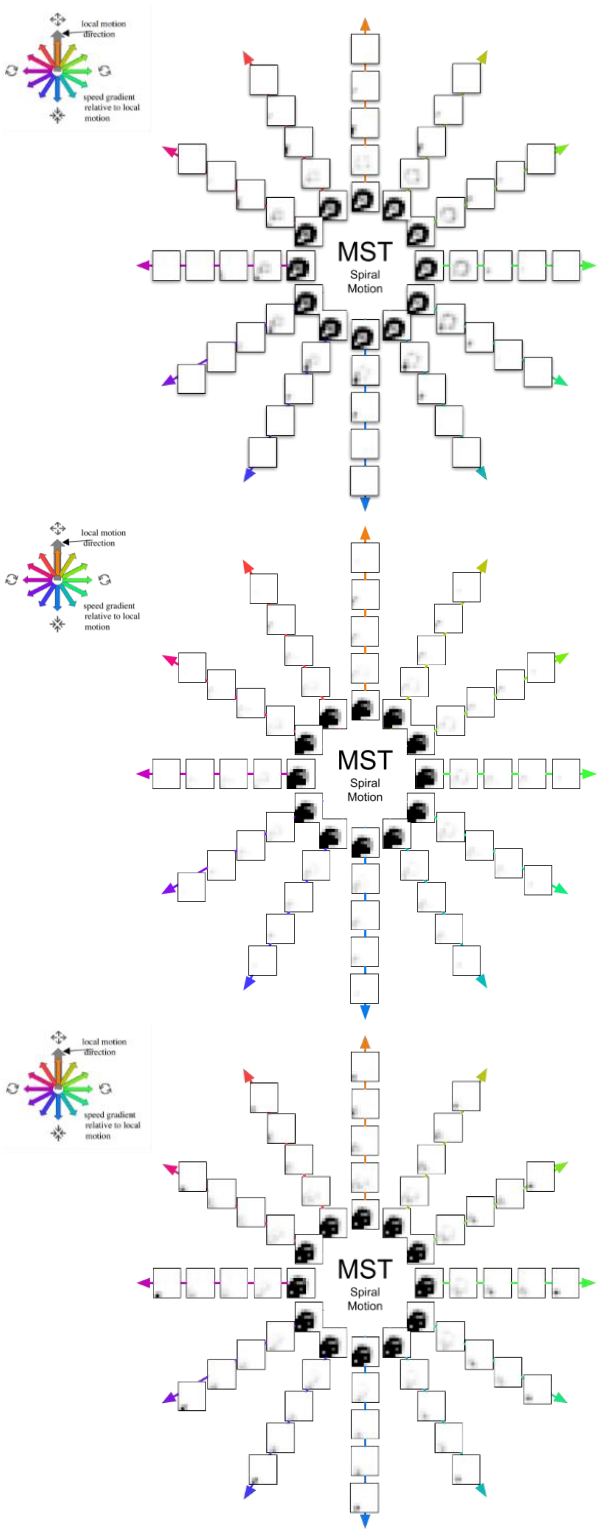

Figure 3.9 (continued): Both the 2-Frame and 6-Frame model detect the clockwise rotation (around δ=90˚) of the soccer ball and the contraction (around δ=180˚) of the volleyball. Inner angle δ of the peak response is close to groundtruth.

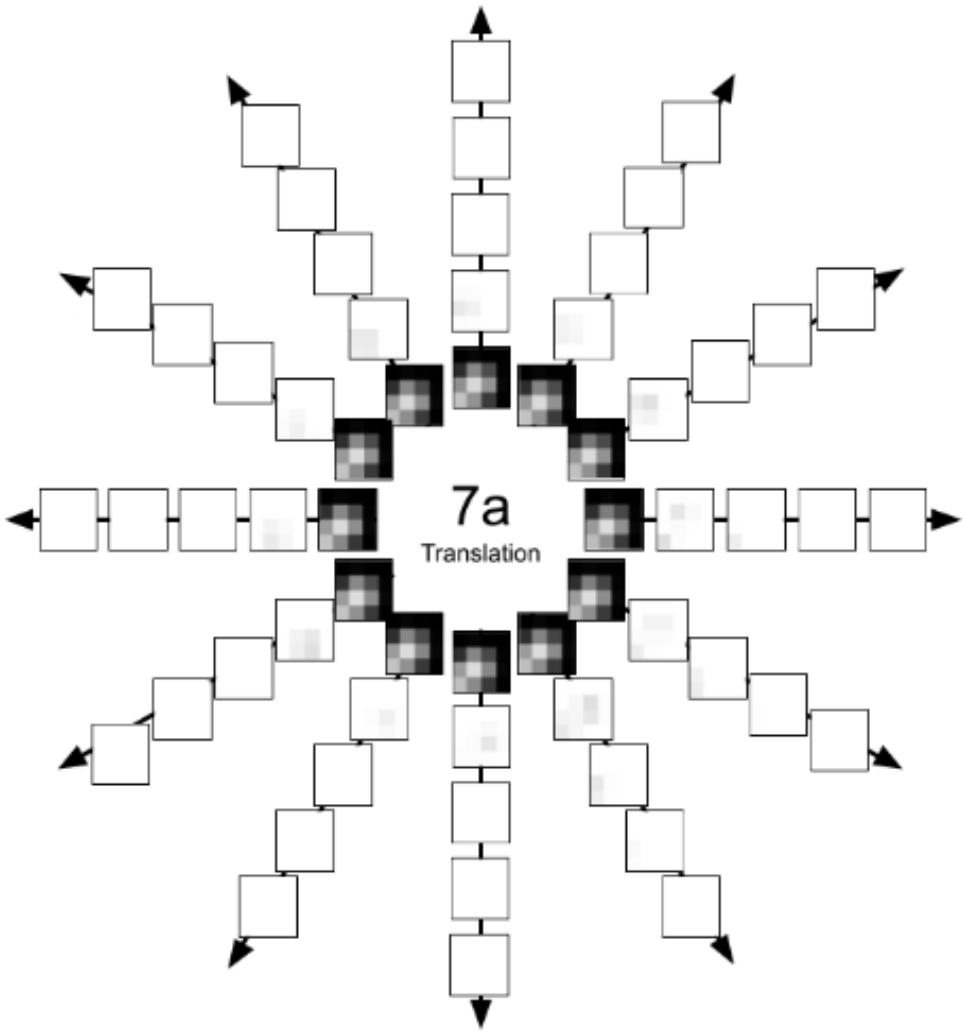

Figure 3.9 (continued): 7a Translation shows the translation output in Area 7a. This shows a close-up view of the GT output for 7a Translation.

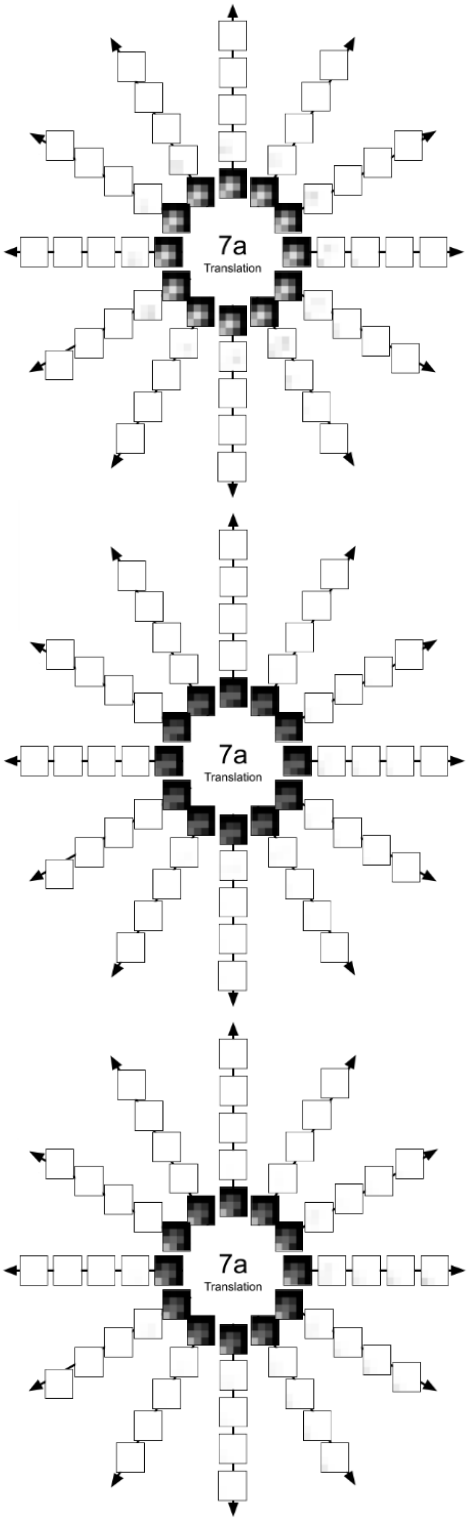

Figure 3.9 (continued)

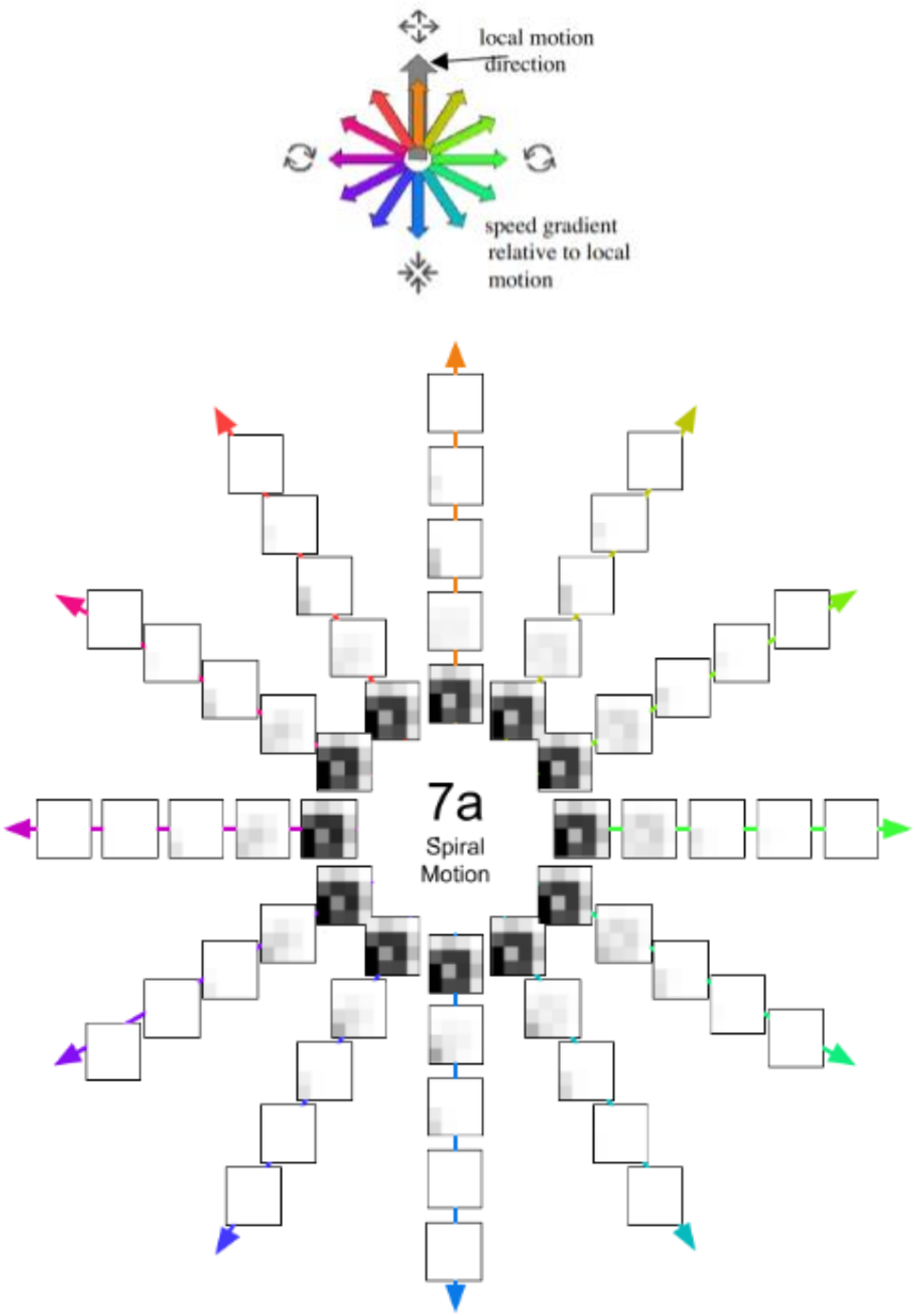

Figure 3.9 (continued): 7a Spiral Motion shows the generalized spiral output in Area 7a. This shows a close-up view of the GT output for 7a Spiral Motion.

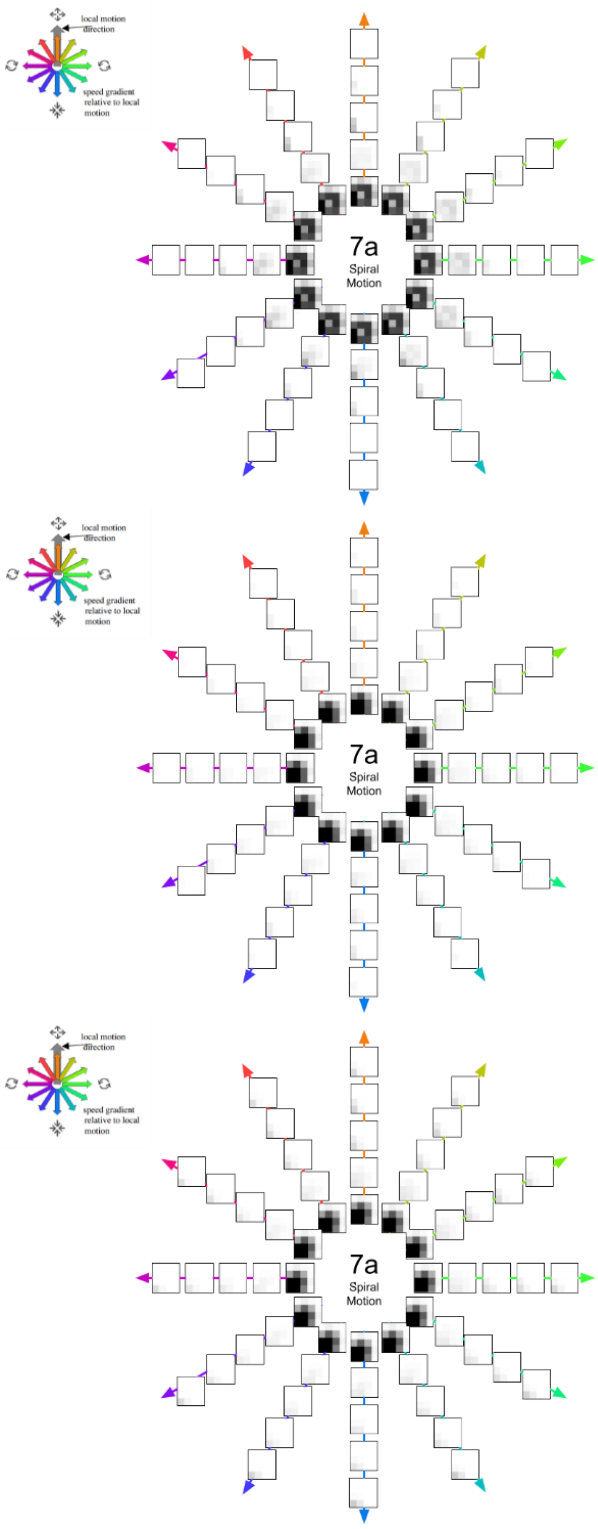

Figure 3.9 (continued): Both the 2-Frame and 6-Frame model detect the clockwise rotation (around δ=90˚) of the soccer ball and the contraction (around δ=180˚) of the volleyball. Inner angle δ of the peak response is close to groundtruth,

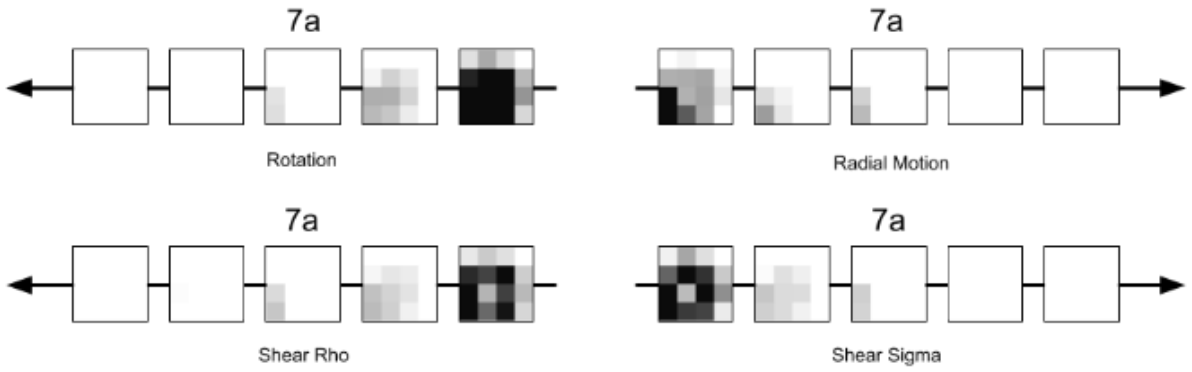

Figure 3.9 (continued): Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the GT outputs for AP.

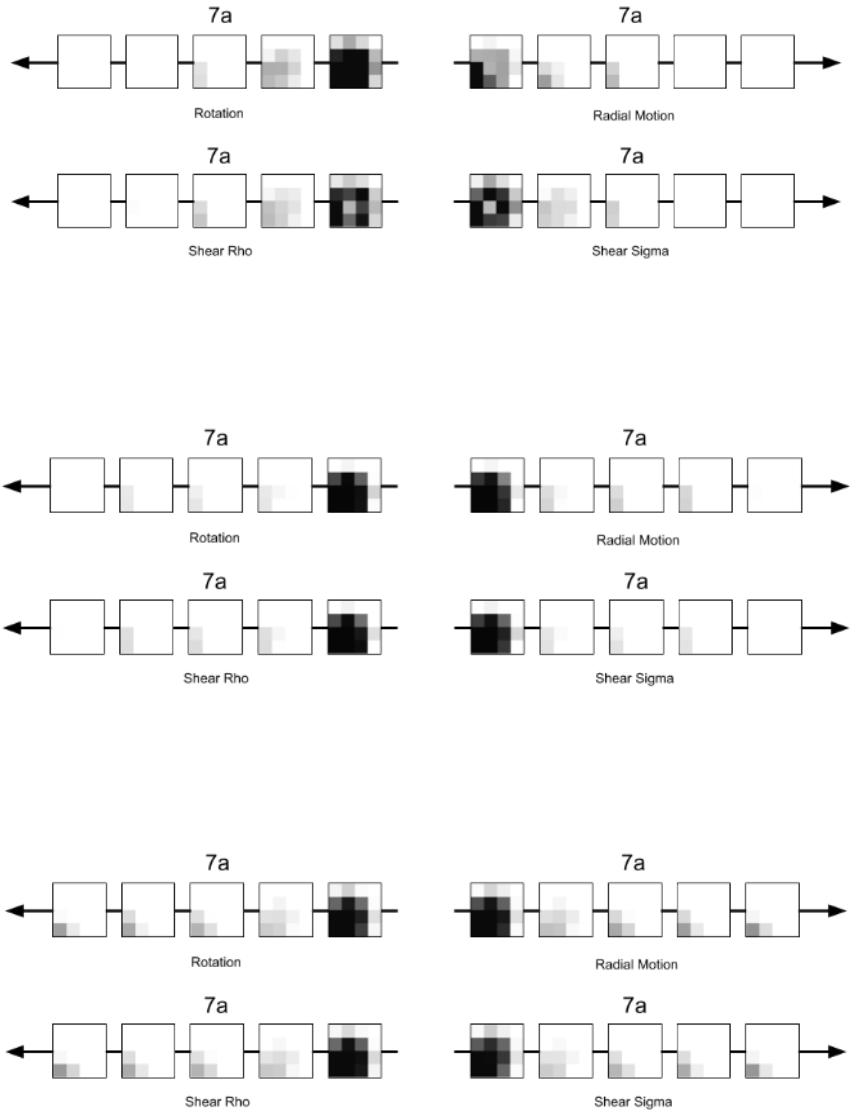

Figure 3.9 (continued): Both the 2-frame and 6-frame model detect radial motion of the volleyball in AP. The speed selectivity of the peak response from the 2-frame model is closer to the groundtruth. Both models also detect the rotation of the soccer ball among the top detected motion patterns.

We now examine some examples with deformation. Figure 3.10 shows a sequence from Blender-MP dataset containing a Grassy Rectangle deforming from normal strain on a still background. We compare the GT outputs, 2-frame model outputs, and 6-frame model outputs from AP in Figure 3.11. Close-up views of the 2-frame and 6-frame model outputs are available in Figure A.4.

Figure 3.10: Sequence from Blender-MP of a grassy rectangle deforming from normal strain on a still background. Inputs to 2-Frame Area V1 are the 2 center frames and inputs to 6-Frame Area V1 are the 6 frames.

For the outputs from AP, Both the 2-frame and 6-frame model detect the deformation from normal strain (Shear Rho) with strong response. The responses from the 2-frame model are closer to the GT responses and the peak responses have more accurate speed selectivity. For the 6-frame model, the speed selectivities of the peak responses are slightly different from ground truth, as the 6-Frame-V1 model is less accurate than the 2-Frame-V1 model (See 3.5.2).

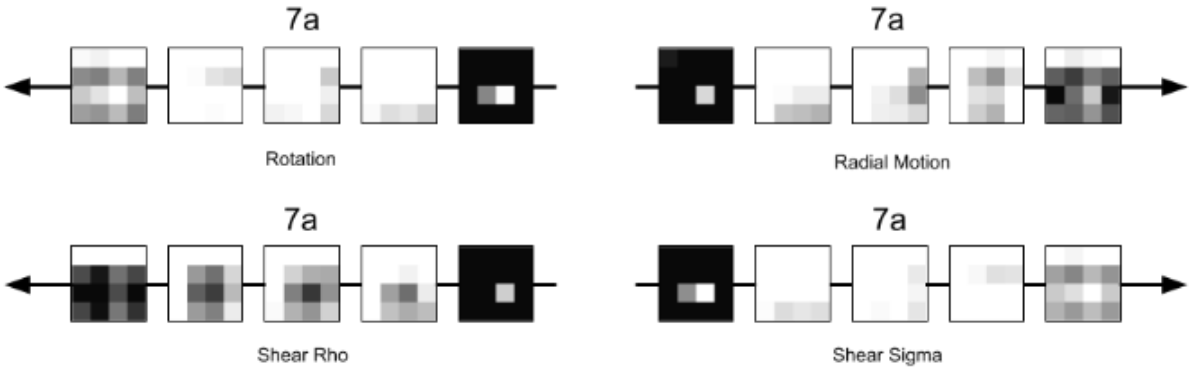

Figure 3.11: Output of area computations of AP of ST-Motion-Net for sequence in Figure 3.10. Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the GT outputs for AP.

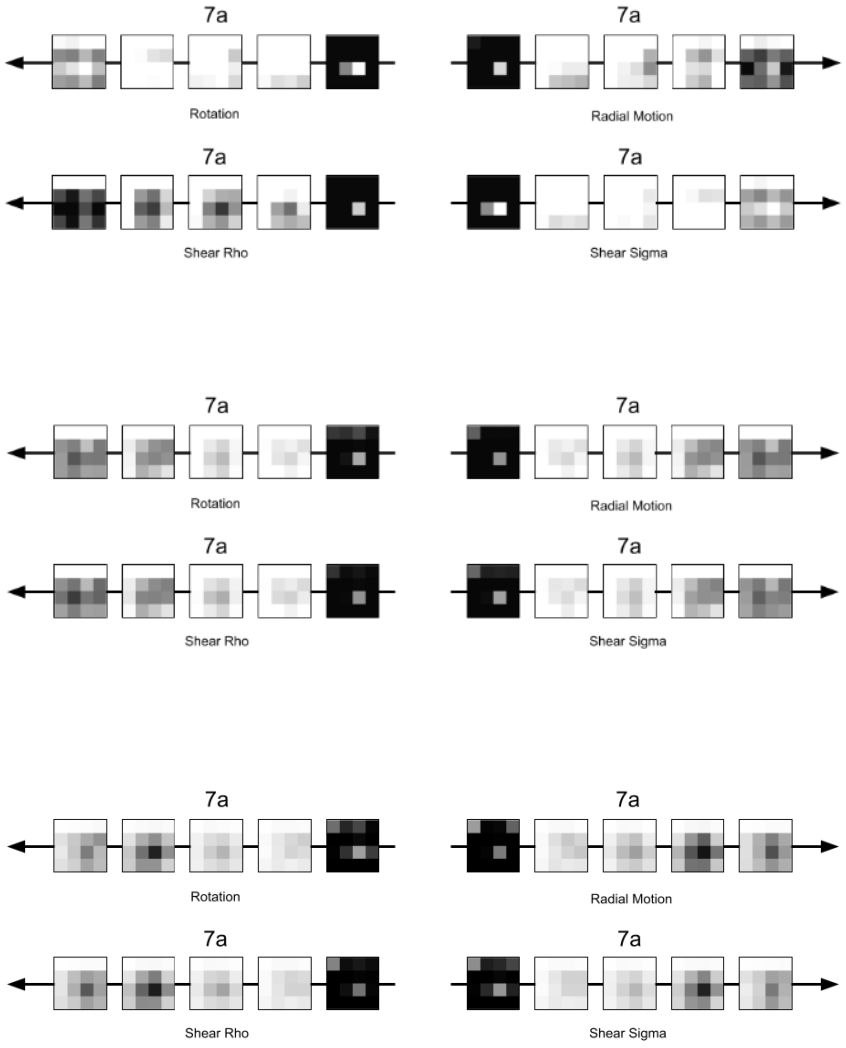

Figure 3.11 (continued): The outputs from top to bottom are the GT output, 2-frame model output, and 6-frame model output. Both the 2-Frame and 6-Frame model detect deformation from normal strain (See Shear Rho) with strong response. The responses from the 2-Frame model are closer to the ground truth responses and have more accurate speed selectivity. For the 6-Frame model, the speed selectivities of the peak responses are slightly different from groundtruth.

Figure 3.12 shows a sequence from Blender-Complex Dataset containing an UFO (Unidentified Flying Object) deforming from horizontal shear on a still background. An UFO is an object or a phenomenon in the sky which cannot be readily identified or explained by an observer [178]. We compare the GT outputs, 2-frame model outputs, and 6-frame model outputs from AP in Figure 3.13. Close-up views of the 2-frame and 6-frame model outputs are available in Figure A.5.

Figure 3.12: Sequence from Blender-MP of an UFO deforming from horizontal shear on a still background. Inputs to 2-Frame Area V1 are the 2 center frames and inputs to 6-Frame Area V1 are the 6 frames.

For the outputs from AP, both the 2-Frame and 6-Frame model detect deformation from shear strain as the peak response in AP. The speed selectivity of the neuron with peak response is the same as the ground truth. For the 2-Frame model, the response for radial motion is quite close to the peak response, shear strain (see Shear Sigma). For the 6-Frame model, the response for rotation is close to the peak response, shear strain.

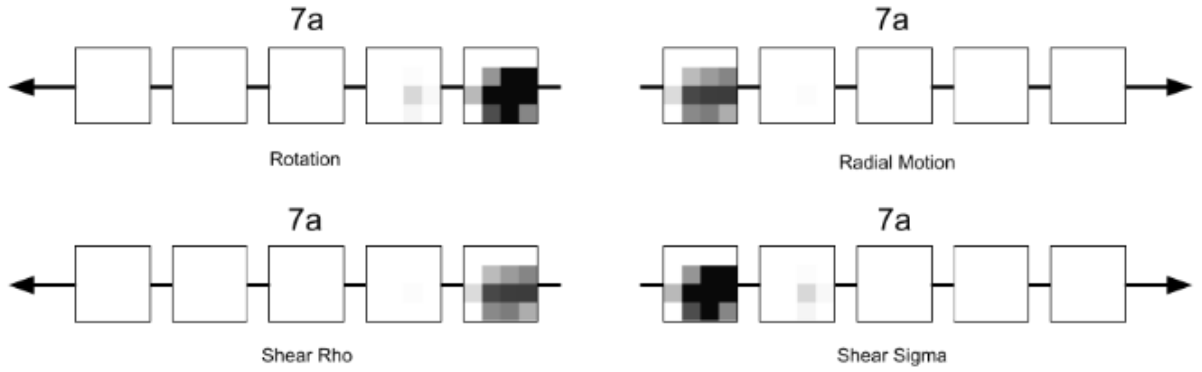

Figure 3.13: Output of area computations of AP of ST-Motion-Net for sequence in Figure 3.12. Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the GT outputs for AP.

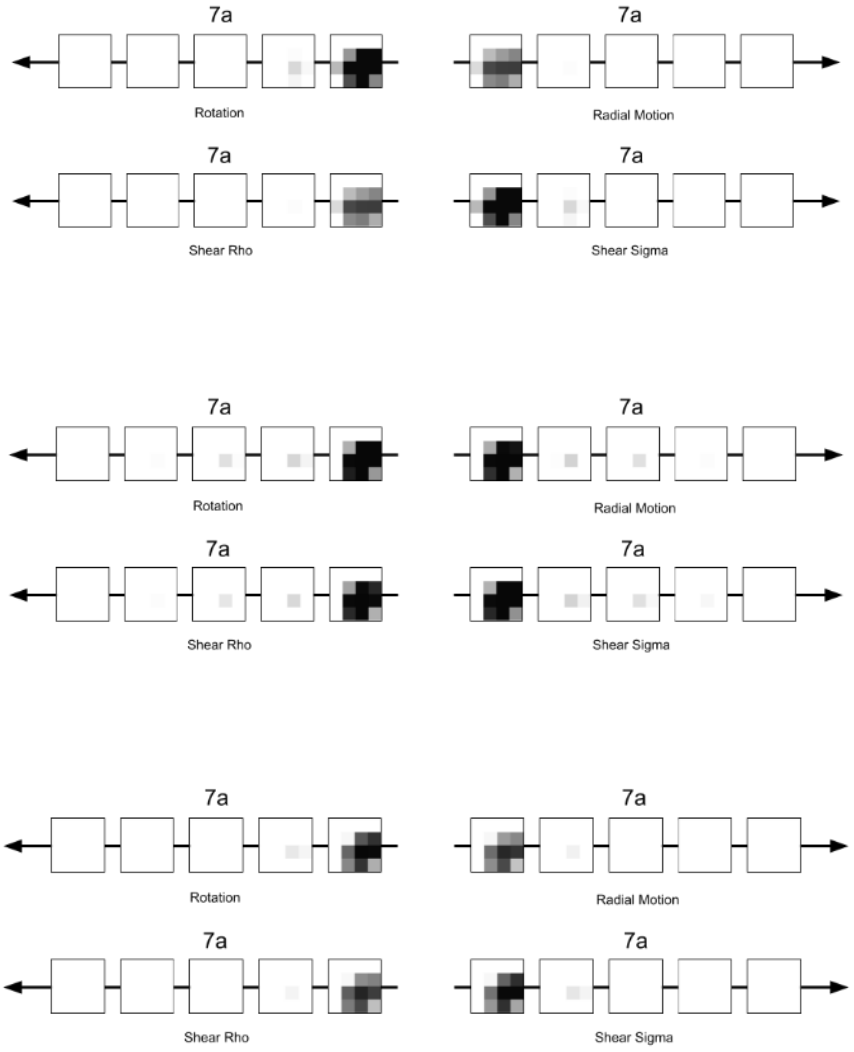

Figure 3.13 (continued): The outputs from top to bottom are the GT output, 2-frame model output, and 6-frame model output. Both the 2-Frame and 6-Frame model detect deformation from shear strain (See Shear Sigma) as the peak response in AP. The speed selectivity of the detection is the same as the groundtruth. For the 2-frame model, the response for radial motion is quite close to the peak response, shear strain (See Shear Sigma). For the 6-frame model, the response for rotation is close to the peak response, shear strain.

In Tsotsos et al. (2005) [26], the output feature maps from the motion hierarchy can be quite noisy in lower areas. An example is shown in Figure 7.38 of [85], where the feature maps for V1 and MT have substantial noise in regions corresponding to the background of the frames. Apparently, these noises are not present on the feature maps for lower areas in ST-Motion-Net. The use of CNN models and training data with accurate ground truth (i.e. computed from GT optical flow) reduced the errors in the predictions. The output feature maps from ST-Motion-Net are cleaner.

3.6 Conclusion and Future Works

3.6.1 Conclusion

In this chapter, we described a learnable motion processing hierarchy for attending to visual motion, ST-Motion-Net. We tested each area of ST-Motion-Net on the motion pattern detection task using two datasets, Blender-MP and Blender-Complex. The computed responses from each area of ST-Motion-Net are fairly accurate compared to GT responses on sequences with different motion patterns, such as translation, rotation, radial motion, and deformation. Mixed motion patterns (e.g. rotation + radial motion) or motion around occlusion boundary can also be detected by the model. The 2-Frame-V1 model has better performance than the 6-Frame-V1 model. We will test the motion hierarchy on motion pattern localization and motion segmentation tasks in Chapter 4 and 5 using the 2-Frame-V1 model. For ST-Motion-Net, the noise in the background regions on the outputs from lower areas (i.e. V1 and MT) of the motion hierarchy are no longer present, compared to Tsotsos et al. (2005) [26]. Thus, the output feature maps from ST-Motion-Net are also cleaner.

3.6.2 Future Works

Currently, the motion model can detect motion with speed around 0.5 pixels / frame and up to 2.0 pixels / frame. Faster motion is detected by neurons with speed selectivity of 2.0 pixels / frame. Additional neurons can be implemented in the motion hierarchy to detect very slow or fast motion patterns. This can help with detection of speed gradients, spiral motion patterns, and motion patterns in AP. Very fast or slow motion patterns can also be localized more accurately.

6-Frame-V1 is currently pretrained on input subsequences with 2 unique central frames, which are repeated to produce 6 input frames. A new dataset similar to ChairsSDHom extended [12] with 6 unique frames per input subsequence can potentially improve the performance of 6-Frame-V1. GT feature maps for each area are currently stored as 8-bit images, due to storage cost. Storing the feature maps as 16-bit images or NumPy arrays will improve the quality of the ground truth. Thus, model predictions will be more accurate, after training.

Chapter 4

Motion Pattern Localization on ST-Motion-Net

In this chapter, we evaluate ST-Motion-Net on the motion pattern localization task using Blender-MP and Blender-Complex dataset. We test localization of both zero speed and non-zero speed motion patterns from each area of ST-Motion-Net on the two datasets.

4.1 Motion Pattern Localization Task

In section, we describe the task in terms of input, output, and evaluation metric.

Input. Assume a temporal sequence of images is given. Input to the motion processing hierarchy is a continuous subsequence of T images.

Output. For each input subsequence to the model, outputs from the motion pattern detection task (See 3.4) plus accurate location of the detected motion categories should be produced (See examples in Figure 4.1).

4.1.1 Evaluation Metric

For the motion pattern localization task, the output localization mask contains the locations of the motion patterns detected by the neurons in an area. For performance evaluation, we use the standard intersection over union score (IoU) between the predicted and GT motion pattern localization masks, $\mathrm{IoU} = \frac{|l_t \cap l_o|}{|l_t \cup l_o|}$. This score measures the area of overlap between the predicted localization mask (l_t) and GT localization mask (l_o) over the union of them. If neurons

with zero speed selectivity ($v = 0.0$) are present in an area, we evaluate the localization performance for detections from the zero speed ($v = 0.0$) and non-zero speed ($v > 0.0$) selectivity neurons separately. The corresponding IoU scores are denoted by $IoU_{v=0.0}$ and $IoU_{v>0.0}$.

Figure 4.1: Sequence of a rainbow hot air balloon moving towards the top-left corner of the frames and the GT localization of zero speed and non-zero speed motion patterns.

4.2 Attention Initialization

Once the forward process of ST-Motion-Net outputs the feature maps up to an area Λ, we generate an initialization signal to determine the set of elements on the top gating layer of Λ. We first follow [26] to group the motion categories of each unit into subsets representing motion categories that are mutually exclusive or can coexist. An initialization strategy then selects the categories in each subset for which the TD selection process (See 4.4) is activated.

4.2.1 Mutually Exclusive Motion Categories

We experiment with 2 initialization strategies from [4] to produce the attention signal for TD initialization of mutually exclusive motion categories. We follow [26] to implement the max /

argmax function using Parametric WTA (P-WTA) which returns winners within a safe margin θ from the winner so that multiple winners may be selected. We denote the computed detection strength for motion category k of unit i by $M_{i,k}$. We search over a range of θ values on the validation set of the dataset to find the θ value that is tight enough to improve the IoU score.

Top-1 Strategy

This strategy finds the motion categories with the maximum detection strengths and then sets the signal for the motion categories to 1.

$$d^{top-1} = \left\{ d_{ij} = 1 \mid j \in \underset{k}{\operatorname{argmax}} M_{i,k} \right\}_{i=0}^{A} \tag{4.1}$$

Thresholding Strategy

This strategy combines the Top-1 Strategy with thresholding to reduce selection of targets with low confidence (false alarm).

$$d^{\theta} = \left\{ d_{ij} = 1 \mid j \in \underset{k}{\operatorname{argmax}} M_{i,k}, \; M_{i,j} > \theta^{attention} \right\}_{i=0}^{A} \tag{4.2}$$

Here, $\theta^{attention}$ is the cross-validated threshold.

We search over a range of $\theta^{attention}$ values on the validation set of the dataset to find the $\theta^{attention}$ value that improves the IoU score.

4.2.2 Coexisting Motion Categories

We experiment with 2 initialization strategies to generate the initialization signal for TD initialization of motion categories that can co-exist. We denote the computed detection strength for motion category k of unit i by $M_{i,k}$.

Select-All Strategy

This strategy initializes coexisting motion categories by setting the signal for the categories to 1.

$$d^{all} = \{d_{ij} = 1 \mid j \in C^{motion}\}_{i=0}^{A} \quad (4.3)$$

Here, C^{motion} is the subset of motion categories that can coexist for each unit.

Thresholding Strategy

This strategy combines Select-All Strategy with thresholding to reduce selection of targets with low confidence (false alarm).

$$d^{all} = \{d_{ij} = 1 \mid j \in C^{motion}, \ M_{i,j} > \theta^{attention}\}_{i=0}^{A} \quad (4.4)$$

Here, C^{motion} is the subset of motion categories that can coexist for each unit.

$\theta^{attention}$ is the cross-validated threshold.

We search over a range of $\theta^{attention}$ values on the validation set of the dataset to find the $\theta^{attention}$ value that improves the IoU score.

4.2.3 Combination of Motion Categories

If the motion categories contain subsets of categories that are mutually exclusive and categories that can coexist, then we initialize each subset of categories based on 4.2.1 and 4.2.2.

$$d^{comb} = \bigcup_i E_i \cup \bigcup_i C_i \tag{4.5}$$

Here, E_i is the set of winners in the ith subset of mutually exclusive categories (selected using 4.2.1). C_i is the set of categories that can coexist in the ith subset of coexisting categories (selected using 4.2.2).

4.2.4 Properties of Motion Categories

As mentioned earlier, we follow [26] to group the motion categories of the top area into subsets that are mutually exclusive or can coexist. An initialization strategy then selects the categories in each subset for which the TD process is activated. The initialization of attention described in 4.2.1-4.2.3 depends on the properties of the motion categories in each area of ST-Motion-Net. The properties of different subsets of motion categories in the model are listed in Table 4.1. Translation, Spiral Motion, or AP motion categories are mutually exclusive, as only the top detections in each subset are selected. The selections from the subsets can coexist.

Motion Categories	Property
Translation	Mutually Exclusive
Spiral Motion	Mutually Exclusive
Translation and Spiral Motion	Coexist
AP (ART, ARD, ADR, ADS)	Mutually Exclusive
AP and AS Spiral Motion	Coexist
AP and AT Translation	Coexist

Table 4.1: Properties of different subsets of motion categories in ST-Motion-Net

4.3. Attention Initialization for Motion Pattern Localization

For motion pattern localization, we initialize the attention signal differently for localization of detections from zero speed (v = 0.0) and non-zero speed (v > 0.0) selectivity neurons. Given a subset of motion categories $S_{\text{categories}}$ containing motion categories that are mutually exclusive or can coexist (See Table 4.1), we select from the corresponding subset $\mathrm{S}_{\text{select}}$ during attention initialization using 4.2.

For localization of detections from zero speed selectivity neurons,

$$\mathrm{S}_{\text{select}} = \{\mathrm{i} \in \mathrm{S}_{\text{categories}} \mid \mathrm{v}_{\mathrm{i}} = 0.0\} \tag{4.6}$$

For localization of detections from non-zero speed selectivity neurons,

$$\mathrm{S}_{\text{select}} = \{\mathrm{i} \in \mathrm{S}_{\text{categories}} \mid \mathrm{v}_{\mathrm{i}} > 0.0\} \tag{4.7}$$

Here, v_{i} is the speed of motion category i.

We experiment with strategies from 4.2.1-4.2.2 to select motion categories from subset $\mathrm{S}_{\text{select}}$ which contains mutually exclusive or co-existing motion categories. We search over a range of hyper-parameters on the validation set of the benchmark dataset and find the following tunings (See Table 4.2) improve the motion localization IoU score.

Dataset	Blender-MP						Blender-Complex					
	v = 0.0			v > 0.0			v = 0.0			v > 0.0		
Top Area	θ	θ_m	θ_c	θ	θ_m	θ_c	θ	θ_m	θ_c	θ	θ_m	θ_c
2-Frame V1	0.01	0.9935	0.9935	0.01	0.5000	0.5000	0.01	0.9800	0.9800	0.01	-	-
MT	0.01	0.9990	0.9990	0.01	0.5500	0.5500	0.01	0.9997	0.9997	0.01	-	-
MST	0.10	-	-	0.01	-	-	0.15	-	-	0.35	-	-
7a	0.01	-	-	0.01	-	-	0.10	-	-	0.40	-	-

Table 4.2: Settings for margin θ of P-WTA; and thresholds θ_m and θ_c for Thresholding Strategy on mutually exclusive and coexisting motion categories. A dash sign indicates the corresponding Thresholding Strategy is not used during attention initialization for the motion pattern localization task.

4.4. Top-down Selection

The TD selection process within ST-Motion-Net is activated using the initialized attention signal from 4.2-4.3. At each layer the TD gating activities are computed from the BU hidden activities. The selections are then passed down to the layer below.

It is shown in [4] the TD gating activities and BU hidden activities can be utilized to learn a model for object segmentation. In this project, we follow [26] to compute the motion pattern localization directly using the TD selection process. We use the ST Library implementation from the STNet model [3] for the TD process.

During the TD process for localization of non-zero speed (v > 0.0) motion patterns, the gating sheet for the top gating layer of some areas below the top area may contain gating values from zero speed (v = 0.0) motion categories or vice versa. We set such gating values to zero to initialize the gating sheet of the top gating layer of such areas for motion pattern localization.

4.5 Motion Pattern Localization

After feeding a sequence of RGB frames into the model, the BU process predicts the motion feature maps up to an area Λ. We then start the TD process from the top gating layer of Λ. Once the attention signals are initialized and propagated downward to the input layer, the attention map for the first frame of 2 center input frames, $\mathrm{A}_1^{\mathrm{inp}}$, is generated by collapsing the corresponding channels of the gating volume [3].

$$\mathrm{A}_1^{\mathrm{inp}} = \sum_{\mathrm{i}=\mathrm{c}_1}^{\mathrm{c}_1+2} \mathrm{g}_1^{\mathrm{inp}} \tag{4.8}$$

Here, g_1^{inp} is a 2D gating sheet of the input layer.

Here, c_1 is the starting index of the first frame of the 2 center frames of the input layer.

We then postprocess the attention map $\mathrm{A}_1^{\mathrm{inp}}$ by setting all collapsed values below the pth percentile to zero. We denote the postprocessed attention map by $\mathrm{A}_1^{\mathrm{post}}$. Here, p is the cross-validated hyperparameter. We search over a range of values for p on the validation set of each benchmark dataset and find the following tunings of p improve the motion pattern localization IoU score for zero-speed (v = 0.0) and non-zero (v > 0.0) speed motion patterns.

Dataset	**Blender-MP**		**Blender-Complex**	
Top Area	**v = 0.0**	**v > 0.0**	**v = 0.0**	**v > 0.0**
2-Frame V1	-	0.05	-	0.10
MT	-	0.10	-	0.10
MST	-	0.05	-	0.05
7a	-	0.05	-	0.05

Table 4.3: Settings for percentile p, during postprocessing stage of motion pattern localization. A dash sign indicates postprocessing is not used for the motion pattern localization task.

We then compute a coarse motion pattern localization, S_1^{coarse} from the postprocessed attention map, A_1^{post}, by setting all non-zero values to 1.0. The coarse motion localization, S_1^{coarse}, can still contain 'holes' in the motion region and noise around the motion boundary. We further refine the motion pattern localization mask, S_1^{coarse}, in OpenCV using morphological filters. We apply an opening transformation with a kernel of size k_o, followed by a closing transformation with a kernel of size k_c. We denote the refined motion localization as S_1^{fine}. Here, k_o and k_c are cross-validated hyperparameters.

For localization of zero-speed (v = 0.0) motion patterns, refinement by morphology may 'cover up' the non-zero speed motion patterns or remove too many pixels of the zero-speed motion pattern. Therefore, we do not use refinement, during localization of zero speed motion patterns. For localization of non-zero speed (v > 0.0) motion patterns, we search over a range of values and find that the following tunings of k_o and k_c improve the motion pattern localization IoU score.

Dataset	**Blender-MP**				**Blender-Complex**			
	v = 0.0		**v > 0.0**		**v = 0.0**		**v > 0.0**	
Top Area	**k_o**	**k_c**	**k_o**	**k_c**	**k_o**	**k_c**	**k_o**	**k_c**
2-Frame V1	-	-	3	25	-	-	3	23
MT	-	-	3	29	-	-	3	23
MST	-	-	3	31	-	-	3	25
7a	-	-	3	31	-	-	3	25

Table 4.4: Settings for opening kernel size k_c and closing kernel size k_o during refinement of motion pattern localization using morphological filters. A dash sign indicates refinement is not used for the motion pattern localization task.

We predict the motion pattern localization from the refined mask, S_1^{fine}.

4.6. Experiments and Results

We evaluate each area of ST-Motion-Net on the motion pattern localization task with Blender-MP and Blender-Complex dataset (See 3.5.1). We test each area of the ST-Motion-Net on localization of zero-speed (v = 0.0) and non-zero speed (v > 0.0) motion patterns.

4.6.1 Datasets

We evaluate each area of ST-Motion-Net on the motion pattern localization task using two datasets, Blender-MP and Blender-Complex (See 3.5.1). GT motion pattern localization masks are available for each sequence of each dataset. GT motion pattern localization masks for zero speed (v = 0.0) and non-zero speed (v > 0.0) motion categories are computed by thresholding the speed feature map, which is obtained from the GT optical flow map.

For localization of zero speed motion patterns, a pixel **p** is assigned a value of 255 on the mask if the speed of the pixel $v_p \leq \theta_3$. For localization of non-zero speed motion patterns, a pixel **p** is assigned a value of 255 on the mask if the speed of the pixel $v_p \geq \theta_3$. Here, $\theta_3 = 0.25$ is the speed threshold for zero-speed (v = 0.0) motion categories (See 3.2).

4.6.2 Motion Pattern Localization Benchmark

For the motion pattern localization task, the goal is to compute a localization mask, which contains pixels of the detected motion patterns. Since 3D Convolution is currently not implemented in the ST Library [3], we evaluate ST-Motion-Net on motion pattern localization task using the 2-Frame Early Fusion model for Area V1 (See 3.1.1). We evaluate the localization performance on the Blender-MP and Blender-Complex dataset using the mean

intersection over union (IoU) score between the predicted localization masks and the GT localization masks (See 4.1.1).

Option	Blender-MP							
	v = 0.0				v > 0.0			
	2-Frame-V1	**MT**	**MST**	**7a**	**2-Frame-V1**	**MT**	**MST**	**7a**
Without Threshold Strategy	0.52976	0.52540	0.52541	0.52538	0.38154	0.38626	0.38443	0.35852
Threshold Strategy	**0.53177**	0.52544	-	-	0.38192	0.38700	-	-
Postprocessing	-	-	-	-	0.38322	0.38986	0.38655	0.36102
Morphology Refine	-	-	-	-	0.57543	**0.59271**	0.57527	0.53335

Option	Blender-Complex							
	v = 0.0				v > 0.0			
	2-Frame-V1	**MT**	**MST**	**7a**	**2-Frame-V1**	**MT**	**MST**	**7a**
Without Threshold Strategy	0.52576	0.52120	0.52114	0.52071	0.33933	0.32034	0.26777	0.22497
Threshold Strategy	**0.52747**	0.52125	-	-	-	-	-	-
Postprocessing	-	-	-	-	0.34050	0.32132	0.26873	0.22541
Morphology Refine	-	-	-	-	**0.52823**	0.50802	0.43760	0.37232

Table 4.5: Performance of ST-Motion-Net on the motion pattern localization task from each top area on the Val Set of Blender-MP and Blender Complex dataset after each option is applied in order. A dash sign indicates the corresponding option is not used, according to Table 4.2-4.4.

We first evaluate ST-Motion-Net on the motion pattern localization task without Thresholding Strategy for attention initialization. We use Top-1 Strategy for mutually exclusive motion categories and Select-All Strategy for co-existing motion categories, during attention initialization. We use margin θ found in 4.3 for the initialization of mutually exclusive motion categories. Table 4.5 shows the performance of ST-Motion-Net on the motion pattern localization task without Thresholding Strategy.

Next, we evaluate ST-Motion-Net on the motion pattern localization task using settings for Thresholding Strategy found in 4.3. As described in 4.2.1-4.2.2, Thresholding Strategy combines Select-All / Top-1 Strategy with thresholding to reduce selection of targets with low

confidence. Motion categories which responses are below or equal to the threshold $\theta^{\text{attention}}$ are filtered from the selection. We report the performance of ST-Motion-Net using settings for Thresholding Strategy found in 4.3 in Table 4.5.

Thresholding Strategy slightly improves the motion pattern localization performance from some top areas. For localization of zero speed motion patterns, the mean IoU score on the Validation set of Blender-MP and Blender-Complex increases for top areas, V1 and MT. For localization of non-zero speed motion patterns, the mean IoU score on the Validation set of Blender-MP also improves for top areas, V1 and MT. As low confidence motion categories are filtered out, the output localization masks from the top-down process contain less errors (See Figure 4.2).

We then benchmark ST-Motion-Net with postprocessing percentile p found in 4.4. The performance of ST-Motion-Net with percentile postprocessing is reported in Table 4.5. For localization of non-zero speed motion patterns, the performance of ST-Motion-Net on the motion pattern localization task on Blender-MP and Blender-Complex dataset further improves after percentile postprocessing for all top areas. Some of the values on the collapsed attention map A_1^{inp} are predictions with low confidence. Thus, the performance on the motion pattern localization task improves after filtering them (See Figure 4.2).

The coarse motion pattern localization mask S_1^{coarse} can still contain 'holes' and small artifacts after postprocessing. We further refine S_1^{coarse} using morphological filters in OpenCV. As described in 4.5, the refinement consists of an opening transformation with a kernel of size k_o, followed by a closing transformation with a kernel of size k_c. The refinement computes a fine motion pattern localization S_1^{fine}. We then benchmark ST-Motion-Net with this additional refinement step using settings for k_c and k_o found in 4.5. We report the performance of ST-Motion-Net on motion pattern localization task with refinement in Table 4.5.

For localization of non-zero speed motion patterns, the performance of ST-Motion-Net improves significantly after refinement using morphological filters. The ‘holes’ on the mask are mostly closed and some artifacts may remain (See Figure 4.2). Motion pattern localization from MT achieves the best IoU score on the Blender-MP Validation set. Motion pattern localization from V1 achieves the best performance on the Blender-Complex Validation dataset.

As mentioned in 4.5, refinement by morphology on the localization of zero-speed motion patterns may ‘cover up’ the non-zero speed motion patterns or remove too many pixels of the zero-speed motion pattern. Therefore, we do not use refinement for localization of zero speed motion patterns. For zero-speed motion patterns, localization from top area, V1, achieves the best performance on the Validation set of Blender-MP and Blender-Complex dataset.

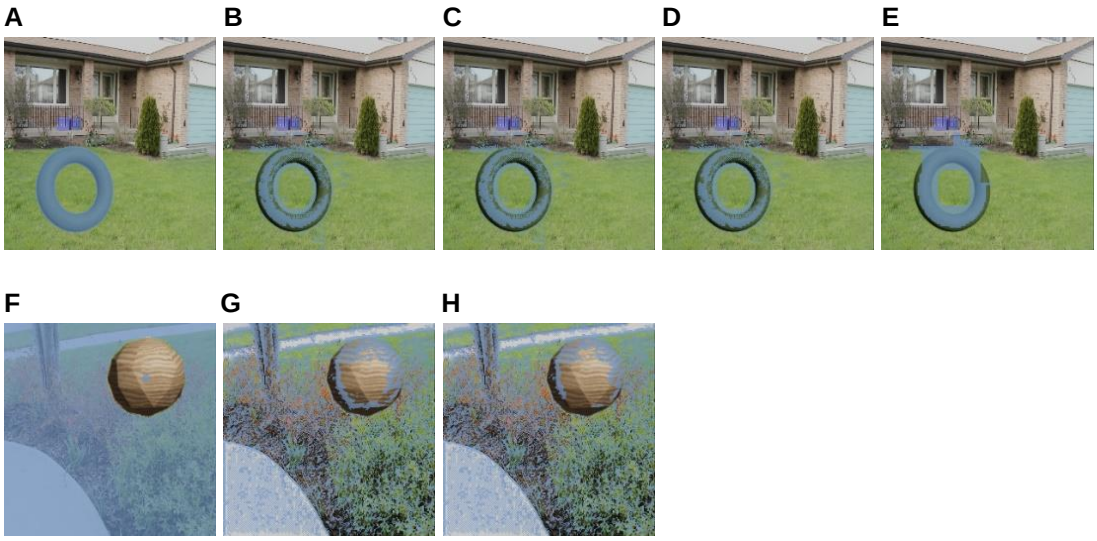

Figure 4.2: Comparison of motion pattern localization results after options in Table 4.5 are applied in order. The options include Without Thresholding Strategy for Attention Initialization (See 4.3), Thresholding Strategy (See 4.3), Postprocessing (See 4.5), and Morphology Refinement (See 4.5). For localization of non-zero speed ($v > 0.0$) motion patterns, all 4 options may be applied. For localization of zero-speed ($v = 0.0$) motion patterns, the first 2 options may be applied (See Table 4.5). (Continued on the following page.)

Figure 4.2 (continued): 1st row: localization of non-zero speed (v > 0.0) motion patterns from 2-Frame-V1. Left-Right: GT (A), Without Thresholding Strategy (B), Thresholding Strategy (C), Postprocessing (D), Morphology Refinement (E). 2nd row: localization of zero-speed motion pattern (v = 0.0) from 2-Frame-V1. Left-Right: GT (F), Without Thresholding Strategy (G), Thresholding Strategy (H). Localization errors reduced slightly after applying Thresholding Strategy (C or H). Errors further reduced after Postprocessing (D). 'Holes' on the localization mask are closed after Morphology Refinement (E).

We show the results for the motion pattern localization task on Blender-MP and Blender-Complex dataset in Table 4.6.

		v = 0.0			
Blender MP Val Sequence		**2-Frame-V1**	**MT**	**MST**	**7a**
0	Wooden Sphere	0.5923	0.5845	0.5845	0.5845
1	Small Stony Cube 1	0.5319	0.5253	0.5253	0.5253
2	Leafy Cylinder	0.4561	0.4422	0.4422	0.4421
3	Rock	0.5703	0.5647	0.5647	0.5647
4	Grassy Ring	0.5223	0.5179	0.5178	0.5178
5	Leafy Rectangle	0.5314	0.5243	0.5243	0.5243
6	Grassy Cone	0.5445	0.5394	0.5394	0.5394
7	Grassy Cylinder	0.5518	0.5468	0.5467	0.5465
8	Wooden Polyhedron	0.5071	0.5026	0.5024	0.5018
9	Gray Diamond	0.5295	0.5233	0.5233	0.5233
10	Grassy Rectangle 1	0.5529	0.5467	0.5467	0.5467
11	Stony Ellipsoid 1	0.5133	0.5092	0.5092	0.5092
12	Wooden Rectangle 1	0.4936	0.4894	0.4895	0.4895
13	Grassy Ellipsoid	0.5532	0.5448	0.5448	0.5448
14	Leafy Parallelepiped	0.4986	0.4925	0.4925	0.4925
15	Wooden Rectangle 2	0.5288	0.5165	0.5166	0.5165
16	Stony Sphere	0.5878	0.5817	0.5818	0.5818
17	2 Gray Stars	0.5178	0.5121	0.5121	0.5121
18	Stony Polyhedron 1	0.5104	0.5059	0.5059	0.5059
19	Stony Ring	0.5417	0.5388	0.5387	0.5387
	Mean IoU	**0.5318**	0.5254	0.5254	0.5254

		v > 0.0			
Blender MP Val Sequence		**2-Frame-V1**	**MT**	**MST**	**7a**
0	Wooden Sphere	0.5980	0.6301	0.5934	0.5899
1	Small Stony Cube 1	0.1001	0.1032	0.0575	0.0575
2	Leafy Cylinder	0.7300	0.7647	0.7529	0.7548
3	Rock	0.4567	0.4784	0.4627	0.4680
4	Grassy Ring	0.6067	0.5595	0.5457	0.5361
5	Leafy Rectangle	0.5833	0.6279	0.5882	0.5853

6	Grassy Cone	0.5880	0.6005	0.5597	0.4509
7	Grassy Cylinder	0.6467	0.6993	0.6529	0.6516
8	Wooden Polyhedron	0.6489	0.6459	0.6283	0.6299
9	Gray Diamond	0.5219	0.5490	0.5421	0.5490
10	Grassy Rectangle 1	0.7435	0.7342	0.7326	0.0000
11	Stony Ellipsoid 1	0.5444	0.6394	0.6345	0.6338
12	Wooden Rectangle 1	0.5711	0.6046	0.5966	0.5969
13	Grassy Ellipsoid	0.7080	0.7286	0.7111	0.7168
14	Leafy Parallelepiped	0.6399	0.6240	0.6100	0.6129
15	Wooden Rectangle 2	0.8357	0.8615	0.8257	0.8260
16	Stony Sphere	0.5833	0.5941	0.5869	0.5859
17	2 Gray Stars	0.4417	0.4290	0.4385	0.4363
18	Stony Polyhedron 1	0.5532	0.5584	0.5722	0.5722
19	Stony Ring	0.4076	0.4219	0.4137	0.4131
	Mean IoU	0.5754	**0.5927**	0.5753	0.5334

		v = 0.0			
	Blender MP Test Sequence	**2-Frame-V1**	**MT**	**MST**	**7a**
0	Large Stony Sphere	0.4972	0.4848	0.4836	0.4836
1	Grassy Cube	0.5246	0.5117	0.5111	0.5111
2	Large Rock	0.5430	0.5326	0.5326	0.5326
3	Stony Cone and Sphere	0.5360	0.5303	0.5302	0.5302
4	Stony Polyhedron 2	0.5027	0.4957	0.4952	0.4952
5	Wooden Cube	0.5188	0.5075	0.5070	0.5070
6	Stony Cone	0.5026	0.4964	0.4964	0.4964
7	Small Stony Cube 2	0.5492	0.5439	0.5435	0.5434
8	Wooden Cylinder	0.4733	0.4710	0.4709	0.4704
9	Clay Teapot	0.5705	0.5624	0.5624	0.5624
10	Grassy Rectangle 2	0.5419	0.5429	0.5429	0.5429
11	Stony Ellipsoid 2	0.5256	0.5217	0.5217	0.5217
12	Grassy Parallelepiped	0.5139	0.5056	0.5052	0.5052
13	Reflective Ellipsoid	0.5512	0.5407	0.5405	0.5405
14	Large Gassy Parallelepiped	0.4333	0.4520	0.4519	0.4509
15	Wooden Star	0.5856	0.5740	0.5740	0.5738
16	Stony Square	0.5281	0.5156	0.5156	0.5156
17	Leafy Star and Stony Star	0.5099	0.5028	0.5027	0.5027
18	Small Stony Sphere	0.5713	0.5656	0.5655	0.5655
19	Stony Cube	0.5667	0.5629	0.5629	0.5629
	Mean IoU	**0.5273**	0.5210	0.5208	0.5207

		v > 0.0			
	Blender MP Test Sequence	**2-Frame-V1**	**MT**	**MST**	**7a**
0	Large Stony Sphere	0.7185	0.7375	0.7002	0.7005
1	Grassy Cube	0.6478	0.6937	0.6535	0.6527
2	Large Rock	0.8365	0.8458	0.8298	0.8305
3	Stony Cone and Sphere	0.5393	0.5580	0.5310	0.2201
4	Stony Polyhedron 2	0.8457	0.8618	0.8555	0.8554

5	Wooden Cube	0.6510	0.6680	0.6908	0.6933
6	Stony Cone	0.7060	0.7147	0.6995	0.7013
7	Small Stony Cube 2	0.4265	0.4996	0.5241	0.5340
8	Wooden Cylinder	0.7394	0.7849	0.7533	0.7530
9	Clay Teapot	0.6742	0.7019	0.6563	0.6561
10	Grassy Rectangle 2	0.7416	0.8013	0.7841	0.7799
11	Stony Ellipsoid 2	0.8067	0.7913	0.7992	0.7979
12	Grassy Parallelepiped	0.6051	0.6129	0.5943	0.4678
13	Reflective Ellipsoid	0.7315	0.7546	0.7469	0.7475
14	Large Gassy Parallelepiped	0.6858	0.7193	0.6896	0.6863
15	Wooden Star	0.6170	0.6463	0.6162	0.6165
16	Stony Square	0.8198	0.8159	0.8195	0.8194
17	Leafy Star and Stony Star	0.2445	0.2682	0.2749	0.2749
18	Small Stony Sphere	0.7449	0.7747	0.7340	0.7289
19	Stony Cube	0.6904	0.7471	0.6911	0.6910
	Mean IoU	0.6736	**0.6999**	0.6822	0.6604

		v = 0.0			
	Blender-Complex Val Sequence	**2-Frame-V1**	**MT**	**MST**	**7a**
1	Airplane	0.5266	0.5212	0.5212	0.5211
11	Wooden Toy Car	0.5447	0.5405	0.5405	0.5405
19	Cow Balloon	0.5335	0.5280	0.5279	0.5277
21	Two Butterflies	0.5051	0.4953	0.4950	0.4935
	Mean IoU	**0.5275**	0.5212	0.5211	0.5207

		v > 0.0			
	Blender-Complex Val Sequence	**2-Frame-V1**	**MT**	**MST**	**7a**
1	Airplane	0.2591	0.1822	0.0963	0.0850
11	Wooden Toy Car	0.5578	0.5496	0.3644	0.2549
19	Cow Balloon	0.6698	0.6750	0.6544	0.5557
21	Two Butterflies	0.6262	0.6252	0.6352	0.5937
	Mean IoU	**0.5282**	0.5080	0.4376	0.3723

		v = 0.0			
	Blender-Complex Test Sequence	**2-Frame-V1**	**MT**	**MST**	**7a**
2	Hover Car	0.4982	0.4928	0.4928	0.4928
3	Rainbow Hot Air Balloon	0.5232	0.5149	0.5149	0.5149
4	Drone	0.5571	0.5523	0.5523	0.5523
5	Paper Airplane	0.5747	0.5684	0.5684	0.5684
6	Eagle	0.4404	0.4356	0.4356	0.4355
7	Butterfly	0.5601	0.5545	0.5544	0.5541
8	Seagull	0.5398	0.5342	0.5342	0.5342
9	Crab Balloon	0.4590	0.4618	0.4618	0.4617
10	Soccer Ball	0.5289	0.5221	0.5222	0.5221
12	Toy Airplane	0.5454	0.5398	0.5397	0.5397
13	Volleyball	0.5567	0.5483	0.5482	0.5481
14	Shopping Cart	0.5367	0.5341	0.5341	0.5340

15	Sword	0.5442	0.5395	0.5395	0.5395
16	Rusted Tire	0.5228	0.5160	0.5160	0.5159
17	Fighter Jet	0.4876	0.4836	0.4836	0.4836
18	Wooden Wheel	0.4930	0.4945	0.4942	0.4932
20	Flying Saucer	0.5210	0.5157	0.5157	0.5156
22	Soccer Ball and Volleyball	0.5657	0.5599	0.5599	0.5598
23	Two Drones	0.4954	0.4915	0.4915	0.4913
24	Two Hot Air Balloons	0.5572	0.5525	0.5523	0.5522
25	Cow Balloon and Penguin Balloon	0.5058	0.5009	0.5008	0.5008
	Mean IoU	**0.5244**	0.5197	0.5196	0.5195

		v > 0.0			
Blender-Complex Test Sequence		**2-Frame-V1**	**MT**	**MST**	**7a**
2	Hover Car	0.5079	0.5095	0.5011	0.2998
3	Rainbow Hot Air Balloon	0.5169	0.5082	0.4016	0.2116
4	Drone	0.1573	0.1137	0.0977	0.0732
5	Paper Airplane	0.2194	0.2023	0.2026	0.1447
6	Eagle	0.3389	0.3293	0.3313	0.2632
7	Butterfly	0.4233	0.4186	0.4048	0.3521
8	Seagull	0.5209	0.5148	0.5065	0.4369
9	Crab Balloon	0.5173	0.5090	0.4898	0.4247
10	Soccer Ball	0.6680	0.6582	0.6233	0.4782
12	Toy Airplane	0.5348	0.5297	0.5067	0.4162
13	Volleyball	0.6451	0.6405	0.6340	0.5118
14	Shopping Cart	0.3507	0.3457	0.3576	0.3151
15	Sword	0.1031	0.0770	0.0655	0.0536
16	Rusted Tire	0.5863	0.5881	0.5892	0.5389
17	Fighter Jet	0.1711	0.1314	0.0840	0.0804
18	Wooden Wheel	0.5169	0.5193	0.4948	0.4222
20	Flying Saucer	0.2855	0.2780	0.2713	0.2164
22	Soccer Ball and Volleyball	0.5046	0.4913	0.4831	0.3997
23	Two Drones	0.4364	0.4323	0.4201	0.3776
24	Two Hot Air Balloons	0.5445	0.5438	0.5032	0.3430
25	Cow Balloon and Penguin Balloon	0.5383	0.5325	0.5306	0.4291
	Mean IoU	**0.4327**	0.4225	0.4047	0.3233

Table 4.6: Performance results for the motion pattern localization task on the Val and Test set of Blender-Blender-MP and Blender-Complex dataset from 2-Frame-V1, MT, MST, and 7a for localization of zero-speed (v = 0.0) and non-zero (v > 0.0) speed motion patterns. Evaluation uses attention initialization settings found in 4.3; and postprocessing and refinement settings found in 4.5.

4.6.3 Qualitative Results

We now show some qualitative results from each area of ST-Motion-Net on the motion pattern localization task. We show the GT masks and the corresponding refined localization masks from each area of the 2-frame model for each example from the Test set of Blender-MP and Blender-Complex dataset in Figure 4.3.

For each example, the motion pattern localization from each area is relatively accurate compared to the ground truth. For localization of non-zero speed ($v > 0.0$) motion patterns, most pixels of the moving objects are localized. The localization is concentrated on the pixels of the motion patterns. Motion patterns of both geometric (e.g. Stony Square) and real-world objects (e.g. Seagull) can be localized. The model can also localize motion patterns of partially occluded objects (Soccer Ball and Volleyball) and objects with changes in illumination (Stony Polyhedron 2). The model may miss some regions with homogeneous intensity, as the motion patterns are difficult to detect (e.g. Seagull). Most false positive predictions seem to occur around the motion boundary of objects. Some noise may remain around the motion boundary, after refinement using morphology (e.g. Soccer Ball and Volleyball). This suggests the localization results can be used for moving object segmentation, after refinement based on appearance. For localization of zero speed ($v = 0.0$) motion patterns, most regions of the static background are localized. Some false positive predictions may occur on the moving object (e.g. Stony Square). Localization errors may increase slightly in higher areas, as the output feature maps become coarser (e.g. Stony Square).

Figure 4.3: GT Motion Pattern Localization and Localization from top areas, 2-Frame-V1, MT, MST, and 7a for example input subsequences from Blender-MP and Blender-Complex Test set. Left-Right: Groundtruth, 2-Frame-V1 Localization, MT Localization, MST Localization, 7a Localization. The first 3 examples are from the Blender-MP dataset and the last 3 examples are from the Blender-Complex dataset. Odd Rows: Localization of non-zero speed (v > 0.0) motion patterns.

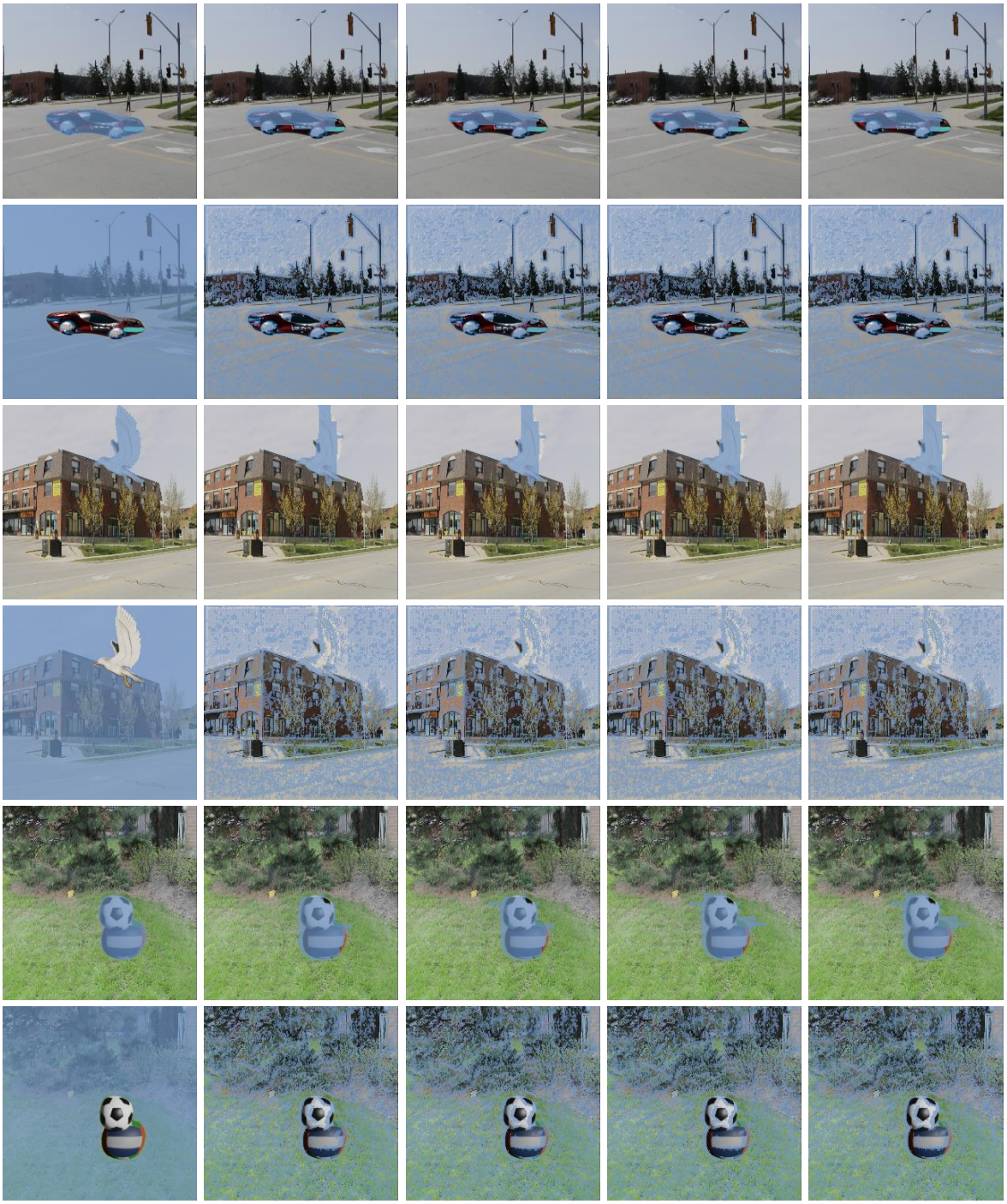

Figure 4.3 (continued): Most pixels of the moving objects can be localized. Localization is concentrated on the pixels of the motion patterns. Motion patterns of both geometric (e.g. Stony Square) and real-world objects (e.g. Seagull) can be localized. Motion patterns of partially occluded objects (e.g. Soccer Ball and Volleyball) and objects with changes in illumination (e.g. Stony Polyhedron 2) can be localized.

(Continued on the following page.)

Figure 4.3 (continued): The model may miss some regions with homogeneous intensity (e.g. Seagull), as the motion patterns are difficult to detect. Noise may remain around motion boundary of objects (e.g. Soccer Ball and Volleyball), after Morphology Refinement. Even Rows: Localization of zero speed (v = 0.0) motion patterns. Most regions of the static background can be localized. Some false-positive predictions may appear on the moving objects (e.g. Stony Square). Localization errors may increase slightly in higher areas as output feature maps become coarser (e.g. Stony Square).

4.7 Conclusion and Future Works

4.7.1 Conclusion

In this chapter, we tested each area of ST-Motion-Net on the motion pattern localization task using Blender-MP and Blender-Complex dataset. We evaluated each area on the localization of zero speed (v = 0.0) and non-zero speed (v > 0.0) motion patterns.

The motion pattern localization from each area is fairly accurate compared to the ground truth on examples from both datasets. The localization is concentrated on the pixels of the motion patterns. For non-zero speed motion patterns, motion patterns of both geometric and real objects can be localized. The model can also localize motion patterns of partially occluded objects and objects with illumination changes. The model may miss some regions with homogeneous intensity, as the motion patterns are difficult to detect. Some noise may remain around the motion boundary of objects. This suggests the localization can be used for moving object segmentation, after some postprocessing and refinement based on appearance. For zero-speed motion patterns, most regions of the static background can also be localized.

4.7.2 Future Works

3D convolution layer can be implemented in the ST Library [3] to test ST-Motion-Net on motion pattern localization with an input subsequence of multiple frames. The ST Library currently only supports square-sized inputs to convolution layers. Thus, support for non-square sized inputs can be implemented to test ST-Motion-Net on localization involving input subsequences with non-square frames. The ST Library can also be updated to support newer GPUs for faster computation during localization.

Chapter 5

Motion Segmentation Using 2-Frame-V1 of ST-Motion-Net

In this chapter, we evaluate 2-Frame-V1 of ST-Motion-Net on the motion segmentation task with 2 datasets, Blender-MP and Blender-Complex. We compare the motion segmentation results from 2-Frame-V1 with some recent motion segmentation models.

5.1 Motion Segmentation Task

As mentioned in 2.1, this thesis tests ST-Motion-Net on the binary motion segmentation task. In this section, we describe the task in terms of input, output, and evaluation metric.

Input. Assume a temporal sequence of images is given.

Output. Accurate location of the independently moving objects (See example in Figure 5.1).

5.1.1 Evaluation Metric

For the motion segmentation task, different metrics have been used for evaluation on relevant datasets [56]. Common evaluation metrics include intersection over union score (IoU) and precision, recall, and F-measure scores. For performance evaluation, we use the standard intersection over union score (IoU) [56] between the predicted and GT moving object segmentation masks, $\mathrm{IoU} = \frac{|\mathrm{m_t} \cap \mathrm{m_o}|}{|\mathrm{m_t} \cup \mathrm{m_o}|}$. This score measures the area of overlap between the predicted segmentation ($\mathrm{m_t}$) and GT segmentation ($\mathrm{m_o}$) over the union of them.

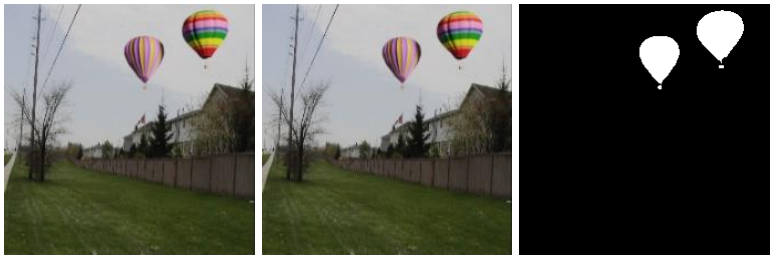

Figure 5.1: Sequence of 2 hot air balloons. The left balloon is contracting, and the right balloon is moving downwards to the left towards the left one. GT motion segmentation for the 1st frame.

5.2 Attention Initialization for Motion Segmentation

For motion segmentation from Area V1, we initialize the attention signal by selecting from non-zero speed ($v > 0.0$) motion categories in the subset, for each subset of motion categories that are mutually exclusive or can coexist. The selection from non-zero speed selectivity neurons is similar to that in 4.3.

For attention initialization, there are some parameters that are tuned on the validation set of each benchmark dataset. This includes margin θ of P-WTA; and thresholds θ_m and θ_c for Thresholding Strategy on mutually exclusive and coexisting motion categories. We search over a range of hyperparameters on the validation set of each benchmark dataset and find the following tunings (See Table 5.1) improve the motion segmentation IoU score.

Dataset	**Blender-MP**			**Blender-Complex**		
Top Area	θ	θ_m	θ_c	θ	θ_m	θ_c
2-Frame-V1	0.01	0.5000	0.5000	0.01	-	-

Table 5.1: Settings for margin θ of P-WTA; and thresholds θ_m and θ_c for Thresholding Strategy on mutually exclusive and coexisting motion categories. A dash sign indicates the corresponding Thresholding Strategy is not used during attention initialization for the motion segmentation task.

5.3 Top-down Selection

The top-down selection process for motion segmentation is similar to that for motion pattern localization (See 4.4). We compute the motion segmentation directly using the TD selection process, which computes the attention map for the input frames. We use the ST Library implementation from [3] for the TD process.

During the top-down process, the gating sheet for the top gating layer of some areas below the top area may contain gating values from zero speed motion categories. We set such gating values to zero to initialize the gating sheet of the top gating layer of such areas for motion segmentation.

5.4 Motion Segmentation

As mentioned earlier, we use the 2-Frame-V1 model for the motion segmentation task. For motion segmentation, the computation process is similar to the motion pattern localization task (See 4.5). Once the attention map for the first frame of 2 center input frames, $\mathrm{A}_1^{\mathrm{inp}}$ is computed, we postprocess the attention map $\mathrm{A}_1^{\mathrm{inp}}$ using the same method from 4.5 to compute the postprocessed attention map $\mathrm{A}_1^{\mathrm{post}}$. We search over a range of values for p on the validation set and find that p = 0.1 improves the motion segmentation IoU score for 2-Frame-V1 for both Blender-MP and Blender-Complex dataset.

We then compute a coarse motion segmentation mask, $\mathrm{M}_1^{\mathrm{coarse}}$, from the postprocessed attention map, $\mathrm{A}_1^{\mathrm{post}}$, by setting all non-zero values to 1.0. The coarse motion segmentation mask, $\mathrm{M}_1^{\mathrm{coarse}}$, may still be too sparse for the motion segmentation task. Therefore, we merge it with the coarse mask $\mathrm{M}_{2'}^{\mathrm{coarse}}$ from the postprocessed attention map of the same frame (i.e.

second frame) from the previous input subsequence $\mathrm{A}_{2,}^{\mathrm{post}}$. For postprocessing of attention map $\mathrm{A}_{2,}^{\mathrm{inp}}$, we also set p = 0.1. We denote the merged motion segmentation mask as $\mathrm{M}^{\mathrm{merge}}$.

$$\mathrm{M}^{\mathrm{merge}} = \mathrm{M}_{1}^{\mathrm{coarse}} + \mathrm{M}_{2,}^{\mathrm{coarse}} \tag{5.1}$$

The combined segmentation mask $\mathrm{M}^{\mathrm{merge}}$ can still contain 'holes' on the moving objects or noise around the motion boundary. We further refine the mask $\mathrm{M}^{\mathrm{merge}}$, in OpenCV using morphological filters. We apply an opening transformation with a kernel of size k_o, followed by a closing transformation with a kernel of size k_c. We denote the refined motion segmentation as M^{fine}. We search over a range of values on the validation set and find $k_o = 3$ and $k_c = 21$ improve the motion segmentation IoU score for both the Blender-MP and Blender-Complex dataset.

The refined segmentation mask M^{fine} may still contain larger 'holes' and some noise may remain around the moving objects. Therefore, we further refine the mask M^{fine} based on appearance using a CNN refiner, CascadePSP [89]. This model refines the mask M^{fine}, given the mask M^{fine} and the corresponding RGB frame of the mask. We chose hyperparameter L = 900 from [89]. We denote the CNN-refined mask as, M_{CPSP}^{fine}. We predict the motion segmentation from the refined mask, M_{CPSP}^{fine}.

5.5 Experiments and Results

We evaluate 2-Frame-V1 of ST-Motion-Net on the motion segmentation task with Blender-MP and Blender-Complex dataset (See 3.5.1). Blender-MP and Blender-Complex dataset are

created in Blender [37] and test the model on the segmentation of geometric and real-world moving objects, respectively.

5.5.1 Datasets

We evaluate 2-Frame-V1 on the motion segmentation task using two datasets, Blender-MP and Blender-Complex (See 3.5.1). The two datasets are created in Blender [37] and contain sequences of geometric and real-world objects, which are moving with different motion patterns. GT motion segmentation masks and camera intrinsics are available for each sequence of each dataset. The GT masks and camera intrinsics are generated using the VisionBlender tool [15].

5.5.2 Motion Segmentation Benchmark

For the motion segmentation task, the goal is to compute a segmentation mask, which contains pixels of independently moving objects. Since 3D Convolution is currently not implemented in the ST Library [3], we evaluate ST-Motion-Net on the motion segmentation task using the 2-Frame Early Fusion model for Area V1 (2-Frame-V1). We evaluate motion segmentation performance on the Blender-MP dataset and the Blender-Complex dataset using the mean intersection over union (IoU) score (See 5.1.1).

We first evaluate 2-Frame-V1 on the motion segmentation task without Thresholding Strategy for attention initialization. We use Top-1 Strategy for mutually exclusive motion categories and Select-All Strategy for co-existing motion categories, during attention initialization. We use margin $\theta = 0.01$ found in 5.2 for both Blender-MP and Blender-Complex dataset. Table 5.2 shows the performance of 2-Frame-V1 on the motion segmentation task on the Val set of Blender-MP and Blender Complex without using Thresholding Strategy.

Next, we evaluate 2-Frame-V1 on the motion segmentation task using settings for Thresholding Strategy found in 5.2. As described in 4.2.1-4.2.2, Thresholding Strategy combines Select-All / Top-1 Strategy with thresholding to reduce selection of targets with low confidence. Motion categories which responses are below or equal to the threshold are filtered from the selection. We report the performance on the Val set of the datasets in Table 5.2 using settings for Thresholding Strategy found in 5.2.

Thresholding Strategy slightly improves the motion segmentation performance from 2-Frame-V1 on the Blender-MP Val set. The mean IoU score on the Blender-MP Val set increases from 0.35425 to 0.35472. As low confidence motion categories are filtered out, the output motion masks from the top-down process contain less errors (See Figure 5.2).

We then benchmark 2-Frame-V1 with postprocessing percentile p = 0.1 found in 5.4. The performance of 2-Frame-V1 with percentile postprocessing is reported in Table 5.2. The performance of 2-Frame-V1 on the motion segmentation task further improves after percentile postprocessing. The mean IoU score on the Val set of Blender-MP and Blender-Complex dataset both increased. Some of the values on the collapsed attention map $\mathrm{A}_1^{\mathrm{inp}}$ are predictions with low confidence. Thus, the performance improves after filtering them (See Figure 5.2).

After percentile postprocessing, we merge the coarse mask of the current frame with that of the same frame (second frame) from the previous 2-frame input subsequence (See 5.4). The performance on the motion segmentation task further improves after the merge. The mean IoU score on the Val set of both datasets increased. The combined mask is denser than the original (See Figure 5.2). We report the results on the Val sets after the merge in Table 5.2.

The motion segmentation can still contain ‘holes’ and small artifacts after postprocessing. We further refine the motion segmentation using morphological filters in OpenCV. As described in 5.4, the refinement consists of an opening transformation with a kernel of size k_o, followed by a closing transformation with a kernel of size k_c. We benchmark 2-Frame-V1 with this additional

refinement step using settings for k_o and k_c found in 5.4. We also report the performance of 2-Frame-V1 on the motion segmentation task with morphological refinement in Table 5.2. The performance of 2-Frame-V1 improves significantly after refinement using morphological filters. Most 'holes' on the motion segmentation masks are closed and some artifacts remain around the moving objects (See Figure 5.2).

The refined segmentation mask may still contain some 'holes' and noise around the moving objects. Therefore, we also refine the segmentation mask using the appearance-based CNN refiner, CascadePSP [89]. The performance further improves after the second refinement. Larger 'holes' are filled, and the shapes of the moving objects are more accurate (See Figure 5.2). Some regions with homogeneous intensity on the moving objects can also be segmented. We report the results on the Val sets after the second refinement in Table 5.2.

	2-Frame-V1	
Option	**Blender-MP**	**Blender-Complex**
Without Threshold Strategy	0.35425	0.27710
Threshold Strategy	0.35472	-
Postprocessing	0.35483	0.27925
Merge	0.36672	0.32049
Morphology	0.59957	0.50963
CascadePSP Refinement	0.77056	0.64693

Table 5.2: Performance of 2-Frame-V1 on the motion segmentation task on the Val Set of Blender-MP and Blender-Complex dataset with Thresholding Strategy settings found in 5.2, postprocessing percentile p = 0.1 found in 5.4, and refinement settings found in 5.4. We measure the performance using mean IoU metrics. Attention initialization uses $\theta = 0.01$ found in 5.2. A dash sign indicates the corresponding option is not used for the motion segmentation task.

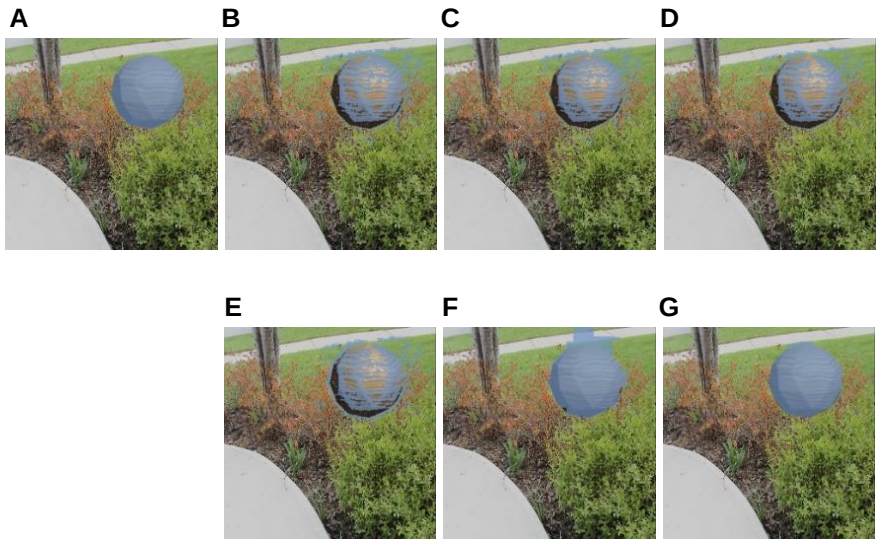

Figure 5.2: Comparison of motion segmentation results after options in Table 5.2 are applied in order. The options include Without Thresholding Strategy for Attention Initialization (See 5.2), Thresholding Strategy (See 5.2), Postprocessing (See 5.4), Merge of Masks (See 5.4), and 2 Stages of Refinement (See 5.4). 1st row: GT Segmentation (A), Without Thresholding Strategy (B), Thresholding Strategy (C), Postprocessing (D). 2nd row: Merge (E), Morphological Refinement (F), and Refinement with CascadePSP (G). Localization errors reduced slightly after applying Thresholding Strategy (C) and Postprocessing (D). Segmentation becomes denser after Merge of Masks (E). 'Holes' on the mask are closed after Morphological Refinement (F). Shape of the moving object is more accurate after Refinement with CascadePSP (G). Some regions with homogeneous intensity can be segmented after refinements.

We now compare 2-Frame-V1 model with some recent models on the motion segmentation task. We report the results for 2-Frame-V1, Bideau et al. (2016) [2], Dave et al. (2019) [6], Yang et al. (2021) [83], and Neoral et al. (2021) [91] on each sequence of Blender-MP and Blender-Complex dataset in Table 5.3. For models that also support instance motion segmentation ([6] [83] [91]), we compare with the results from the corresponding binary motion segmentation masks. [6] also utilizes the motion segmentation to develop a tracker to track the objects. For [6], we compare with the results from the motion segmentation model, instead of the tracker.

Blender-MP Val Sequence		IoU				
		2-Frame-V1	Bideau et al. (2016) [2]	Dave et al. (2019) [6]	Yang et al. (2021) [83]	Neoral et al. (2021) [91]
0	Wooden Sphere	0.7767	0.2085	0.1494	0.9526	0.8740
1	Small Stony Cube 1	0.0428	0.1789	0.0000	0.5901	0.4036
2	Leafy Cylinder	0.9526	0.0918	0.9679	0.8927	0.9223
3	Rock	0.6520	0.1304	0.9377	0.8495	0.3834
4	Grassy Ring	0.9123	0.5175	0.6463	0.5176	0.6402
5	Leafy Rectangle	0.7742	0.7245	0.7637	0.8445	0.0000
6	Grassy Cone	0.8128	0.6804	0.9368	0.8415	0.9227
7	Grassy Cylinder	0.8774	0.7865	0.9658	0.9128	0.9706
8	Wooden Polyhedron	0.8711	0.2299	0.9576	0.9052	0.9398
9	Gray Diamond	0.6953	0.0296	0.1843	0.8373	0.9446
10	Grassy Rectangle 1	0.9605	0.5587	0.0000	0.0000	0.0000
11	Stony Ellipsoid 1	0.7145	0.8087	0.7504	0.8756	0.9541
12	Wooden Rectangle 1	0.8285	0.7620	0.0633	0.5716	0.0000
13	Grassy Ellipsoid	0.9837	0.1763	0.0000	0.9263	0.9630
14	Leafy Parallelepiped	0.8719	0.1578	0.1867	0.9103	0.0000
15	Wooden Rectangle 2	0.9876	0.7559	0.3530	0.6937	0.0000
16	Stony Sphere	0.7407	0.0003	0.9466	0.9128	0.9457
17	2 Gray Stars	0.7590	0.4378	0.0940	0.4183	0.0000
18	Stony Polyhedron 1	0.6837	0.0000	0.1743	0.9268	0.9611
19	Stony Ring	0.5143	0.2850	0.2444	0.6724	0.7240
	Mean IoU	**0.7706**	0.3760	0.4661	0.7526	0.5775

Blender-MP Test Sequence		IoU				
		2-Frame-V1	Bideau et al. (2016) [2]	Dave et al. (2019) [6]	Yang et al. (2021) [83]	Neoral et al. (2021) [91]
0	Large Stony Sphere	0.9862	0.1370	0.9677	0.9299	0.9803
1	Grassy Cube	0.8367	0.0601	0.0000	0.7704	0.0000
2	Large Rock	0.9796	0.1023	0.9689	0.9169	0.9600
3	Stony Cone and Sphere	0.7699	0.7458	0.0000	0.2709	0.3434
4	Stony Polyhedron 2	0.9902	0.1244	0.4982	0.9167	0.9629
5	Wooden Cube	0.6868	0.8320	0.3832	0.3797	0.3736
6	Stony Cone	0.8137	0.6417	0.9254	0.8616	0.1926
7	Small Stony Cube 2	0.6805	0.7623	0.3367	0.6343	0.7936
8	Wooden Cylinder	0.8767	0.7627	0.9703	0.7700	0.7340
9	Clay Teapot	0.9064	0.7602	0.9397	0.7924	0.9408
10	Grassy Rectangle 2	0.9704	0.8172	0.0000	0.8989	0.0000
11	Stony Ellipsoid 2	0.9871	0.5526	0.9632	0.9368	0.7647
12	Grassy Parallelepiped	0.8601	0.4546	0.0000	0.6912	0.0000
13	Reflective Ellipsoid	0.9644	0.5440	0.9333	0.9090	0.9684
14	Large Gassy Parallelepiped	0.9144	0.8479	0.0002	0.8346	0.0000
15	Wooden Star	0.9722	0.8676	0.9022	0.1182	0.8033
16	Stony Square	0.9804	0.0000	0.7973	0.7258	0.5726
17	Leafy Star and Stony Star	0.3866	0.7219	0.0000	0.4674	0.5391
18	Small Stony Sphere	0.8943	0.3903	0.9604	0.8668	0.8434

19	Stony Cube	0.9119	0.7225	0.0000	0.0000	0.0000
	Mean IoU	**0.8684**	0.5423	0.5273	0.6846	0.5386

Blender-Complex Val Sequence		**IoU**				
		2-Frame-V1	**Bideau et al. (2016) [2]**	**Dave et al. (2019) [6]**	**Yang et al. (2021) [83]**	**Neoral et al. (2021) [91]**
1	Airplane	0.2830	0.1217	0.0259	0.1288	0.1873
11	Wooden Toy Car	0.6274	0.3567	0.1799	0.4592	0.5000
19	Cow Balloon	0.8119	0.4574	0.9173	0.6407	0.8697
21	Two Butterflies	0.8654	0.7359	0.8904	0.7595	0.8892
	Mean IoU	**0.6469**	0.4179	0.5034	0.4970	0.6115

Blender-Complex Test Sequence		**IoU**				
		2-Frame-V1	**Bideau et al. (2016) [2]**	**Dave et al. (2019) [6]**	**Yang et al. (2021) [83]**	**Neoral et al. (2021) [91]**
2	Hover Car	0.5997	0.3808	0.9203	0.3487	0.9252
3	Rainbow Hot Air Balloon	0.6657	0.1068	0.7269	0.4291	0.8572
4	Drone	0.1838	0.0910	0.0576	0.1783	0.1654
5	Paper Airplane	0.3075	0.2000	0.2465	0.3675	0.2741
6	Eagle	0.4851	0.0515	0.2511	0.1858	0.0854
7	Butterfly	0.5785	0.3205	0.7230	0.5821	0.5491
8	Seagull	0.6149	0.2840	0.6370	0.4754	0.5759
9	Crab Balloon	0.5763	0.5478	0.1811	0.4508	0.5994
10	Soccer Ball	0.8162	0.5641	0.8972	0.8460	0.8797
12	Toy Airplane	0.6583	0.3274	0.6792	0.4465	0.5670
13	Volleyball	0.7843	0.5103	0.8741	0.6960	0.8672
14	Shopping Cart	0.3592	0.2741	0.5392	0.2252	0.5681
15	Sword	0.1188	0.0650	0.1460	0.0741	0.0378
16	Rusted Tire	0.7701	0.5462	0.4379	0.6986	0.3164
17	Fighter Jet	0.1264	0.0703	0.5510	0.1605	0.0387
18	Wooden Wheel	0.5437	0.2911	0.5386	0.5324	0.2856
20	Flying Saucer	0.3800	0.3294	0.4901	0.3208	0.3302
22	Soccer Ball and Volleyball	0.6194	0.6995	0.8888	0.6868	0.7672
23	Two Drones	0.5154	0.4543	0.4989	0.4569	0.5738
24	Two Hot Air Balloons	0.6719	0.3257	0.7827	0.6909	0.9357
25	Cow Balloon and Penguin Balloon	0.6734	0.4329	0.3421	0.5105	0.6466
	Mean IoU	**0.5261**	0.3273	**0.5433**	0.4459	0.5165

Table 5.3: Performance of 2-Frame-V1 on the motion segmentation task on the Val and Test set of Blender-MP and Blender-Complex dataset. 2-Frame-V1 shows the results from 2-Frame-V1 of ST-Motion-Net with Thresholding Strategy settings found in 5.2, postprocessing percentile p = 0.1 found in 5.4, and refinement settings found in 5.4. We compare with previous motion segmentation models for which we can obtain source code and benchmark results.

2-Frame-V1 achieves excellent performance on the Blender-MP Val and Test set. Its mean IoU scores surpass all the methods we tested. For the Blender-Complex Val set, the performance of 2-Frame-V1 also surpasses all the methods we tested. The performance on the Test set of Blender-Complex is very close to Dave et al. (2019) [6] and surpasses all other models we tested.

5.5.3 Qualitative Results

We now examine some qualitative results from each method on Blender-MP and Blender-Complex dataset. Figure 5.3 shows examples of motion segmentation from each method on the Test set of Blender-MP and Blender-Complex dataset.

2-Frame-V1 produces accurate motion segmentation for the examples. It can handle occlusions (e.g. Soccer Ball and Volleyball example) and segment novel objects which are not present in the training data. It can also segment moving objects with illumination changes. For small / thin objects (e.g. sword) or objects with very slow / fast motion (e.g. fighter jet), it may miss part of the object. The motion patterns are difficult to detect and localize. Fine details of objects (e.g. inside of a wheel) may be over-segmented or segmented to some extent. Some noise may still remain around moving objects (e.g. cube) after the refinements.

[2] may under-segment or miss moving objects (e.g. wheel) and segment parts of the background. Fine details of moving objects are missed (e.g. inside of a wheel). [6] may under-segment moving objects (e.g. sphere) or miss novel objects and segment static objects (e.g. car). Fine details may be over-segmented or missed (e.g. drones). [83] may under-segment or over-segment objects (e.g. eagle); segment static objects (e.g. car); and miss novel objects. Fine details may be over-segmented or missed (e.g. eagle). [91] may under-segment moving objects (e.g. drone); miss novel objects; and segment static objects (e.g. traffic lights). Fine details (e.g. inside of a wheel) may be over-segmented or missed.

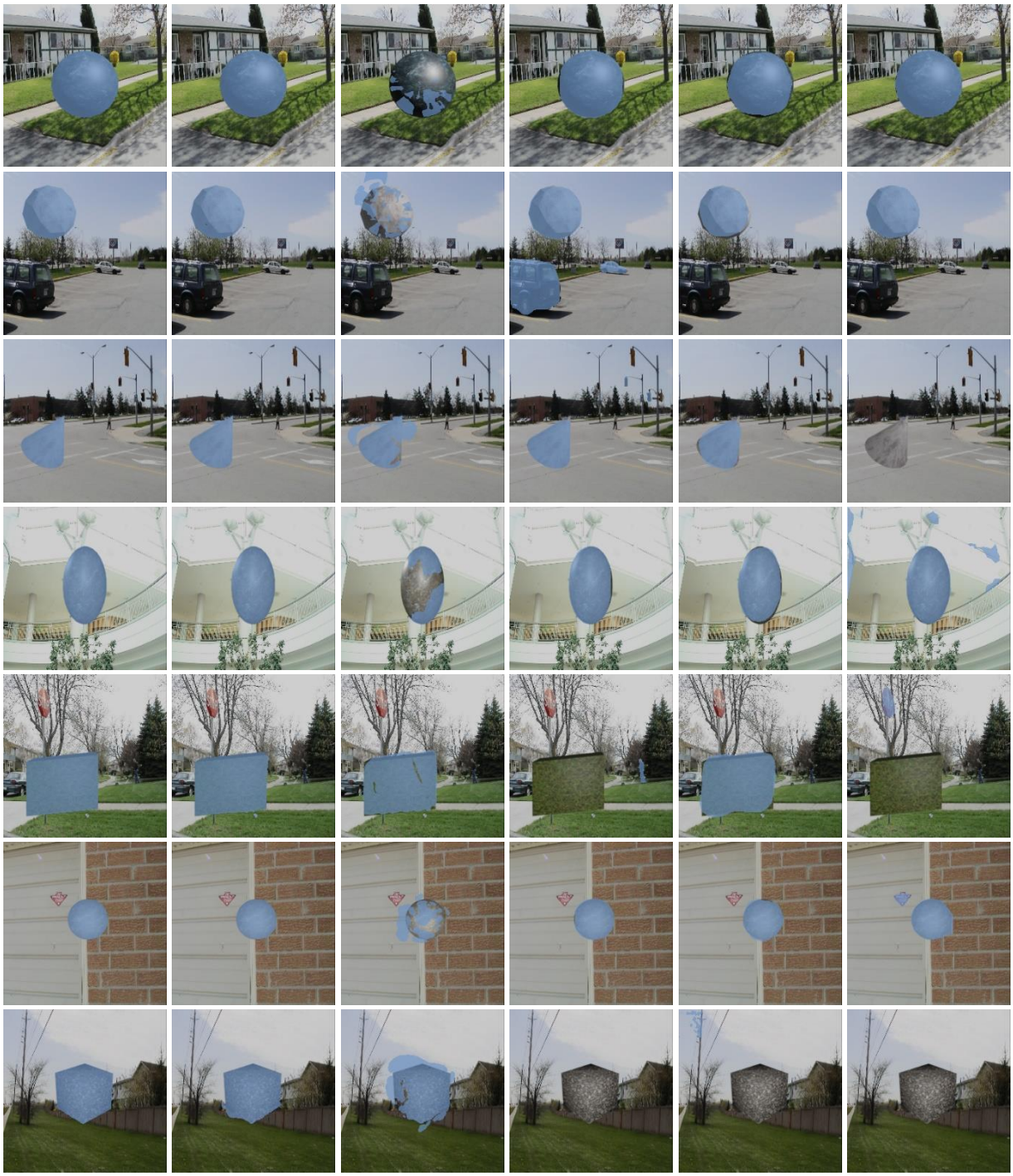

Figure 5.3: GT Motion Segmentation and Results from 2-Frame-V1, [2], [6], [83], and [91] for examples from Blender-MP and Blender-Complex Test set. Left-Right: Groundtruth, 2-Frame-V1 Result, Results from [2], [6], [83], and [91]. The first 8 examples are from the Blender-MP dataset and the last 10 examples are from Blender-Complex dataset. 2-Frame-V1 can segment novel objects (e.g. geometric objects from Blender-MP). [6], [83], [91] may miss novel objects that are not present in their training data.

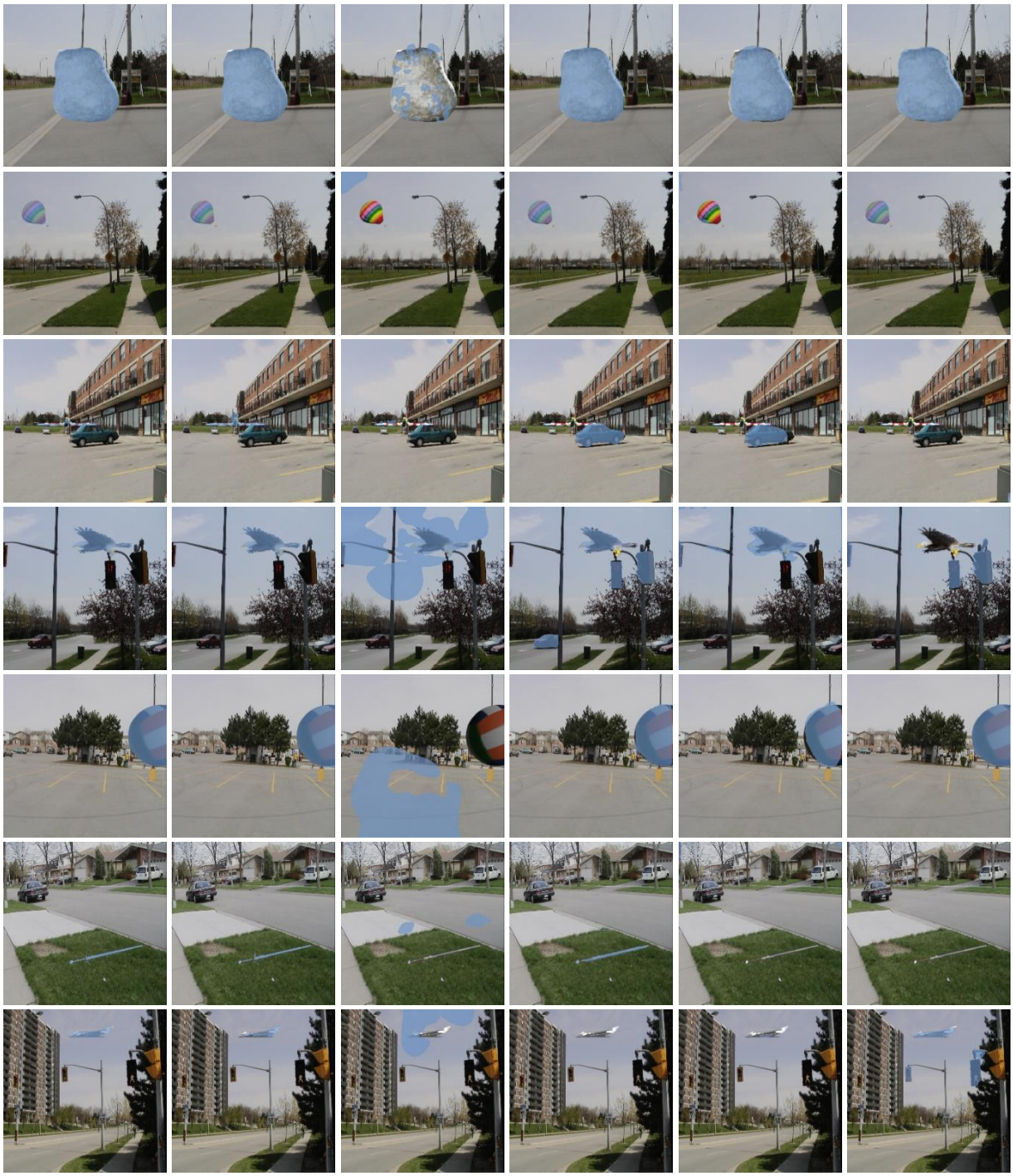

Figure 5.3 (continued)

[6], [83], [91] may incorrectly segment static objects (e.g. parked cars, traffic lights). [2] may segment large regions of the background and greatly under-segment the moving objects. For small / thin objects (e.g. sword) or objects with very fast / slow motion (e.g. fighter jet), 2-Frame-V1 may miss part of the object. The motion patterns are difficult to detect and localize.

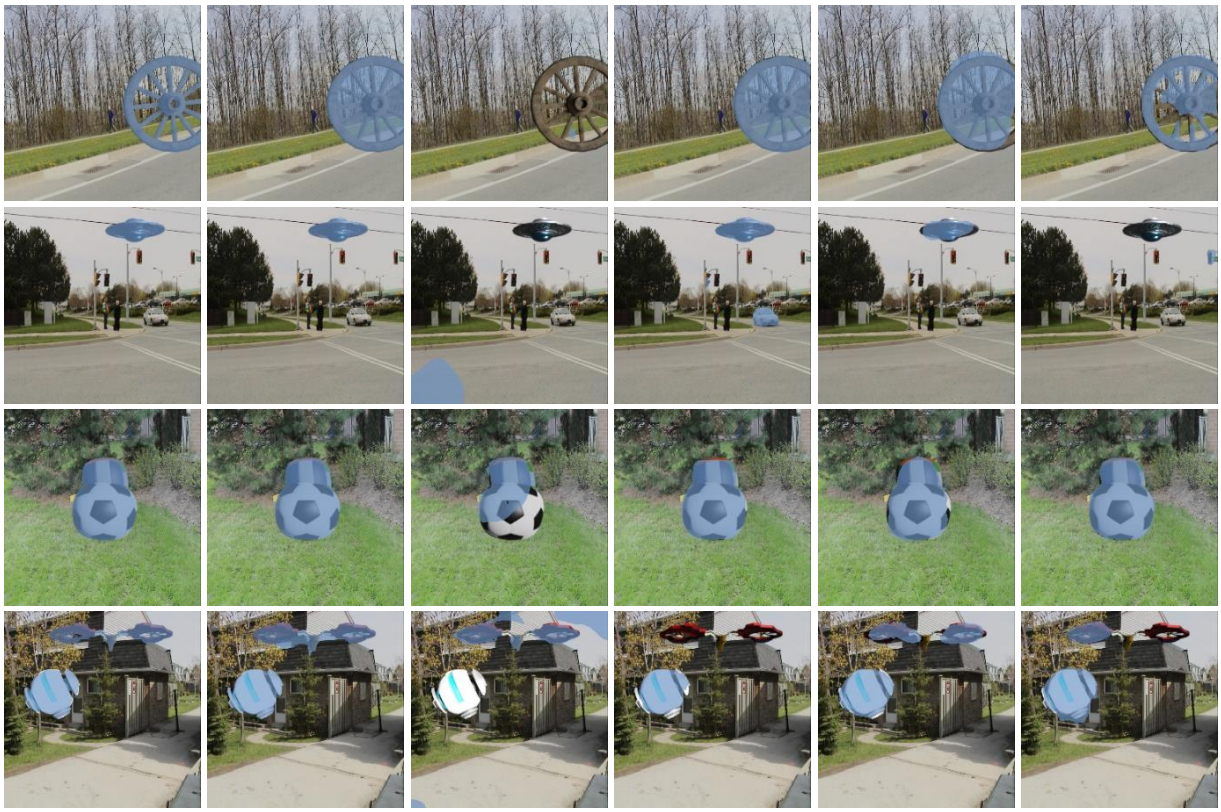

Figure 5.3 (continued)

2-Frame-V1 can handle occlusions (e.g. soccer ball and volleyball) and moving objects with illumination changes (e.g. flying saucer, 2 drones). Some noise from localization may remain around the moving objects (e.g. cube example) after the refinements. 2-Frame-V1 and the comparison models may over-segment or miss fine details of moving objects (e.g. inside of a wheel).

5.6 Conclusion and Future Works

5.6.1 Conclusion

In this chapter, we tested 2-Frame-V1 of ST-Motion-Net on the motion segmentation task using two datasets, Blender-MP and Blender-Complex. The two datasets contain sequences of geometric and real moving objects, respectively.

We compared 2-Frame-V1 with some recent motion segmentation models on both datasets. 2-Frame-V1 achieves excellent performance on the Blender-MP Val and Test set. Its performance surpasses all the methods we tested. 2-Frame-V1 also surpasses all the methods we tested on Blender-Complex Val set. On Blender-Complex Test set, the performance of 2-Frame-V1 is close to Dave et al. (2019) [6] and surpasses all other methods we tested. 2-Frame-V1 can handle occlusions and segment novel objects. It can also segment moving objects with illumination changes. Some regions with homogeneous intensity on the moving objects can also be segmented, after the refinements.

5.6.2 Future Works

2-Frame-V1 segments the moving objects, given an input subsequence of 2 frames. An Area V1 model that supports input subsequence of multiple frames can be tested to merge the collapsed attention maps of the same frame from additional previous input subsequences. This produces a denser mask for motion segmentation. 3D convolution layer can be implemented in the ST Library [3] to perform TD selection on the multi-frame V1 model.

2-Frame-V1 currently does not handle camera motion or dynamic backgrounds (e.g. snow or rain). For future work, camera motion can be estimated and filtered from the selection during attention initialization. Object proposals (e.g. SharpMask [70]) can be used to clean up the segmentation mask and keep only the pixels of moving objects. Finer details of moving objects (e.g. inside of a wheel) may be missed or over-segmented at times. More powerful refinement methods can be employed in the future to segment these details based on appearance.

Finer output feature maps from Area V1 can enable more precise selection of motion categories during attention initialization. Thus, the localization can become more accurate for the motion segmentation task. Additional neurons can also be implemented in the model to detect very slow or fast motion (See 3.6.2 for details). The new motion categories can be

selected during attention initialization to localize such motion patterns. Thus, very slow or fast objects can be segmented more accurately.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

In this thesis, we described a learnable motion processing hierarchy based on fully convolutional networks, ST-Motion-Net, a neurally-inspired model of the primate visual motion system (similar to [26] [27]). ST-Motion-Net models 4 areas (V1, MT, MST, and 7a) of the primate visual motion system and each area sends its output to the next area. Each area contains neurons that respond to translation motion. Neurons in MT are also selective for an angle between motion and speed gradient. MST and 7a contain neurons that respond to spiral motion (clockwise or counterclockwise rotation, expansion or contraction, and mixed motion). 7a contains neurons that respond to rotation, radial motion, and deformation from normal or shear strain regardless of direction. Thus, ST-Motion-Net covers the full spectrum of affine motions (i.e. image translation, rotation, divergence, and deformation). We evaluated ST-Motion-Net on the motion pattern detection task. The Selective Tuning model for visual attention is also implemented within ST-Motion-Net to localize motion patterns and segment moving objects.

We created two datasets in Blender [37] to test ST-Motion-Net on each task with sequences of geometric and real-world moving objects. The objects are moving on still backgrounds with different motion patterns, including translation, radial motion, rotation, and deformation. Shadows or illumination of the objects may change. In the second dataset, the real-world objects have more complex motion patterns, such as mixture of translation, rotation, and radial motion. Some objects may also overlap and may exit or re-enter the scene. The two datasets are called Blender-MP and Blender-Complex, respectively.

For the motion pattern detection task, ST-Motion-Net produces relatively accurate responses compared to GT responses from each area on sequences with different motion patterns. It can also detect mixed motion patterns or motion around occlusion boundaries. 2-Frame-V1 apparently produces more accurate results than 6-Frame-V1. ST-Motion-Net contains additional neural populations (compared to [26] [27]) with more speed selectivity types and selectivity for deformation motion. ST-Montion-Net supports RGB input and the output from the lower areas of ST-Motion-Net (i.e. V1 and MT) are no longer noisy, compared to Tsotsos et al. (2005) [26], as the filters in ST-Motion-Net are learned from accurate GT feature maps.

Since 3D convolution was not implemented in the ST Library [3], we tested ST-Motion-Net on the motion pattern localization and motion segmentation task using the 2-Frame-V1 model. The motion pattern localization from each area is fairly accurate, compared to the ground truth. The localization is concentrated on the pixels of the detected motion patterns. ST-Motion-Net can localize both zero-speed (e.g. regions of static background) and non-zero speed (e.g. motion of objects) motion patterns. Motion patterns of partially occluded objects and objects with illumination changes can also be localized by ST-Motion-Net.

We tested 2-Frame-V1 of ST-Motion-Net on the motion segmentation task with refinement based on appearance using a CNN refiner, CascadePSP [89]. The shapes of the moving objects are more accurate, after the refinement. There are also less ‘holes’ on the mask of moving objects. We compared 2-Frame-V1 with some recent motion segmentation models on the two datasets. 2-Frame-V1 achieves excellent performance on the Blender-MP Val and Test set. Its results surpass all the methods we tested. 2-Frame-V1 also surpass all the methods we tested on the Blender-Complex Val set. Its performance on the Blender-Complex Test set is close to Dave et al. (2019) [6] and surpasses all other methods we tested. 2-Frame-V1 can handle moving objects with occlusions and novel objects. Moving objects with changes in illumination are also handled. Some regions with homogeneous intensity on the moving objects can be segmented, after the refinement based on appearance.

6.2 Future Works

There are many kinds of improvements that can be made on ST-Motion-Net. One key direction is to test ST-Motion-Net on real datasets. We now show some examples of the current model's outputs on real data.

We first examine the outputs of ST-Motion-Net on the motion pattern detection task for a real example from Middlebury Flow dataset [97]. Figure 6.1 shows a sequence from Middlebury Flow dataset containing a rotating hydrangea with camera moving to the left. We compare the GT outputs, 2-frame model outputs, and 6-frame model outputs in Figure 6.2. Close-up views of the 2-frame and 6-frame model outputs are available in Figure A.6. We also show the pixels with known optical flow values on the GT flow map for the two central frames in Figure 6.1. The GT flow map is used to prepare the GT output feature maps for each area of ST-Motion-Net. For preparation of GT feature maps (See 3.2), we set unknown flow values on the GT flow map to zero.

Figure 6.1: Sequence from Middlebury Flow dataset of a rotating hydrangea with camera moving to the left. Inputs to 2-Frame-V1 are the 2 center frames and inputs to 6-Frame-V1 are the 6 frames. The 2nd row shows the pixels with known flow values on the GT flow map for the 2 center frames. The GT flow map is used to prepare the GT output feature maps for each area of ST-Motion-Net (See 3.2).

For translation motion outputs, both the 2-frame and 6-frame model detect the local translation motion patterns on the rotating hydrangea in various directions at different speeds. The 2-frame and 6-frame model also detect the background moving to the right ($270°$), due to the leftward motion of the camera. Some error responses may appear on the GT translation feature maps at zero speed selectivity (v = 0.0), due to the unknown flow values on the GT flow map, which are set to zero. These error responses are not present on the translation motion outputs from the 2-frame and 6-frame models. There are some motions near zero speed around the upper central region of the hydrangea, which have been weakly detected by the models.

Both the 2-frame and 6-frame model successfully detect clockwise rotation on the spiral motion outputs at $\delta = 90°$. Some error responses may appear on the GT feature maps at zero speed (v = 0.0), due to unknown flow values on the GT flow map (See Figure 6.1), which are set to zero. Some responses on the GT spiral motion feature maps are very strong in all directions δ. These are caused by very high speed gradients, which make the responses reach max value during preparation of ground truth (See Equation 3.8 in 3.2). Perhaps the speed gradients can be capped at some high value and normalized to support examples with high speed gradients.

The AP outputs of the 2-frame and 6-frame models also detect the rotation motion patterns with relatively strong response. Some error responses may also appear on the GT feature maps at zero speed (v = 0.0), due to unknown flow values on the GT flow map, which are set to zero. Some responses on the GT feature maps for AP are very strong for each motion pattern. These are caused by spatial derivatives of velocity with large magnitudes. Perhaps the absolute value of the motion pattern responses computed from the spatial derivatives (See Equation 3.17 - 3.20 in 3.2) can also be capped at some high value and normalized during ground truth preparation to support examples with spatial derivatives of large magnitudes. 16-bit images or NumPy arrays can be employed to store the outputs for higher precision.

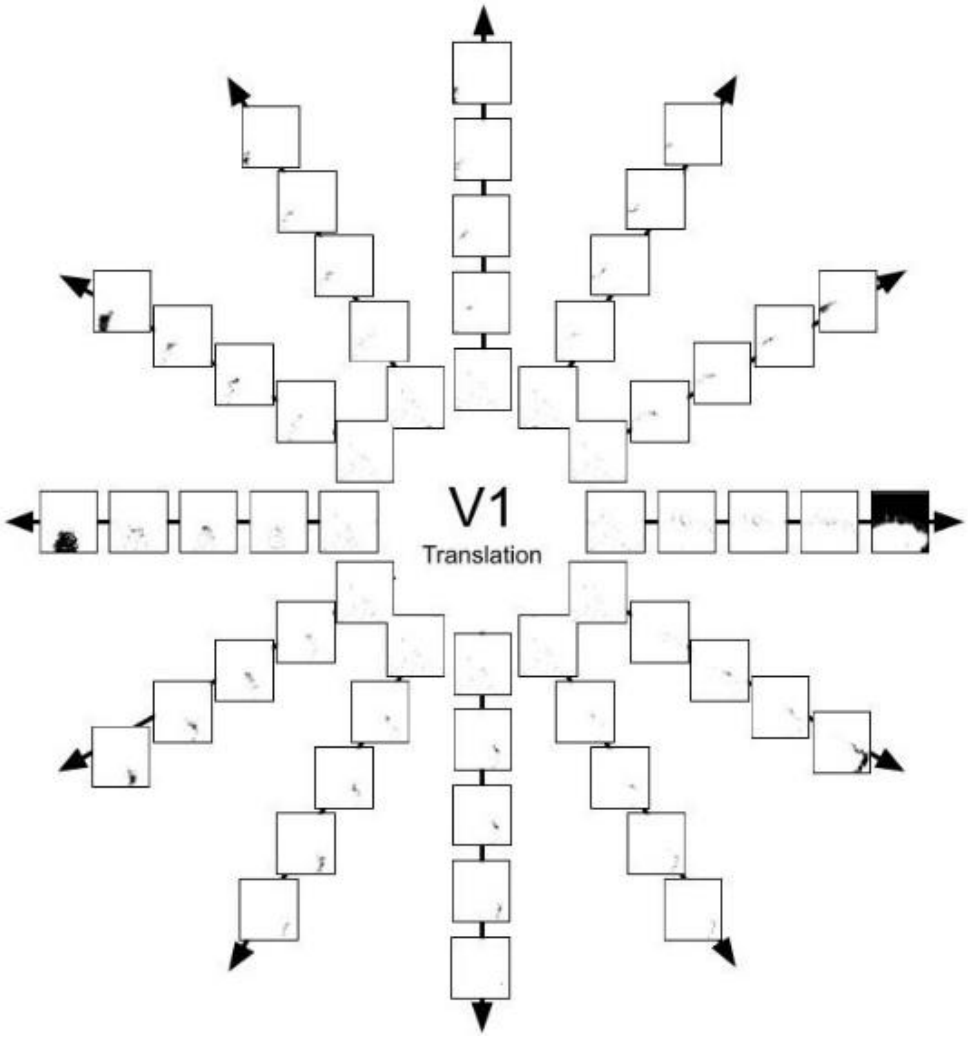

Figure 6.2: Output of area computations of ST-Motion-Net for sequence in Figure 6.1. V1 Translation shows the translation output of Area V1. This shows a close-up view of the GT output for V1 Translation. Error responses may appear on the groundtruth at zero speed (v = 0.0) selectivity due to unknown flow values on the GT flow map (See Figure 6.1).

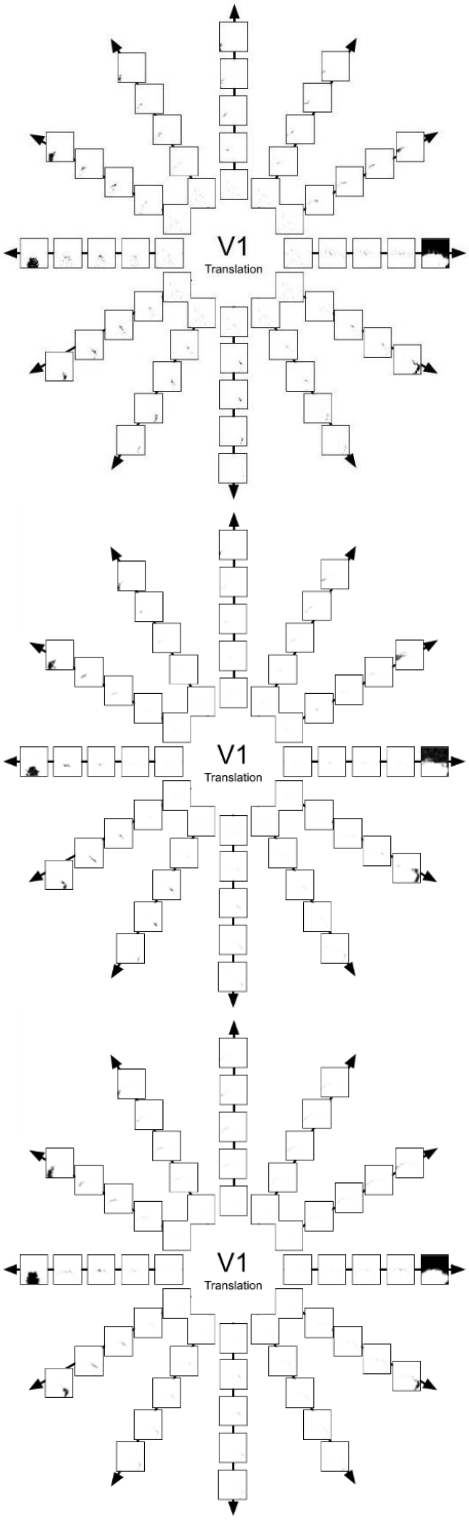

Figure 6.2: Top to bottom: GT output, 2-frame model output, and 6-frame model output. Both the 2-frame and 6-frame model detect translation motions on the rotating hydrangea. Background is moving to the right (270˚) due to leftward motion of the camera.

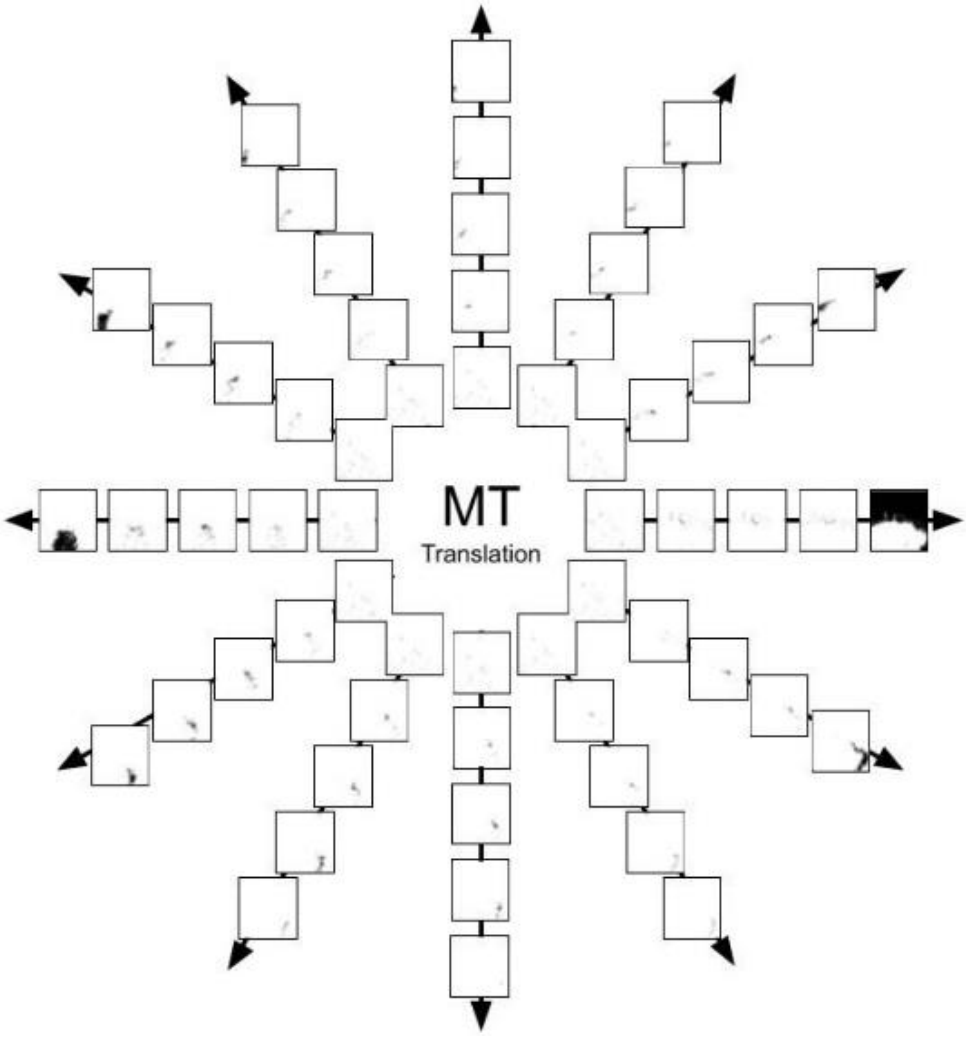

Figure 6.2 (continued): MT Translation shows the translation output of Area MT. This shows a close-up view of the GT output for MT Translation.

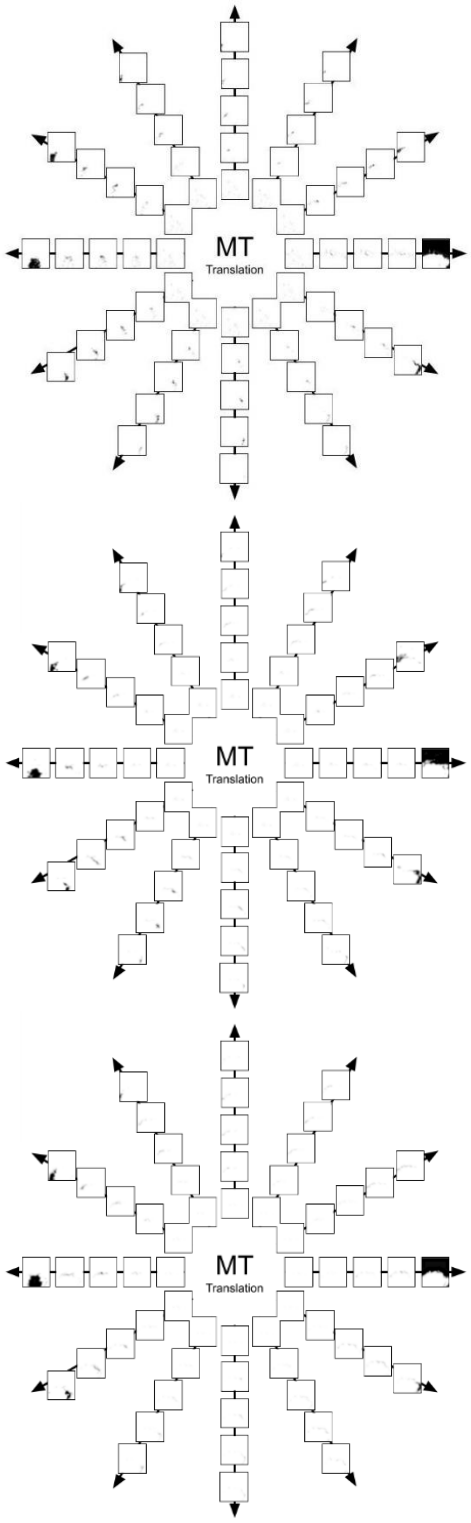

Figure 6.2 (continued)

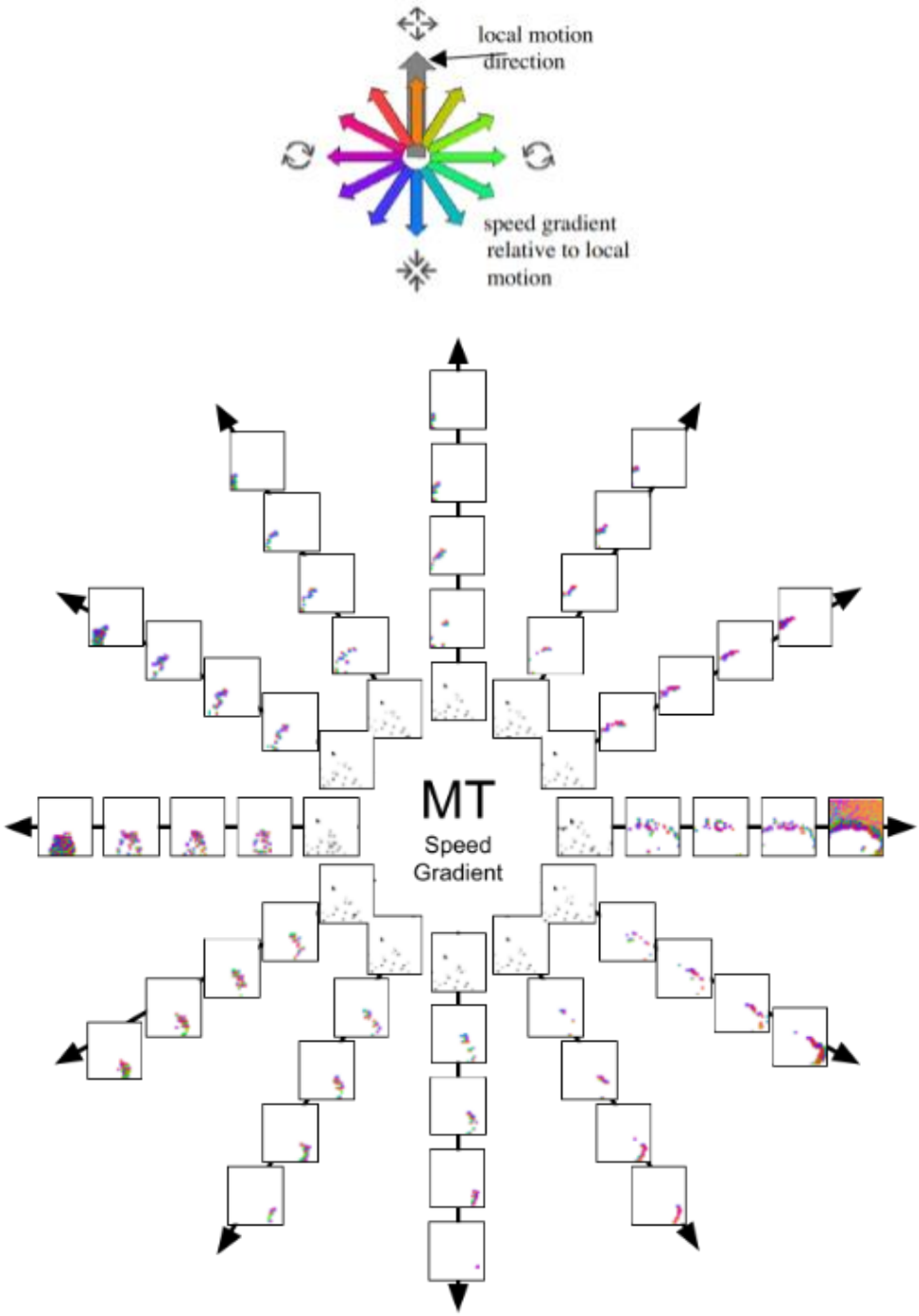

Figure 6.2 (continued): Speed gradient outputs in Area MT are shown in a summary representation of the 12 speed gradients for each local direction and speed. Each coloured dot is the maximum value across the 12 representations. This shows a close-up view of the GT output for MT Speed Gradients in the summary representation.

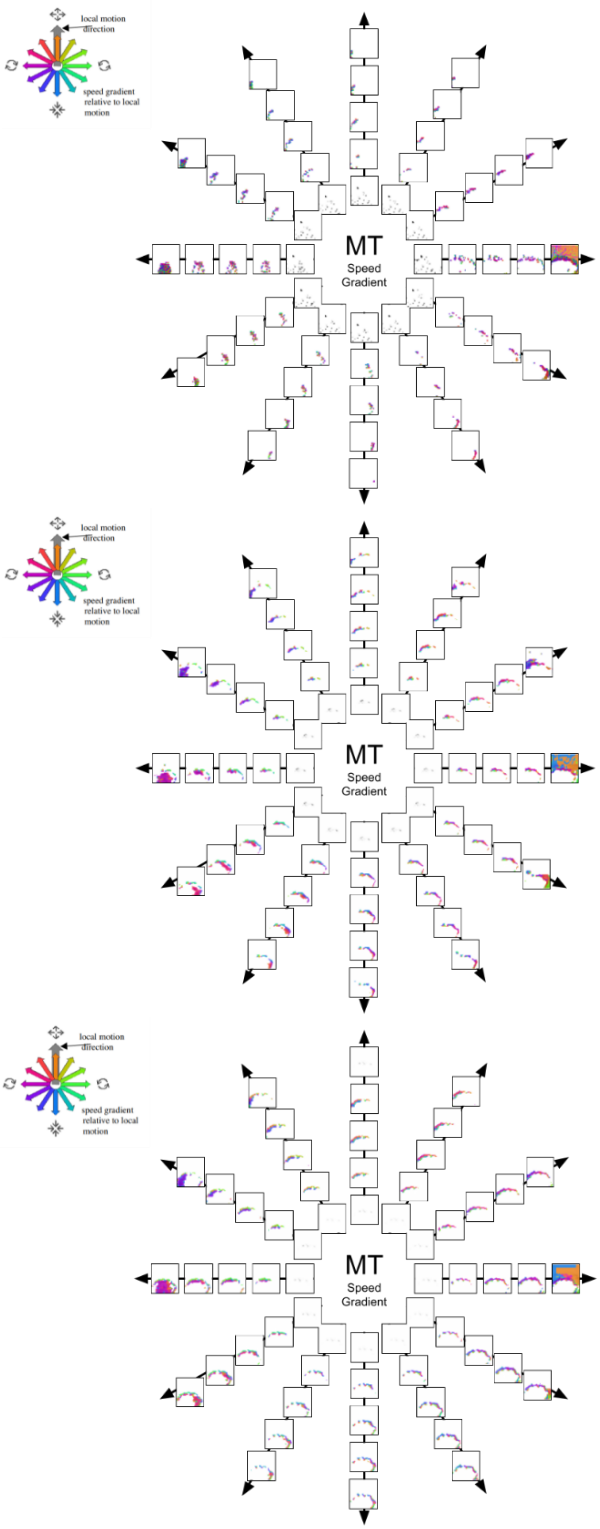

Figure 6.2 (continued): Clockwise rotation of the hydrangea is detected by both the 2-Frame and 6-Frame models (colour is roughly purple).

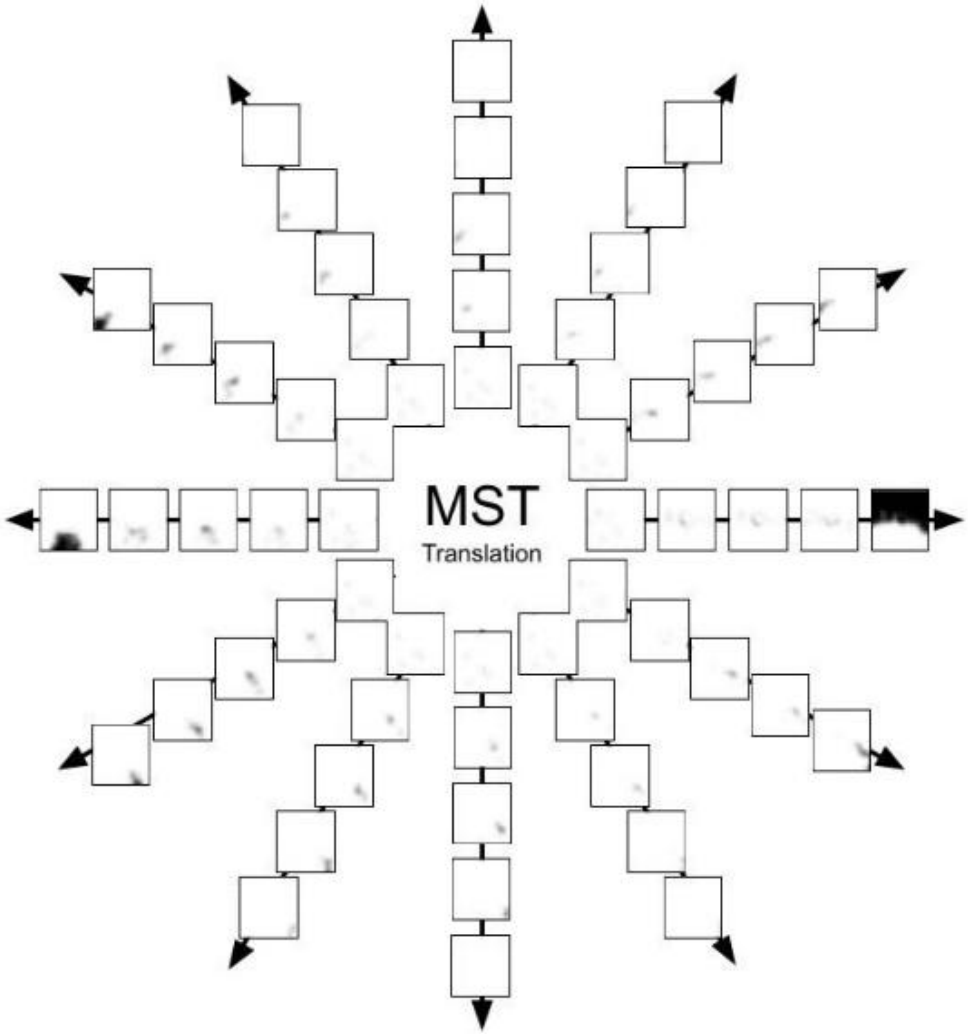

Figure 6.2 (continued): MST Translation shows the translation output in Area MST. This shows a close-up view of the GT output for MST Translation.

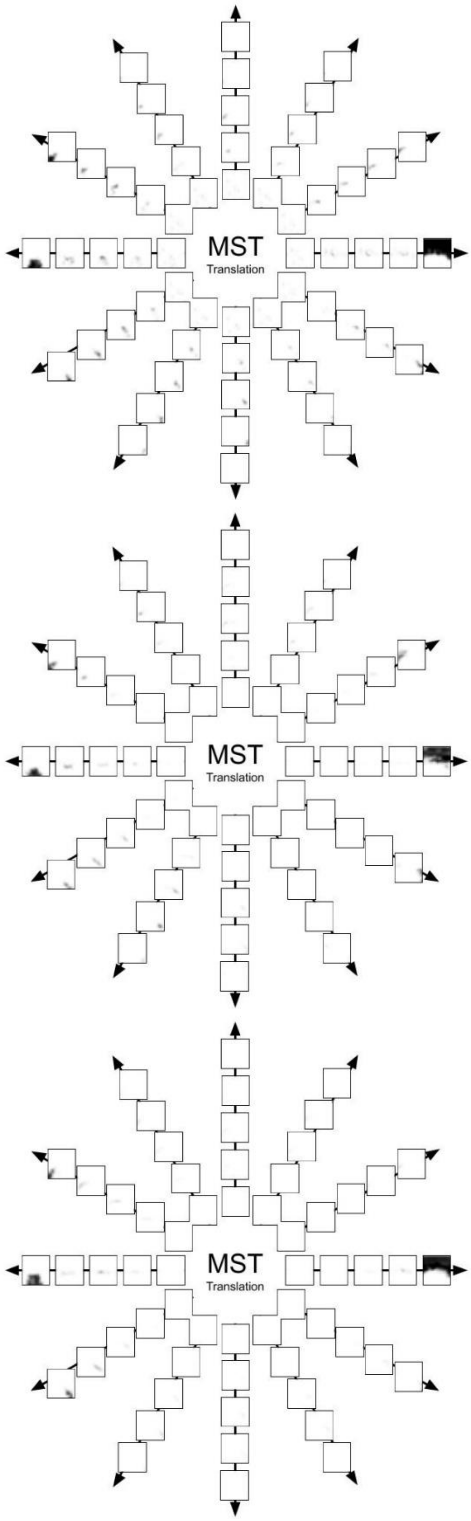

Figure 6.2 (continued)

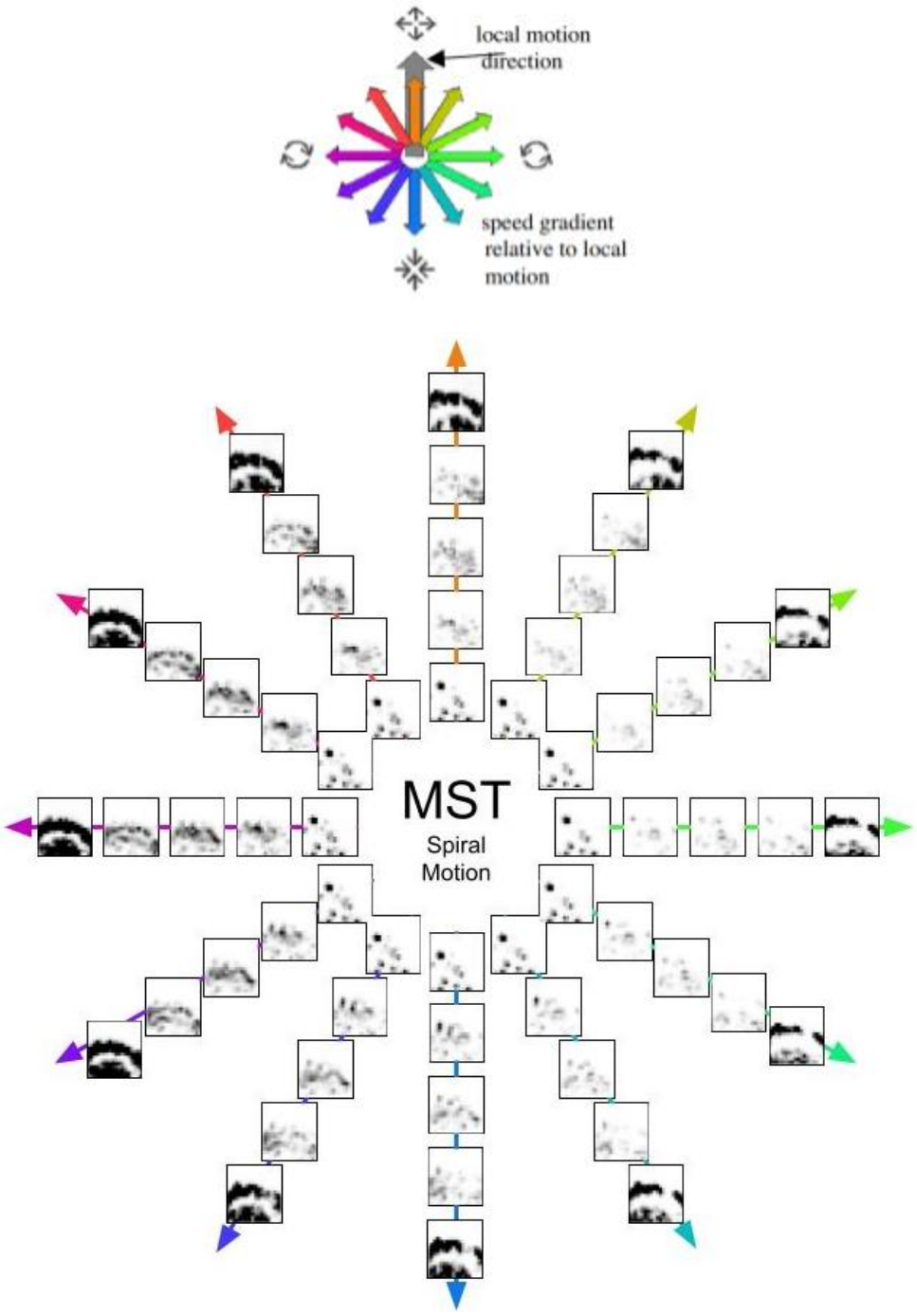

Figure 6.2 (continued): MST Spiral Motion shows the generalized spiral output in Area MST. This shows a close-up view of the GT output for MST Spiral Motion. Some responses on the groundtruth are strong in all directions of δ due to very high speed gradients which make the responses reach max value during preparation of groundtruth.

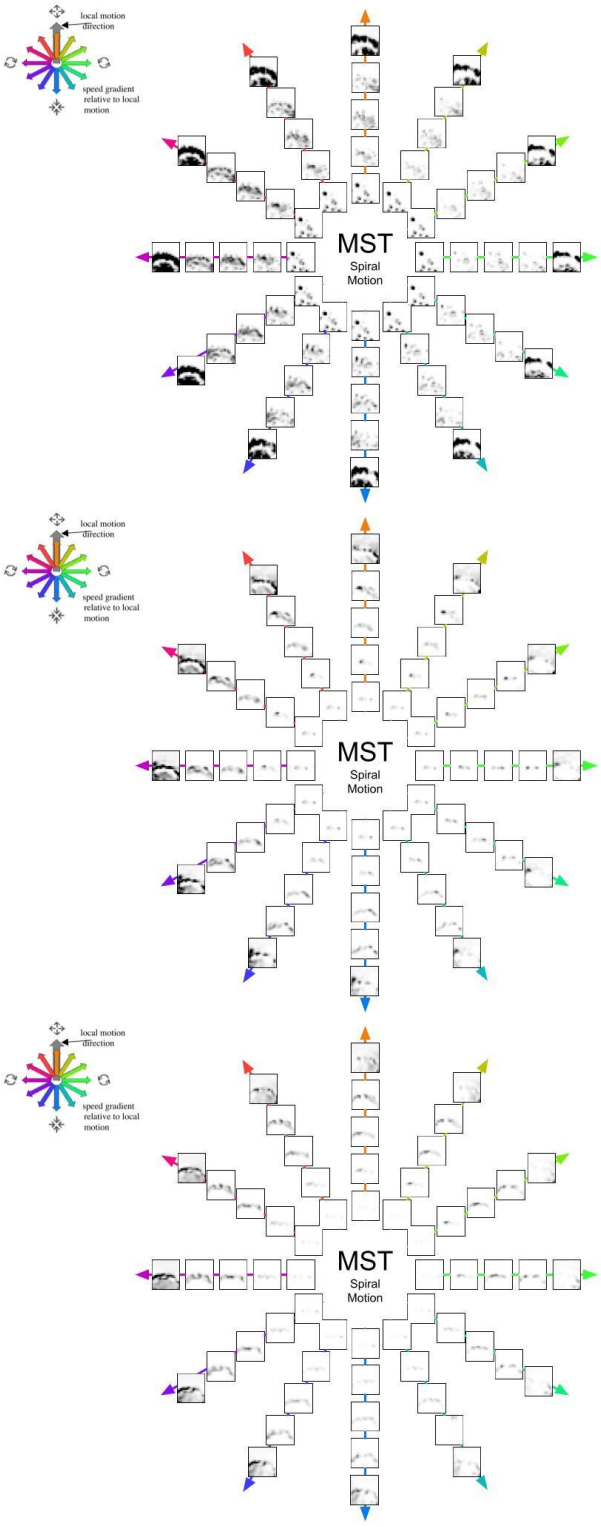

Figure 6.2 (continued): Clockwise rotation is detected at δ=90˚ by both the 2-Frame and 6-Frame models.

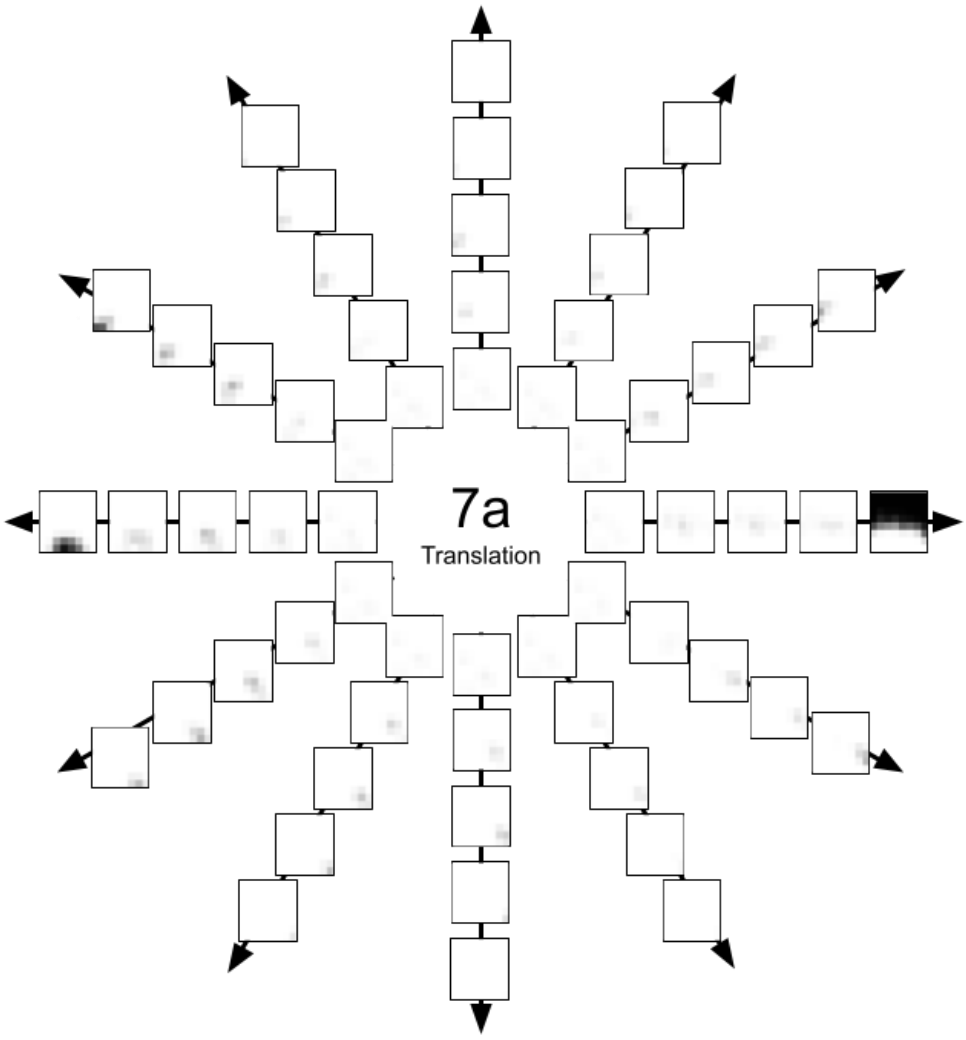

Figure 6.2 (continued): 7a Translation shows the translation output in Area 7a. This shows a close-up view of the GT output for 7a Translation.

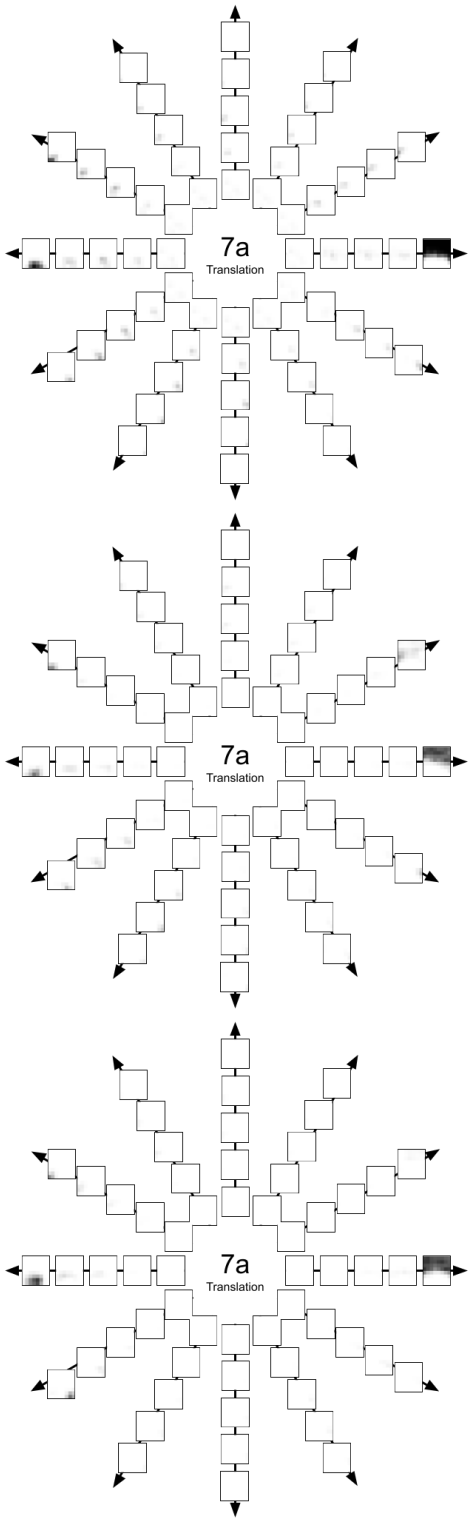

Figure 6.2 (continued)

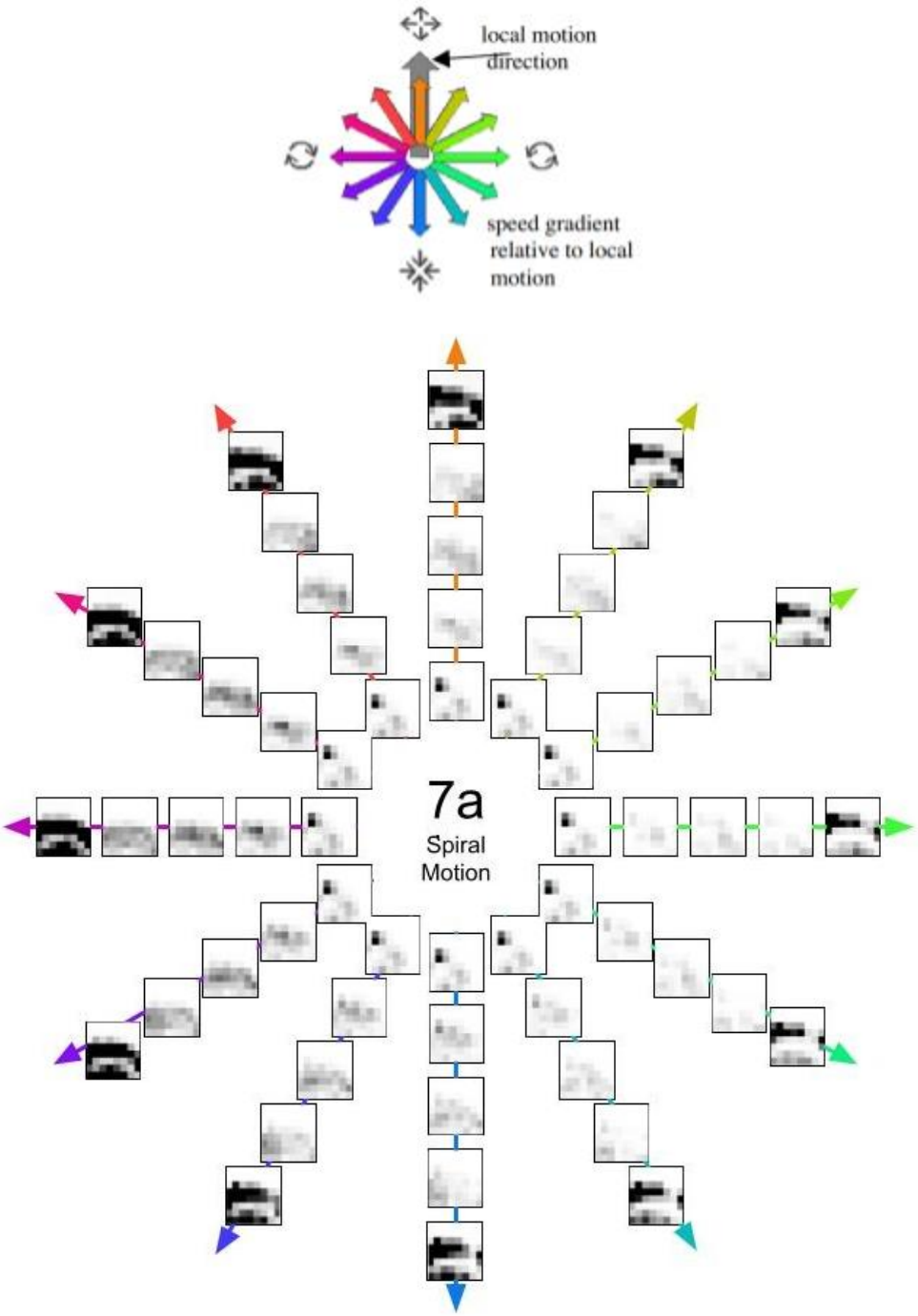

Figure 6.2 (continued): 7a Spiral Motion shows the generalized spiral output in Area 7a. This shows a close-up view of the GT output for 7a Spiral Motion. Some responses on the groundtruth are also very strong in all directions, due to high speed gradients.

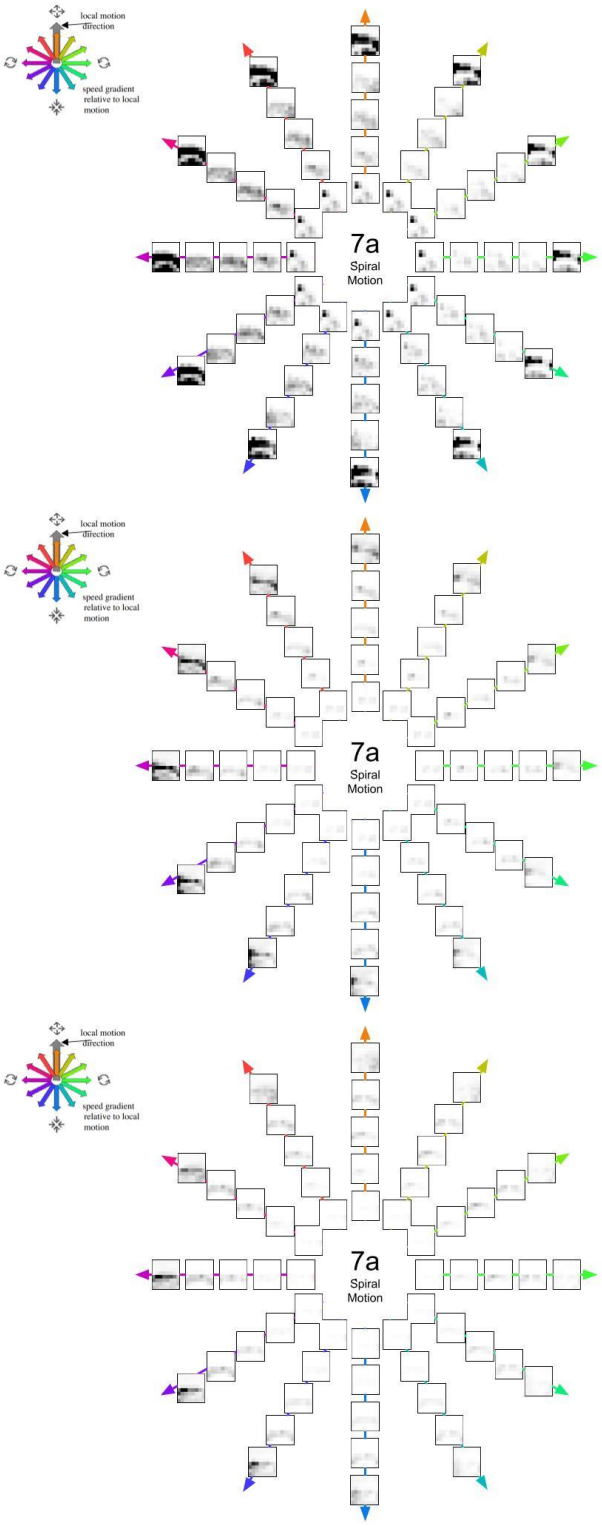

Figure 6.2 (continued): Clockwise rotation is detected at δ=90˚ by both the 2-Frame and 6-Frame models.

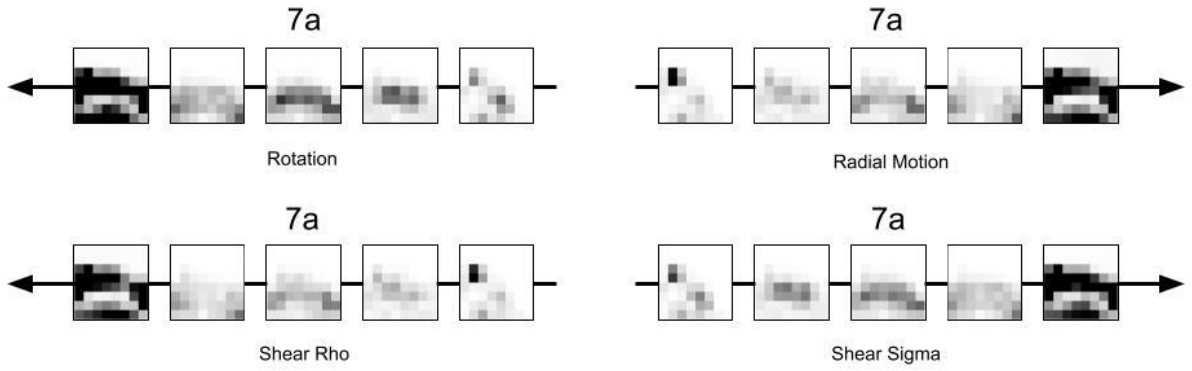

Figure 6.2 (continued): Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the GT outputs for AP. Some responses on the groundtruth are very strong for each motion pattern. These are caused by spatial derivatives of velocity with large magnitudes.

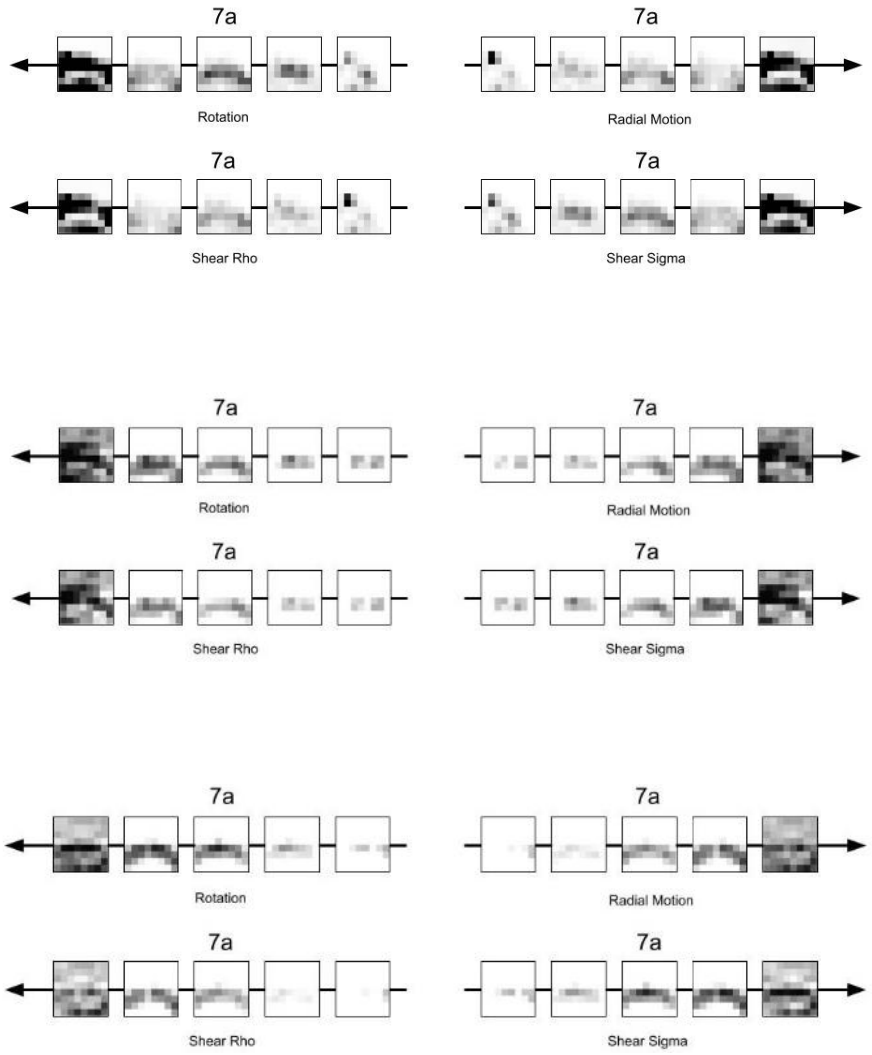

Figure 6.2 (continued): Rotation is detected with relatively strong response by both the 2-Frame and 6-Frame models.

Next, we examine the motion pattern localization from each area of ST-Motion-Net for the example in Figure 6.1. We use the hyperparameter settings tuned on the Val set of Blender-Complex dataset (See 4.3 and 4.5). Figure 6.3 shows the motion pattern localization from the 2-frame model for the example in Figure 6.1.

For the example in Figure 6.1, the GT localization masks prepared with the method in 4.6.1 are less accurate, as the GT flowmap contains pixels with unknown flow values, which are set to zero. We notice there are some pixels with known flow values at the upper central region of the hydrangea, which have zero-speed (v = 0.0) motion patterns. The remaining pixels with known flow values all have non-zero speed (v > 0.0) motion patterns. The pixels with unknown flow values apparently all have non-zero speed motion patterns as ground truth. Based on these observations, we generate more accurate GT localization masks for zero-speed and non-zero speed motion patterns, given the mask of pixels with known flow values (See Figure 6.1) and the GT localization masks prepared using 4.6.1. We show the improved GT localization masks in Figure 6.3.

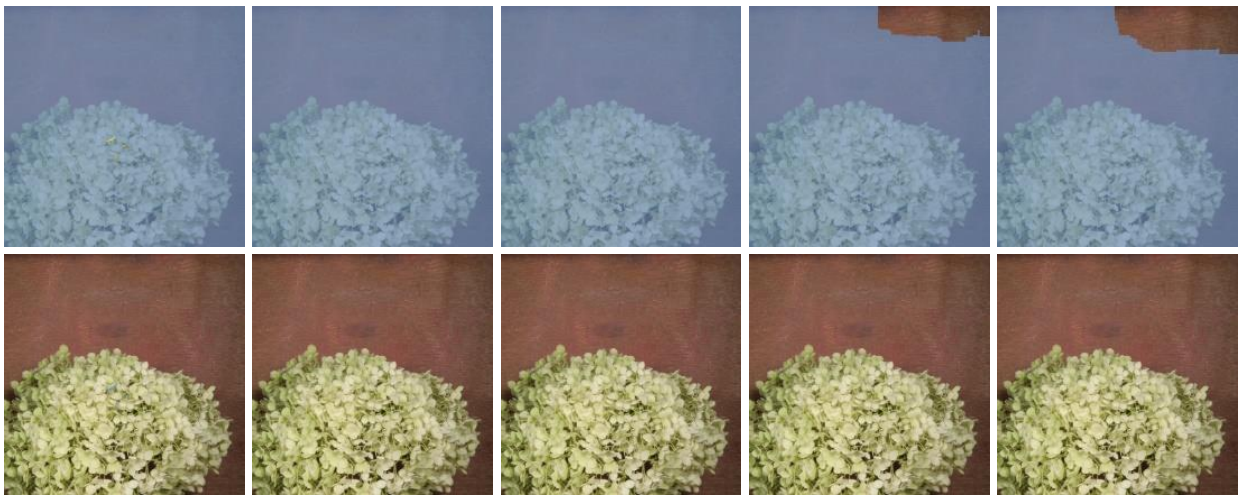

Figure 6.3: GT Motion Pattern Localization and Localization from top areas, 2-Frame-V1, MT, MST, and 7a for the Hydrangea example in Figure 6.1. (Continued on the following page.)

Figure 6.3 (continued): Left-Right: Groundtruth, 2-Frame-V1 Localization, MT Localization, MST Localization, 7a Localization. 1st Row: Localization of non-zero speed (v > 0.0) motion patterns. Most regions with non-zero speed motion patterns are localized from each area. 2nd Row: Localization of zero speed (v = 0.0) motion patterns. Localization misses pixels with zero speed motion patterns (See upper central region of the hydrangea on groundtruth), as the motion regions are very small and difficult to detect. Localization errors may increase in higher areas, as the output feature maps become coarser.

The motion pattern localization from each area is fairly accurate, compared to the GT localization masks. For non-zero speed (v > 0.0) motion patterns, most regions with non-zero speed motion patterns are localized from each area of ST-Motion-Net. For zero-speed (v = 0.0) motion patterns, the pixels with zero-speed motion patterns are missed, as the motion regions are very small and difficult to detect. Nonetheless, the pixels with non-zero speed motion patterns are not localized from each area of ST-Motion-Net. Localization errors may increase in higher areas, as the output feature maps become coarser. Finer output feature maps can lead to more precise selection of motion categories during attention initialization. Thus, the localization can become more accurate.

Next, we examine the outputs from 2-Frame-V1 on the motion segmentation task for some real examples from the Middlebury Flow dataset [97] and the KITTI dataset [92]. For examples from Middlebury dataset, we annotate the examples using CVAT (Computer Vision Annotation Tool) [120]. CVAT is an image and video annotation tool which can be used to create labelled data for computer vision applications. For examples from KITTI, we annotate the examples, based on the object masks in the flow 2015 dataset [92]. We use hyperparameter settings tuned on the Val set of Blender-Complex dataset (See 5.2 and 5.4) for the motion segmentation task. We show each example and the motion segmentation from 2-Frame-V1 in Figure 6.4.

For motion segmentation, the outputs from 2-Frame-V1 on the real examples are less accurate compared to the ground truth. The localization from 2-Frame-V1 includes camera

motion (e.g. Hydrangea and Walking examples), which also appear on the motion segmentation. Perhaps the camera motion can be estimated and filtered from the selections, during attention initialization. Then the detected motion patterns of independently moving objects can be localized and utilized for motion segmentation.

2-Frame-V1 may also detect noise (e.g. White Car example), changes in lighting or shadows, and other moving things (e.g. rain, snow, water) on the background. These motion patterns are also localized and included in the motion segmentation. Perhaps object proposals (e.g. SharpMask [70]) can be employed to filter out these motion patterns from the mask so that motion patterns of objects remain.

The localization from 2-Frame-V1 may miss motion patterns in large regions with homogeneous intensity (e.g. White Van and White Car examples), since such motion patterns are difficult to detect. Such region is also missed on the refined motion segmentation if most pixels in the region are missed on the localization. Perhaps the morphological filters can be tuned on real datasets to partially fill these missing regions, based on the pixels around the motion boundary and some pixels on the moving objects. More powerful appearance-based models can then be used to refine the mask.

The localization from 2-Frame-V1 may also miss some very slow or fast motion patterns (e.g. region around the left hand in Beanbags example). These regions may be missed on the refined motion segmentation. Perhaps additional neurons can be implemented in the model to detect very slow or fast motion patterns. Very fast or slow motion patterns can then be localized more accurately. Motion segmentation performance will also improve.

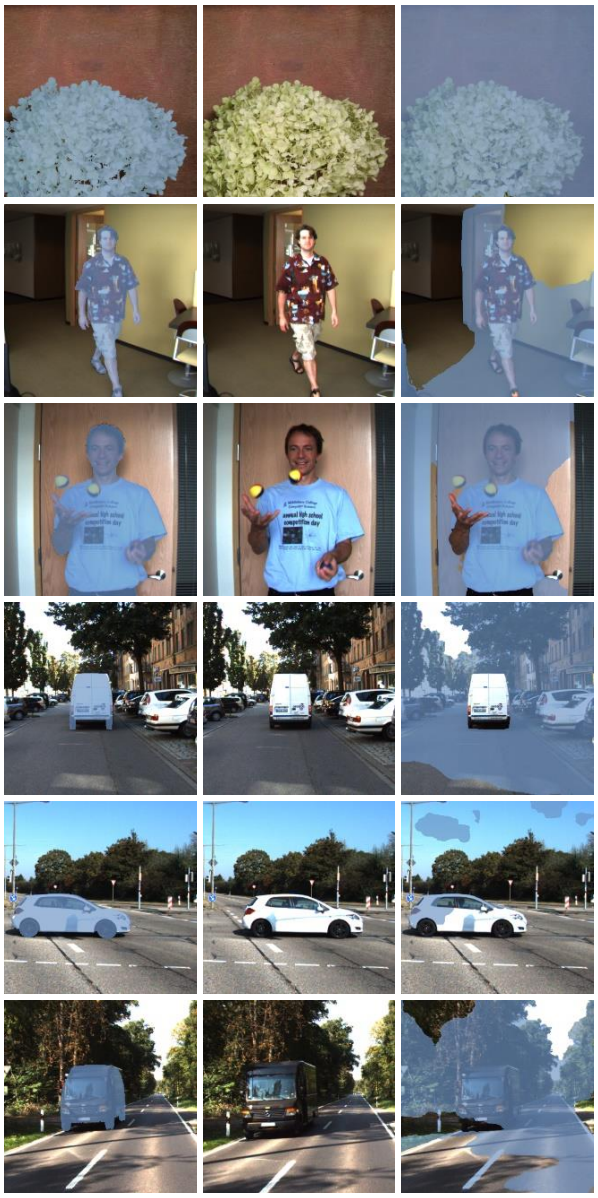

Figure 6.4: GT Motion Segmentation and Results from 2-Frame-V1 for examples from Middlebury Flow [97] and KITTI [92] dataset. Left-Right: Groundtruth for 1st Frame, 2nd Frame, 2-Frame-V1 Result. The first 3 examples are from Middlebury Flow dataset and the last 3 examples are from KITTI dataset. Localization from 2-Frame-V1 may include camera motion, which also appear on the motion segmentation (e.g. Hydrangea example on 1st row and Walking example on 2nd row). 2-Frame-V1 may detect noise, changes in lighting or shadows, and other moving things (e.g. rain, snow, water) on the background. These motion patterns are also localized and included in the motion segmentation (e.g. White Car example on 5th row). (Continued on the following page.)

Figure 6.4 (continued): Localization from 2-Frame-V1 may miss large regions with homogeneous intensity, since such motion patterns are difficult to detect. Such regions may also be missed on the refined motion segmentation (e.g. White Van example on 4th row and White Car example on 5th row). Localization from 2-Frame-V1 may also miss some very slow or fast motion patterns. These regions may be missed on the refined motion segmentation (e.g. region around the left hand on Beanbags example on 3rd row).

Bibliography

[1] S. A. Beardsley and L. M. Vaina, “Computational modelling of optic flow selectivity in MSTd neurons,” *Network: Computation in Neural Systems*, vol. 9, no. 4, p. 467, 1998.

[2] P. Bideau and E. Learned-Miller, “It’s moving! a probabilistic model for causal motion segmentation in moving camera videos,” in *European Conference on Computer Vision*, 2016, pp. 433–449.

[3] M. Biparva and J. Tsotsos, “STNet: selective tuning of convolutional networks for object localization,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2715–2723.

[4] M. Biparva and J. Tsotsos, “Selective Segmentation Networks Using Top-Down Attention,” *arXiv preprint arXiv:2002.01125*, 2020.

[5] K. Daniilidis, “Attentive visual motion processing: computations in the log-polar plane,” in *Theoretical Foundations of Computer Vision*, Springer, 1996, pp. 1–20.

[6] A. Dave, P. Tokmakov, and D. Ramanan, “Towards segmenting anything that moves,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, 2009, pp. 248–255.

[8] M. A. Giese and T. Poggio, “Neural mechanisms for the recognition of biological movements,” *Nature Reviews Neuroscience*, vol. 4, no. 3, pp. 179–192, 2003.

[9] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[10] S. Grossberg, E. Mingolla, and L. Viswanathan, “Neural dynamics of motion integration and segmentation within and across apertures,” *Vision research*, vol. 41, no. 19, pp. 2521–2553, 2001.

[11] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[12] E. Ilg, T. Saikia, M. Keuper, and T. Brox, “Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 614–630.

[13] J. Janai, F. Guney, J. Wulff, M. J. Black, and A. Geiger, “Slow flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3597–3607.

[14] S. Ji, W. Xu, M. Yang, and K. Yu, “3D convolutional neural networks for human action recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[15] J. Cartucho, S. Tukra, Y. Li, D. S. Elson, and S. Giannarou, “VisionBlender: a tool to efficiently generate computer vision datasets for robotic surgery,” *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 9, no. 4, pp. 331–338, 2021.

[16] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[17] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[18] Z.-L. Lu and G. Sperling, “The functional architecture of human visual motion perception,” *Vision research*, vol. 35, no. 19, pp. 2697–2722, 1995.

[19] O. Matan, C. J. Burges, Y. LeCun, and J. Denker, “Multi-digit recognition using a space displacement neural network,” *Advances in neural information processing systems*, vol. 4, 1991.

[20] T. S. Meese and S. J. Anderson, “Spiral mechanisms are required to account for summation of complex motion components,” *Vision Research*, vol. 42, no. 9, pp. 1073–1080, 2002.

[21] S. J. Nowlan and T. J. Sejnowski, “A selection model for motion processing in area MT of primates,” *Journal of Neuroscience*, vol. 15, no. 2, pp. 1195–1214, 1995.

[22] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[23] E. P. Simoncelli and D. J. Heeger, “A model of neuronal responses in visual area MT,” *Vision research*, vol. 38, no. 5, pp. 743–761, 1998.

[24] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[25] S. Treue and J. H. Maunsell, “Attentional modulation of visual motion processing in cortical areas MT and MST,” *Nature*, vol. 382, no. 6591, pp. 539–541, 1996.

[26] J. K. Tsotsos, Y. Liu, J. C. Martinez-Trujillo, M. Pomplun, E. Simine, and K. Zhou, “Attending to visual motion,” *Computer Vision and Image Understanding*, vol. 100, no. 1–2, pp. 3–40, 2005.

[27] J. K. Tsotsos, M. Pomplun, J. C. Martinez-Trujillo, and K. Zhou, “Attending to visual motion: Localizing and classifying affine motion patterns,” in *First Canadian Conference on Computer and Robot Vision, 2004. Proceedings.*, 2004, pp. 452–462.

[28] F. D. Vintila and J. K. Tsotsos, “Motion Estimation Using a General Purpose Neural Network Simulator for Visual Attention,” in *2007 IEEE Workshop on Applications of Computer Vision (WACV’07)*, 2007, pp. 19–19.

[29] R. S. Zemel and T. J. Sejnowski, “A Model for Encoding Multiple Object Motions and Self-Motion in Area MST of Primate Visual Cortex,” *J. Neurosci.*, vol. 18, no. 1, pp. 531–547, Jan. 1998, doi: 10.1523/JNEUROSCI.18-01-00531.1998.

[30] P. Bideau, “Motion Segmentation - Segmentation of Independently Moving Objects in Video,” University of Massachusetts Amherst, 2020. doi: 10.7275/W9VX-9171.

[31] N. A. Mandellos, I. Keramitsoglou, and C. T. Kiranoudis, “A background subtraction algorithm for detecting and tracking vehicles,” *Expert Systems with Applications*, vol. 38, no. 3, pp. 1619–1631, 2011.

[32] M. Piccardi, “Background subtraction techniques: a review,” in *2004 IEEE international conference on systems, man and cybernetics (IEEE Cat. No. 04CH37583)*, 2004, vol. 4, pp. 3099–3104.

[33] L. Zappella, X. Llado, J. Salvi, L. Zappella, X. Llado, and J. Salvi, *New Trends in Motion Segmentation*. IntechOpen, 2009. doi: 10.5772/7551.

[34] “Morphology.” https://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm (accessed Mar. 02, 2023).

[35] Y. Zhang, X. Wang, and B. Qu, “Three-frame difference algorithm research based on mathematical morphology,” *Procedia Engineering*, vol. 29, pp. 2705–2709, 2012.

[36] S. S. Sengar and S. Mukhopadhyay, “A novel method for moving object detection based on block based frame differencing,” in *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, 2016, pp. 467–472.

[37] Blender Online Community, “Blender - a 3D modelling and rendering package.” Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[38] J. Huang, W. Zou, J. Zhu, and Z. Zhu, “Optical flow based real-time moving object detection in unconstrained scenes,” *arXiv preprint arXiv:1807.04890*, 2018.

[39] X. Li and C. Xu, “Moving object detection in dynamic scenes based on optical flow and superpixels,” in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2015, pp. 84–89.

[40] J. Hariyono, V.-D. Hoang, and K.-H. Jo, “Moving object localization using optical flow for pedestrian detection from a moving vehicle,” *The Scientific World Journal*, vol. 2014, 2014.

[41] W. Hu, T. Tan, L. Wang, and S. Maybank, “A survey on visual surveillance of object motion and behaviors,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 3, pp. 334–352, 2004.

[42] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[43] W. Shi *et al.*, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.

[44] C. Ledig *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[45] B. Heo, K. Yun, and J. Y. Choi, “Appearance and motion based deep learning architecture for moving object detection in moving camera,” in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 1827–1831.

[46] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[47] T. Brox and J. Malik, “Object segmentation by long term analysis of point trajectories,” in *European conference on computer vision*, 2010, pp. 282–295.

[48] P. Ochs, J. Malik, and T. Brox, “Segmentation of moving objects by long term video analysis,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 6, pp. 1187–1200, 2013.

[49] Y. Chen, J. Wang, B. Zhu, M. Tang, and H. Lu, “Pixelwise deep sequence learning for moving object detection,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 9, pp. 2567–2579, 2017.

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[51] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[52] C. Szegedy *et al.*, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[53] A. Dosovitskiy *et al.*, “Flownet: Learning optical flow with convolutional networks,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[54] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “Flownet 2.0: Evolution of optical flow estimation with deep networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[55] P. Tokmakov, K. Alahari, and C. Schmid, “Learning motion patterns in videos,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3386–3394.

[56] P. Tokmakov, C. Schmid, and K. Alahari, “Learning to segment moving objects,” *International Journal of Computer Vision*, vol. 127, no. 3, pp. 282–301, 2019.

[57] P. Bideau, R. R. Menon, and E. Learned-Miller, “Moa-net: self-supervised motion segmentation,” in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[58] T.-Y. Lin *et al.*, “Microsoft coco: Common objects in context,” in *European conference on computer vision*, 2014, pp. 740–755.

[59] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 633–641.

[60] M. Everingham, S. M. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[61] Y. Wu, J. Lim, and M.-H. Yang, “Object Tracking Benchmark,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015, doi: 10.1109/TPAMI.2014.2388226.

[62] H. Fan *et al.*, “Lasot: A high-quality benchmark for large-scale single object tracking,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5374–5383.

[63] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, “Motchallenge 2015: Towards a benchmark for multi-target tracking,” *arXiv preprint arXiv:1504.01942*, 2015.

[64] M. Kristan *et al.*, “The sixth visual object tracking vot2018 challenge results,” in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[65] E. Agustsson and R. Timofte, “Ntire 2017 challenge on single image super-resolution: Dataset and study,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 126–135.

[66] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, and L. Zhang, “Ntire 2017 challenge on single image super-resolution: Methods and results,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 114–125.

[67] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2001, vol. 2, pp. 416–423.

[68] J.-B. Huang, A. Singh, and N. Ahuja, “Single image super-resolution from transformed self-exemplars,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5197–5206.

[69] N. Mayer *et al.*, “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4040–4048.

[70] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, “Learning to refine object segments,” in *European conference on computer vision*, 2016, pp. 75–91.

[71] P. Krähenbühl and V. Koltun, “Efficient inference in fully connected crfs with gaussian edge potentials,” *Advances in neural information processing systems*, vol. 24, 2011.

[72] M. Siam, H. Mahgoub, M. Zahran, S. Yogamani, M. Jagersand, and A. El-Sallab, “Modnet: Motion and appearance based moving object detection network for autonomous driving,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2859–2864.

[73] P. Bideau, R. R. Menon, and E. Learned-Miller, “Moa-net: self-supervised motion segmentation,” in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[74] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, “A naturalistic open source movie for optical flow evaluation,” in *European conference on computer vision*, 2012, pp. 611–625.

[75] P. Bideau, A. RoyChowdhury, R. R. Menon, and E. Learned-Miller, “The best of both worlds: Combining cnns and geometric constraints for hierarchical motion segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 508–517.

[76] M. Narayana, A. Hanson, and E. Learned-Miller, “Coherent motion segmentation in moving camera videos using optical flow orientations,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1577–1584.

[77] C. Xie, Y. Xiang, Z. Harchaoui, and D. Fox, “Object discovery in videos as foreground motion clustering,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9994–10003.

[78] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[79] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, “A benchmark dataset and evaluation methodology for video object segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 724–732.

[80] D. Tsai, M. Flagg, and J. Rehg, “Motion Coherent Tracking with Multi-label MRF optimization,” in *Procedings of the British Machine Vision Conference 2010*, Aberystwyth, 2010, p. 56.1-56.11. doi: 10.5244/C.24.56.

[81] Y. Yang, A. Loquercio, D. Scaramuzza, and S. Soatto, “Unsupervised moving object detection via contextual information separation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 879–888.

[82] H. Rashed, M. Ramzy, V. Vaquero, A. El Sallab, G. Sistu, and S. Yogamani, “Fusemodnet: Real-time camera and lidar based moving object detection for robust low-light autonomous driving,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[83] G. Yang and D. Ramanan, “Learning to segment rigid motions from two frames,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1266–1275.

[84] S. T. Steinmetz, O. W. Layton, N. V. Powell, and B. R. Fajen, “A Dynamic Efficient Sensory Encoding Approach to Adaptive Tuning in Neural Models of Visual Motion Processing,” *bioRxiv*, 2021.

[85] J. K. Tsotsos, *A computational perspective on visual attention*. MIT Press, 2011.

[86] D. Ganguli and E. P. Simoncelli, “Efficient sensory encoding and Bayesian inference with heterogeneous neural populations,” *Neural computation*, vol. 26, no. 10, pp. 2103–2134, 2014.

[87] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, “SLIC superpixels compared to state-of-the-art superpixel methods,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[88] C. Zach, T. Pock, and H. Bischof, “A duality based approach for realtime tv-l 1 optical flow,” in *Joint pattern recognition symposium*, 2007, pp. 214–223.

[89] H. K. Cheng, J. Chung, Y.-W. Tai, and C.-K. Tang, “Cascadepsp: Toward class-agnostic and very high-resolution segmentation via global and local refinement,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8890–8899.

[90] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, “Efficient object localization using convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 648–656.

[91] M. Neoral, J. Šochman, and J. Matas, “Monocular Arbitrary Moving Object Discovery and Segmentation,” 2021.

[92] M. Menze and A. Geiger, “Object scene flow for autonomous vehicles,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3061–3070.

[93] X. Xu, L. F. Cheong, and Z. Li, “Motion segmentation by exploiting complementary geometric models,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2859–2867.

[94] S. Qiao, L.-C. Chen, and A. Yuille, “Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 10213–10224.

[95] K. Chen *et al.*, “Hybrid task cascade for instance segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4974–4983.

[96] R. Tron and R. Vidal, “A benchmark for the comparison of 3-d motion segmentation algorithms,” in *2007 IEEE conference on computer vision and pattern recognition*, 2007, pp. 1–8.

[97] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, “A database and evaluation methodology for optical flow,” *International journal of computer vision*, vol. 92, no. 1, pp. 1–31, 2011.

[98] Sirkitree, *Balloon*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/7dab0b937e7d4a7c8d01ee6822c1a37c/embed?autostart=1

[99] Akshat, *Butterfly*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/896bcaf691ea48f3b8c52d77d13e73e6/embed?autostart=1

[100] Smeerws, *Cow-Balloon*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/d3ad59278e6545e5a896f28f9a2f7cf9/embed?autostart=1

[101] Smeerws, *Crab-Balloon*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/1d2915067249497c8e15433b834fe396/embed?autostart=1

[102] Comitre, *Drone Black Version*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/605f3721b47742ec9add89832352a6dc/embed?autostart=1

[103] Filip, *Free Eagle*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/9f25e7c83840435396225970ba47b75e/embed?autostart=1

[104] I. Friedman, *Hover car*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/703eba5d2b60405dbeaf53552331ec24/embed?autostart=1

[105] Amt16, *Low Poly UFO*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/96dc7b3bcc9444c3be17ef54bba21df4/embed?autostart=1

[106] Gloryfish, *Paper Airplane*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/36d00a94129f4556bfffbe6f191f092c/embed?autostart=1

[107] Smeerws, *Penguin-Balloon*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/4cffbf74c70a46ec9d63f33552319215/embed?autostart=1

[108] Onlymorozov, *Rust Wheel*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/355bc1bceb1e46d5a0b80585da1ea084/embed?autostart=1

[109] M. David, *Sci-Fi Recon Drone*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/ff2ede3b4ffe42d4afe7c42470c20c05/embed?autostart=1

[110] J. Johnson, *seagull v1*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/f3fa166ff6bd4ffe8133f6f6eed74d64/embed?autostart=1

[111] AdrianXY, *Shopping Cart*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/b96f896453b240ae804d0399f1faf027/embed?autostart=1

[112] Typhomnt, *Soccer Ball*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/46c91864ef384158b0078e20bdbfe3e9/embed?autostart=1

[113] MinghauLoh, *Sword*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/07463a2658e04d6ab8a42b5639a35d63/embed?autostart=1

[114] Asleshka, *Toy car*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/cb444316e5cd4b9498c98804e6daaafc/embed?autostart=1

[115] L3clarke, *Toy Plane*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/804d6814818247978f28d12efcc38cce/embed?autostart=1

[116] Yusuf, *Turkish Airlines*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/7665251b6fb24b29b30aa087a0cf3c64/embed?autostart=1

[117] KevinAz61, *Viva Balloon*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/ab7a88ee339a4a1b9c36ddbae44ae4a2/embed?autostart=1

[118] J. E. Grayson, *Volleyball*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/5b94ae099617446eb4f5ea60427ccfcf/embed?autostart=1

[119] Vyacheslav_SD, *Wooden Wheel*. Accessed: Dec. 03, 2022. [Online]. Available: https://sketchfab.com/models/bdead7a383844203b8ea245d464ce36e/embed?autostart=1

[120] B. Sekachev *et al.*, “opencv/cvat: v1.1.0.” Zenodo, Aug. 31, 2020. doi: 10.5281/zenodo.4009388.

[121] L. Zappella, X. Lladó, and J. Salvi, “Motion segmentation: A review,” *Artificial Intelligence Research and Development*, pp. 398–407, 2008.

[122] A. Cavallaro and T. Ebrahimi, “Video object extraction based on adaptive background and statistical change detection,” in *Visual Communications and Image Processing 2001*, 2000, vol. 4310, pp. 465–475.

[123] F.-H. Cheng and Y.-L. Chen, “Real time multiple objects tracking and identification based on discrete wavelet transform,” *Pattern recognition*, vol. 39, no. 6, pp. 1126–1139, 2006.

[124] A. Colombari, A. Fusiello, and V. Murino, “Segmentation and tracking of multiple video objects,” *Pattern Recognition*, vol. 40, no. 4, pp. 1307–1317, Apr. 2007, doi: 10.1016/j.patcog.2006.07.008.

[125] D. Cremers and S. Soatto, “Motion Competition: A Variational Approach to Piecewise Parametric Motion Segmentation,” *Int J Comput Vision*, vol. 62, no. 3, pp. 249–265, May 2005, doi: 10.1007/s11263-005-4882-4.

[126] H. Shen, L. Zhang, B. Huang, and P. Li, “A MAP approach for joint motion estimation, segmentation, and super resolution,” *IEEE Transactions on Image processing*, vol. 16, no. 2, pp. 479–490, 2007.

[127] Y. Rathi, N. Vaswani, A. Tannenbaum, and A. Yezzi, “Tracking deforming objects using particle filtering for geometric active contours,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 8, pp. 1470–1475, 2007.

[128] R. Stolkin, A. Greig, M. Hodgetts, and J. Gilby, “An EM/E-MRF algorithm for adaptive model based tracking in extremely poor visibility,” *Image and Vision Computing*, vol. 26, no. 4, pp. 480–495, Apr. 2008, doi: 10.1016/j.imavis.2007.06.008.

[129] D. J. Felleman and D. C. Van Essen, “Distributed hierarchical processing in the primate cerebral cortex.,” *Cerebral cortex (New York, NY: 1991)*, vol. 1, no. 1, pp. 1–47, 1991.

[130] L. Wiskott, “Segmentation from motion: Combining Gabor- and Mallat-wavelets to overcome aperture and correspondence problem,” in *Computer Analysis of Images and Patterns*, Berlin, Heidelberg, 1997, pp. 329–336. doi: 10.1007/3-540-63460-6_134.

[131] J. Mattheus, H. Grobler, and A. M. Abu-Mahfouz, “A review of motion segmentation: Approaches and major challenges,” in *2020 2nd International multidisciplinary information technology and engineering conference (IMITEC)*, 2020, pp. 1–8.

[132] M. Kong, J.-P. Leduc, B. K. Ghosh, and V. M. Wickerhauser, “Spatio-temporal continuous wavelet transforms for motion-based segmentation in real image sequences,” in *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269)*, 1998, vol. 2, pp. 662–666.

[133] S. Anthwal and D. Ganotra, “An overview of optical flow-based approaches for motion segmentation,” *The Imaging Science Journal*, vol. 67, no. 5, pp. 284–294, 2019.

[134] D. Sun, S. Roth, J. P. Lewis, and M. J. Black, “Learning optical flow,” in *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III 10*, 2008, pp. 83–97.

[135] J. Y. Wang and E. H. Adelson, “Layered representation for motion analysis,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1993, pp. 361–366.

[136] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *IJCAI’81: 7th international joint conference on Artificial intelligence*, 1981, pp. 674–679.

[137] M. Pawan Kumar, P. H. S. Torr, and A. Zisserman, “Learning Layered Motion Segmentations of Video,” *Int J Comput Vis*, vol. 76, no. 3, pp. 301–319, Mar. 2008, doi: 10.1007/s11263-007-0064-x.

[138] Y. Y. Boykov and M.-P. Jolly, “Interactive graph cuts for optimal boundary & region segmentation of objects in ND images,” in *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, IEEE, 2001, pp. 105–112.

[139] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[140] C. Min and G. Medioni, “Inferring segmented dense motion layers using 5D tensor voting,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 9, pp. 1589–1602, 2008.

[141] G. Medioni, M.-S. Lee, and C.-K. Tang, *A computational framework for segmentation and grouping*. Elsevier, 2000.

[142] L. Zappella, “Manifold clustering for motion segmentation,” Ph.D. Thesis, Universitat de Girona, 2011. Accessed: Apr. 15, 2023. [Online]. Available: https://www.tdx.cat/handle/10803/34765

[143] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman, “Clustering appearances of objects under varying illumination conditions,” in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, IEEE, 2003, p. I–I.

[144] N. P. Da Silva and J. P. Costeira, “Subspace segmentation with outliers: a grassmannian approach to the maximum consensus subspace,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2008, pp. 1–6.

[145] 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.

[146] J. Yan and M. Pollefeys, “Articulated motion segmentation using ransac with priors,” in *Dynamical Vision: ICCV 2005 and ECCV 2006 Workshops, WDV 2005 and WDV*

2006, Beijing, China, October 21, 2005, Graz, Austria, May 13, 2006. Revised Papers, Springer, 2007, pp. 75–85.

[147] C. Tomasi and T. Kanade, “Shape and motion from image streams under orthography: a factorization method,” *Int J Comput Vision*, vol. 9, no. 2, pp. 137–154, Nov. 1992, doi: 10.1007/BF00129684.

[148] D. L. Donoho, “Compressed sensing,” *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[149] S. R. Rao, R. Tron, R. Vidal, and Y. Ma, “Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories,” in *2008 IEEE conference on computer vision and pattern recognition*, IEEE, 2008, pp. 1–8.

[150] Y. Ma, H. Derksen, W. Hong, and J. Wright, “Segmentation of multivariate mixed data via lossy data coding and compression,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 9, pp. 1546–1562, 2007.

[151] L. Zappella, X. Lladó, E. Provenzi, and J. Salvi, “Enhanced local subspace affinity for feature-based motion segmentation,” *Pattern Recognition*, vol. 44, no. 2, pp. 454–470, 2011.

[152] E. E. R. Vidal, “Sparse subspace clustering,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol*, 2009, pp. 2790–2797.

[153] D. L. Donoho, “For most large underdetermined systems of linear equations the minimal $\ell 1$-norm solution is also the sparsest solution,” *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 6, pp. 797–829, 2006.

[154] A. Ng, M. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” *Advances in neural information processing systems*, vol. 14, 2001.

[155] G. H. Golub and C. Reinsch, “Singular value decomposition and least squares solutions,” *Linear algebra*, vol. 2, pp. 134–151, 1971.

[156] J. Yan and M. Pollefeys, “A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 865–877, 2008.

[157] J. P. Costeira and T. Kanade, “A multibody factorization method for independently moving objects,” *International Journal of Computer Vision*, vol. 29, pp. 159–179, 1998.

[158] G. Chen and G. Lerman, “Spectral curvature clustering (SCC),” *International Journal of Computer Vision*, vol. 81, pp. 317–330, 2009.

[159] G. Chen and G. Lerman, “Motion segmentation by scc on the hopkins 155 database,” in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, IEEE, 2009, pp. 759–764.

[160] J. Yan and M. Pollefeys, “A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate,” in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part IV 9*, Springer, 2006, pp. 94–106.

[161] P. Bideau, A. RoyChowdhury, R. R. Menon, and E. Learned-Miller, “Supplementary Material — The Best of Both Worlds: Combining CNNs and Geometric Constraints for Hierarchical Motion Segmentation”.

[162] T. Kobayashi and N. Otsu, “Von mises-fisher mean shift for clustering on a hypersphere,” in *2010 20th International Conference on Pattern Recognition*, IEEE, 2010, pp. 2130–2133.

[163] J. K. Tsotsos, “Analyzing vision at the complexity level,” *Behavioral and brain sciences*, vol. 13, no. 3, pp. 423–445, 1990.

[164] C. Wloka, *Integrating Overt and Covert Attention Using Peripheral and Central Processing Streams*. York University, 2012.

[165] J. Zhang, S. A. Bargal, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff, “Top-down neural attention by excitation backprop,” *International Journal of Computer Vision*, vol. 126, no. 10, pp. 1084–1102, 2018.

[166] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, “Flickr30k entities: Collecting region-to-phrase correspondences for richer

image-to-sentence models," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2641–2649.

[167] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[168] M. Biparva and J. Tsotsos, "Compact neural representation using attentive network pruning," *arXiv preprint arXiv:2005.04559*, 2020.

[169] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[170] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[171] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[172] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[173] J. Wang and A. L. Yuille, “Semantic part segmentation using compositional model combining shape and appearance,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1788–1797.

[174] P. Mehrani and J. K. Tsotsos, “Self-attention in vision transformers performs perceptual grouping, not attention,” Frontiers in Computer Science, vol. 5, 2023, Accessed: Aug. 25, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fcomp.2023.1178450

[175] R. M. Siegel and H. L. Read, “Analysis of optic flow in the monkey parietal area 7a.,” Cerebral cortex (New York, NY: 1991), vol. 7, no. 4, pp. 327–346, 1997.

[176] H. L. Read and R. M. Siegel, “Modulation of responses to optic flow in area 7a by retinotopic and oculomotor cues in monkey.,” Cerebral cortex (New York, NY: 1991), vol. 7, no. 7, pp. 647–661, 1997.

[177] G. A. Orban, “Higher order visual processing in macaque extrastriate cortex,” Physiological reviews, vol. 88, no. 1, pp. 59–89, 2008.

[178] “Air Force Regulation No. 200-2.” Feb. 05, 1958. [Online]. Available: https://www.cia.gov/readingroom/docs/CIA-RDP81R00560R000100040072-9.pdf

Appendix A

Close-up Views of Outputs from ST-Motion-Net

A.1 Close-up Views of Outputs from 2-Frame and 6-Frame Models

For some example sequences in this thesis, close-up views of the GT outputs for area computations in ST-Motion-Net are shown in Figure 3.5, Figure 3.7, Figure 3.9, Figure 3.11, Figure 3.13, and Figure 6.2. In this section, we provide close-up views of the outputs from the 2-Frame and 6-Frame models of ST-Motion-Net. Table A.1 lists the figures for these sequences, the outputs, and the close-up views for the 2-Frame and 6-Frame model outputs.

Sequence	Outputs	Close-up Views of 2-Frame and 6-Frame Model Outputs
Figure 3.4	Figure 3.5	Figure A.1
Figure 3.6	Figure 3.7	Figure A.2
Figure 3.8	Figure 3.9	Figure A.3
Figure 3.10	Figure 3.11	Figure A.4
Figure 3.12	Figure 3.13	Figure A.5
Figure 6.1	Figure 6.2	Figure A.6

Table A.1: Figures for the example sequences, outputs of area computations of ST-Motion-Net, and the close-up views of 2-Frame and 6-Frame model outputs.

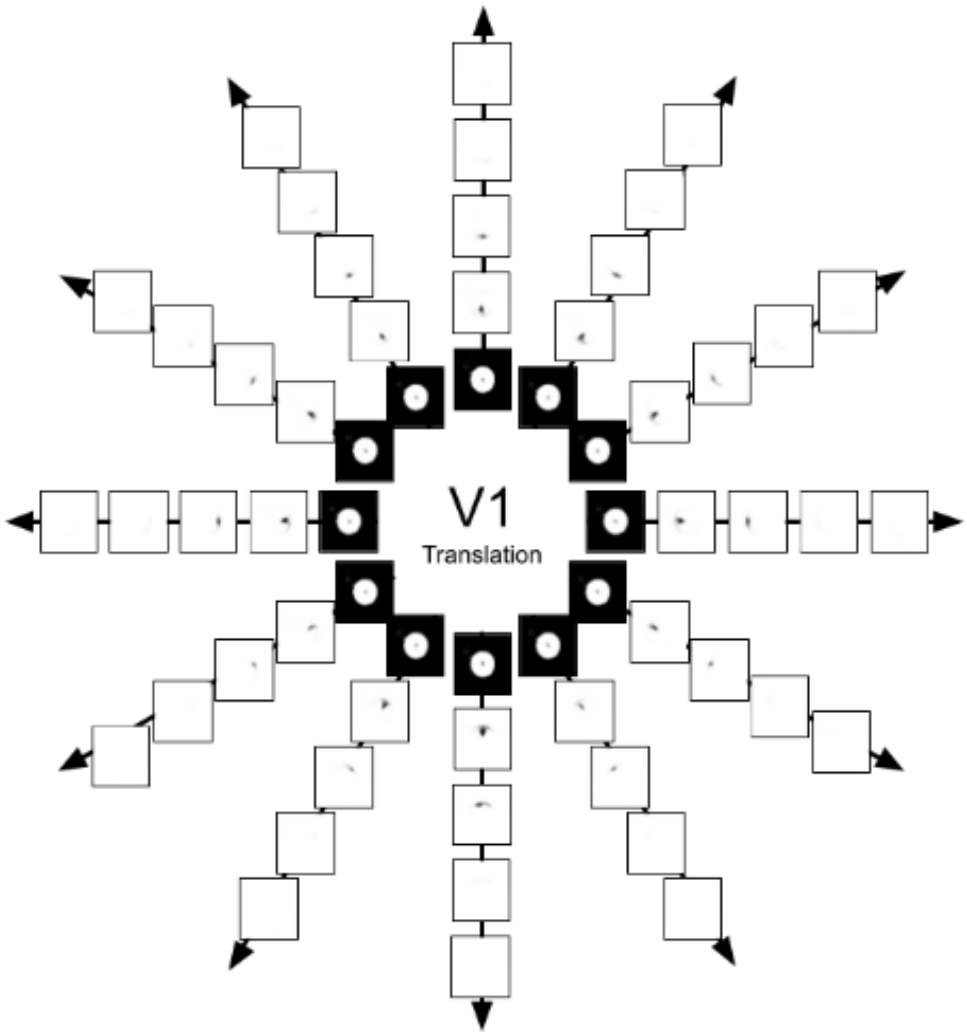

Figure A.1: Close-up view of the output of area computations of ST-Motion-Net for sequence in Figure 3.4. V1 Translation shows the translation output of Area V1. This shows a close-up view of the 2-Frame model output for V1 Translation.

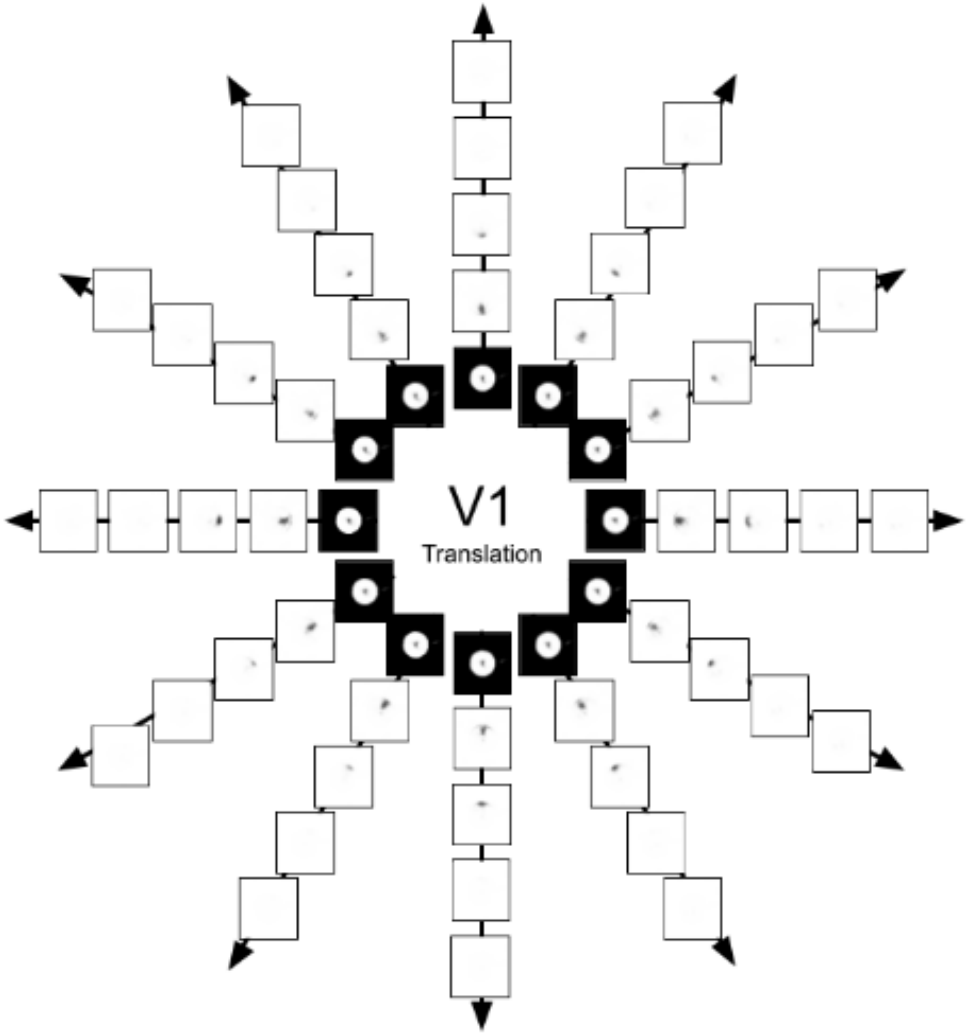

Figure A.1 (continued): This shows a close-up view of the 6-Frame model output for V1 Translation.

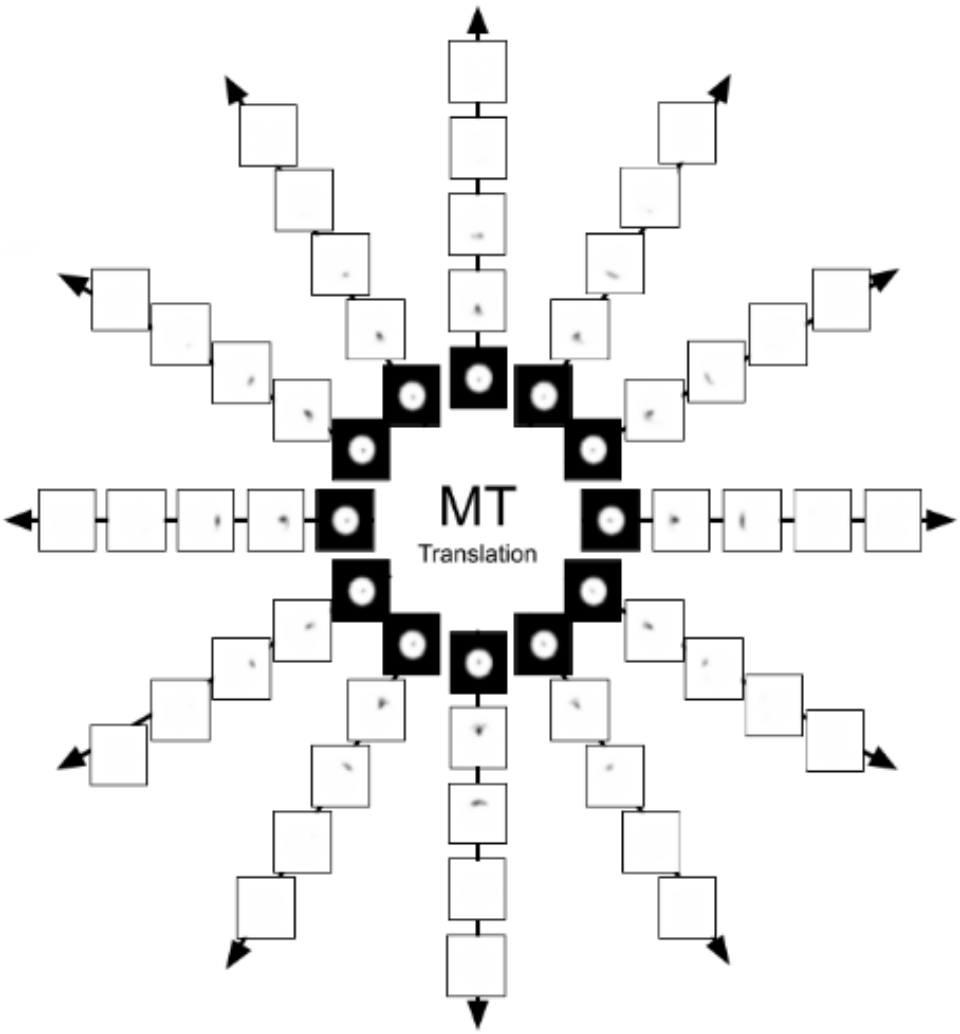

Figure A.1 (continued): MT Translation shows the translation output of Area MT. This shows a close-up view of the 2-Frame model output for MT Translation.

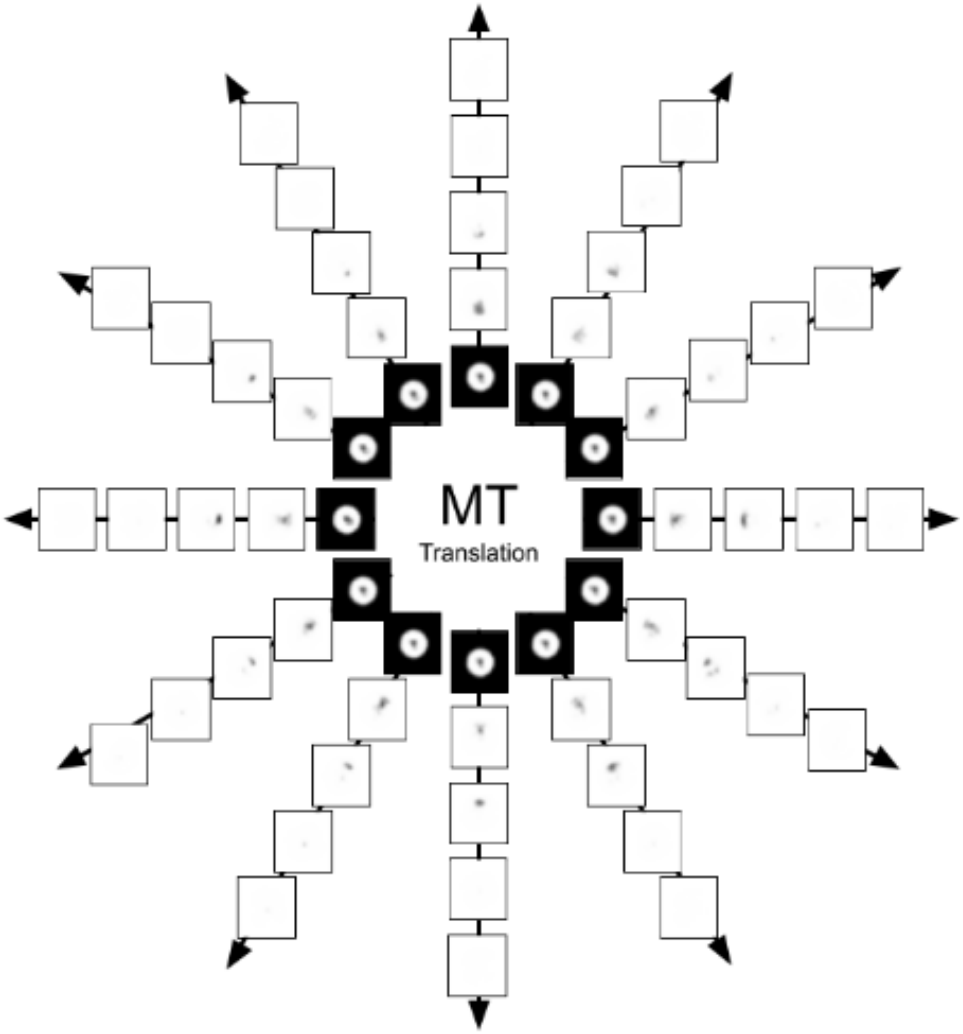

Figure A.1 (continued): This shows a close-up view of the 6-Frame model output for MT Translation.

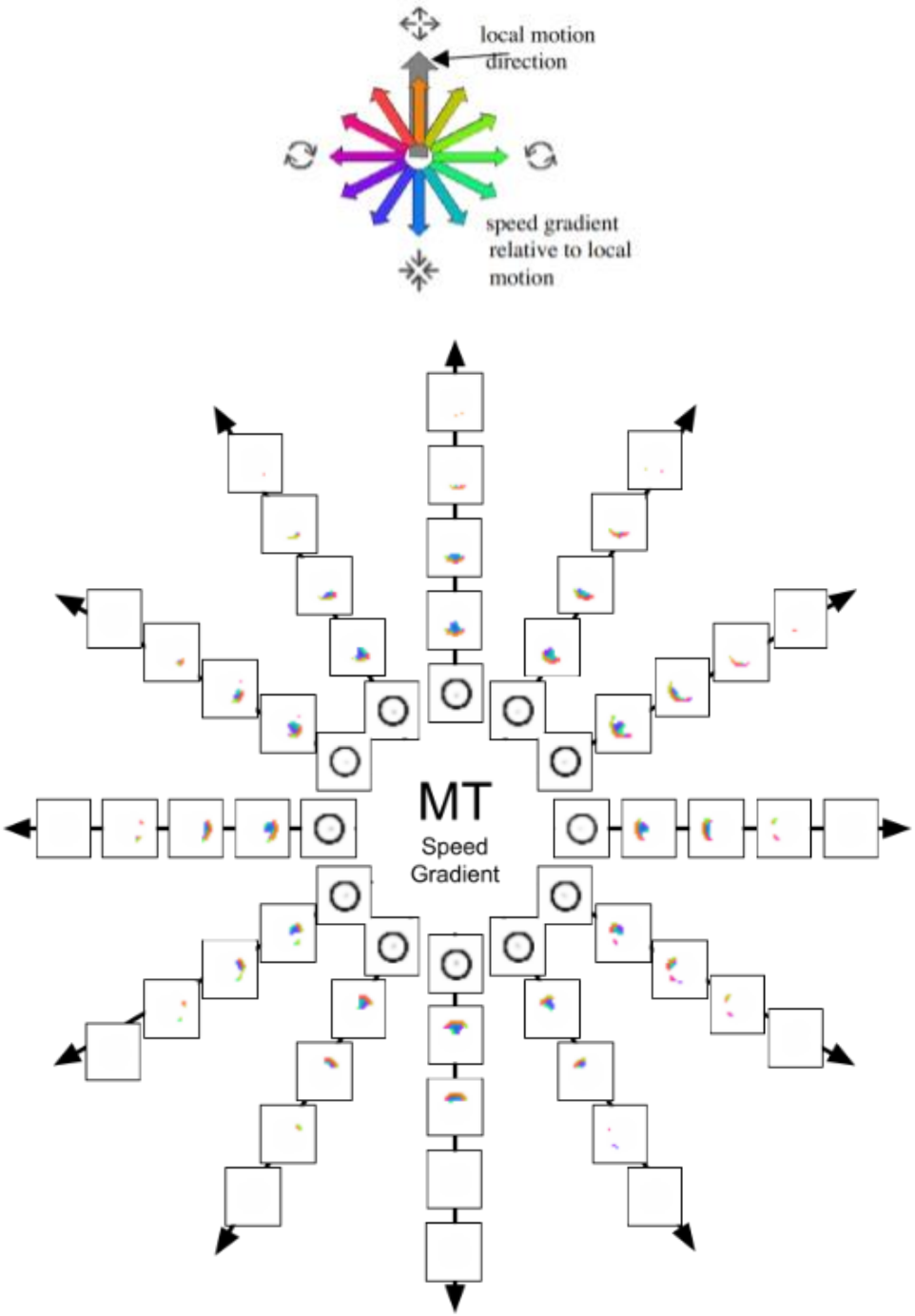

Figure A.1 (continued): Speed gradient outputs in Area MT are shown in a summary representation of the 12 speed gradients for each local direction and speed. Each coloured dot is the maximum value across the 12 representations. This shows a close-up view of the 2-Frame model output for MT Speed Gradients in the summary representation.

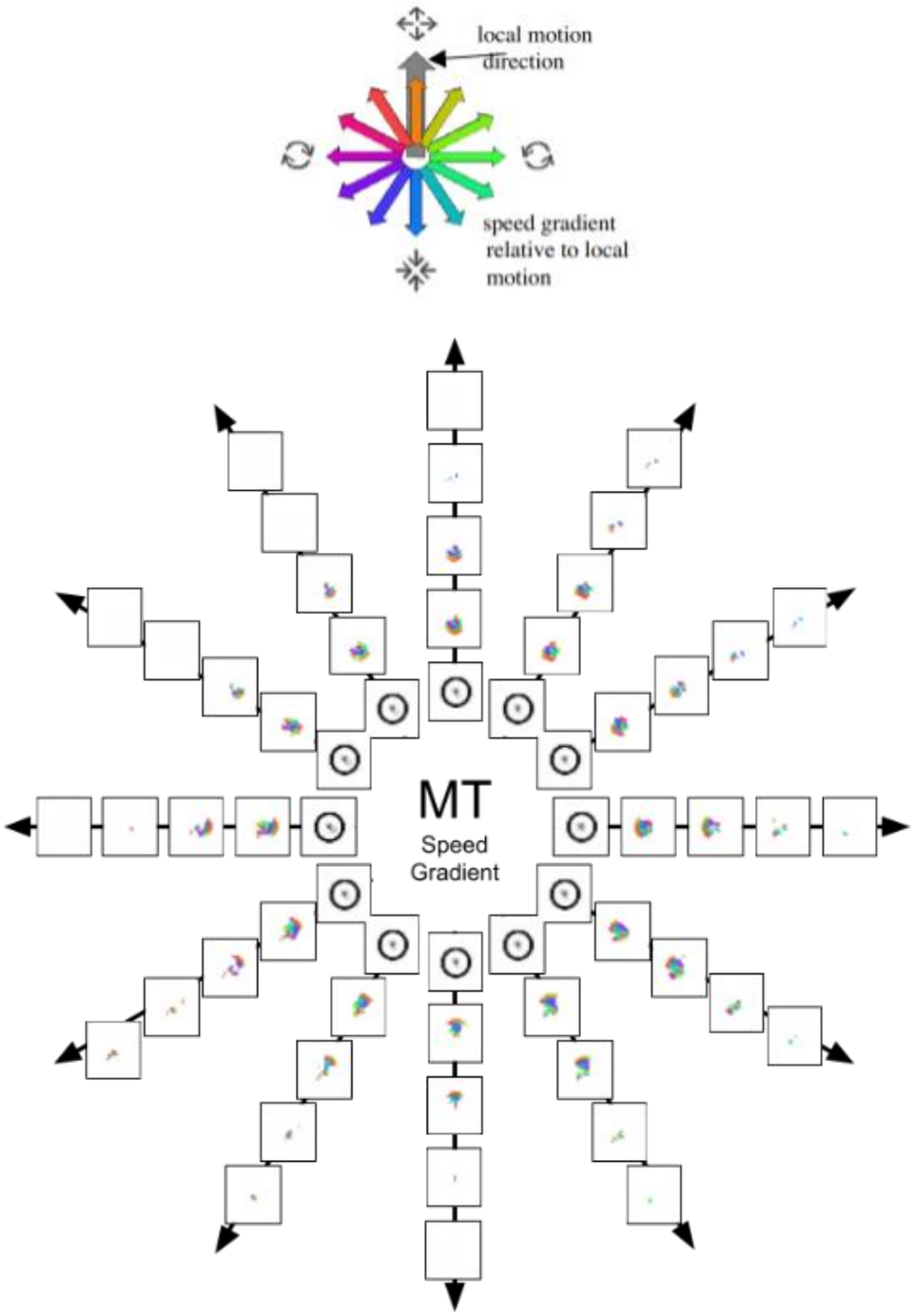

Figure A.1 (continued): This shows a close-up view of the 6-Frame model output for MT Speed Gradients in the summary representation.

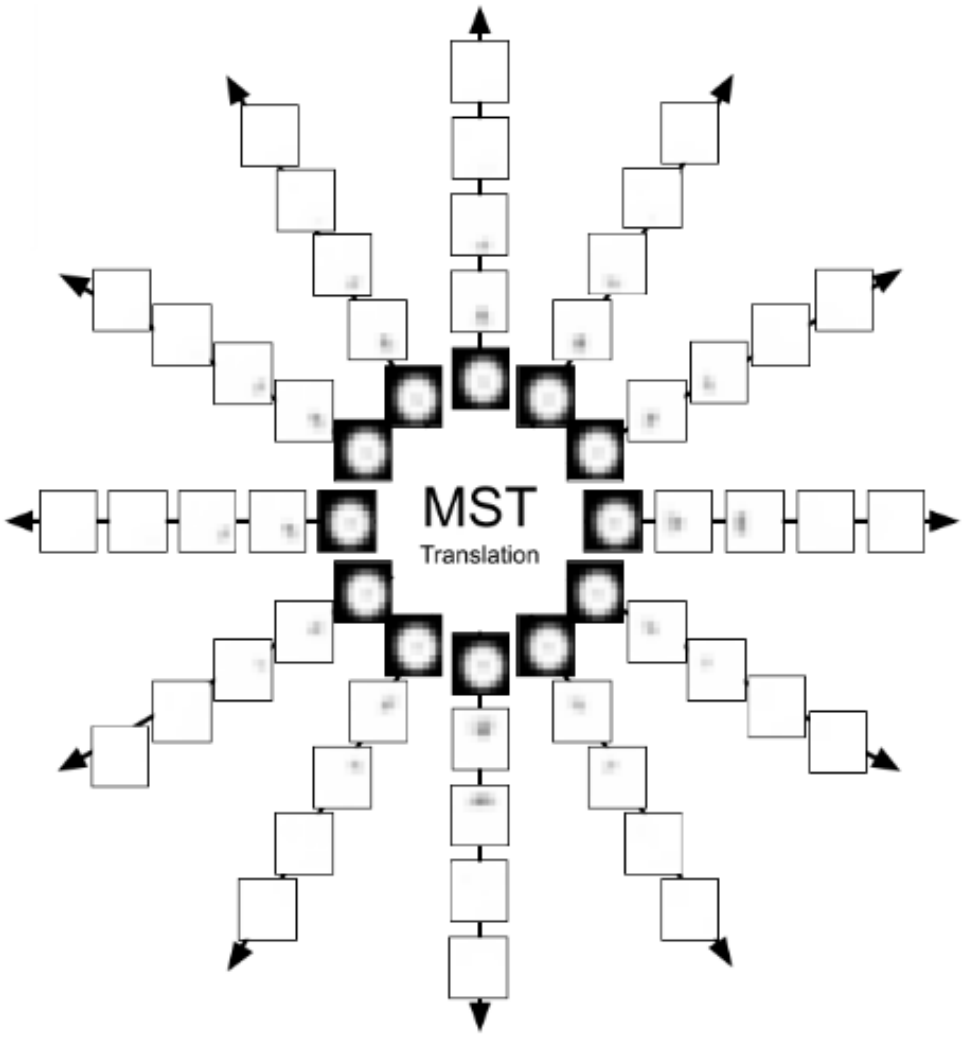

Figure A.1 (continued): MST Translation shows the translation output in Area MST. This shows a close-up view of the 2-Frame model output for MST Translation.

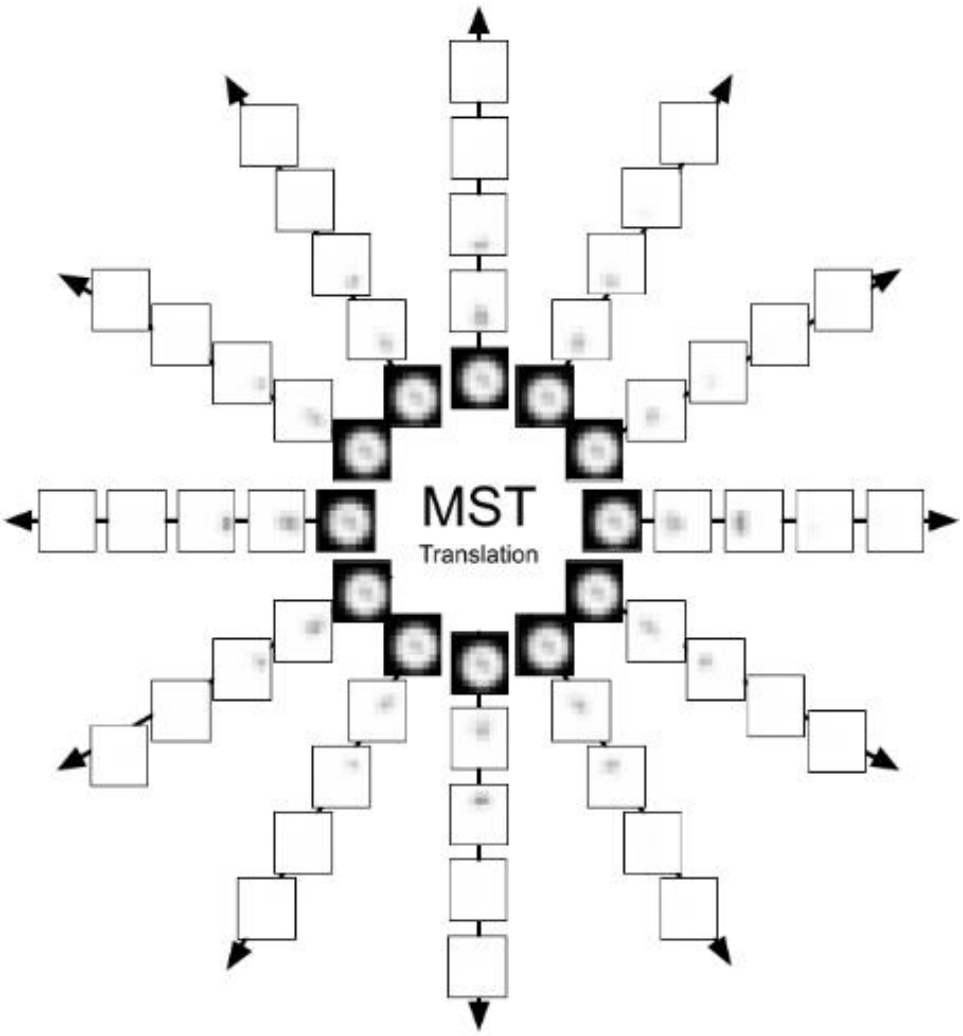

Figure A.1 (continued): This shows a close-up view of the 6-Frame model output for MST Translation.

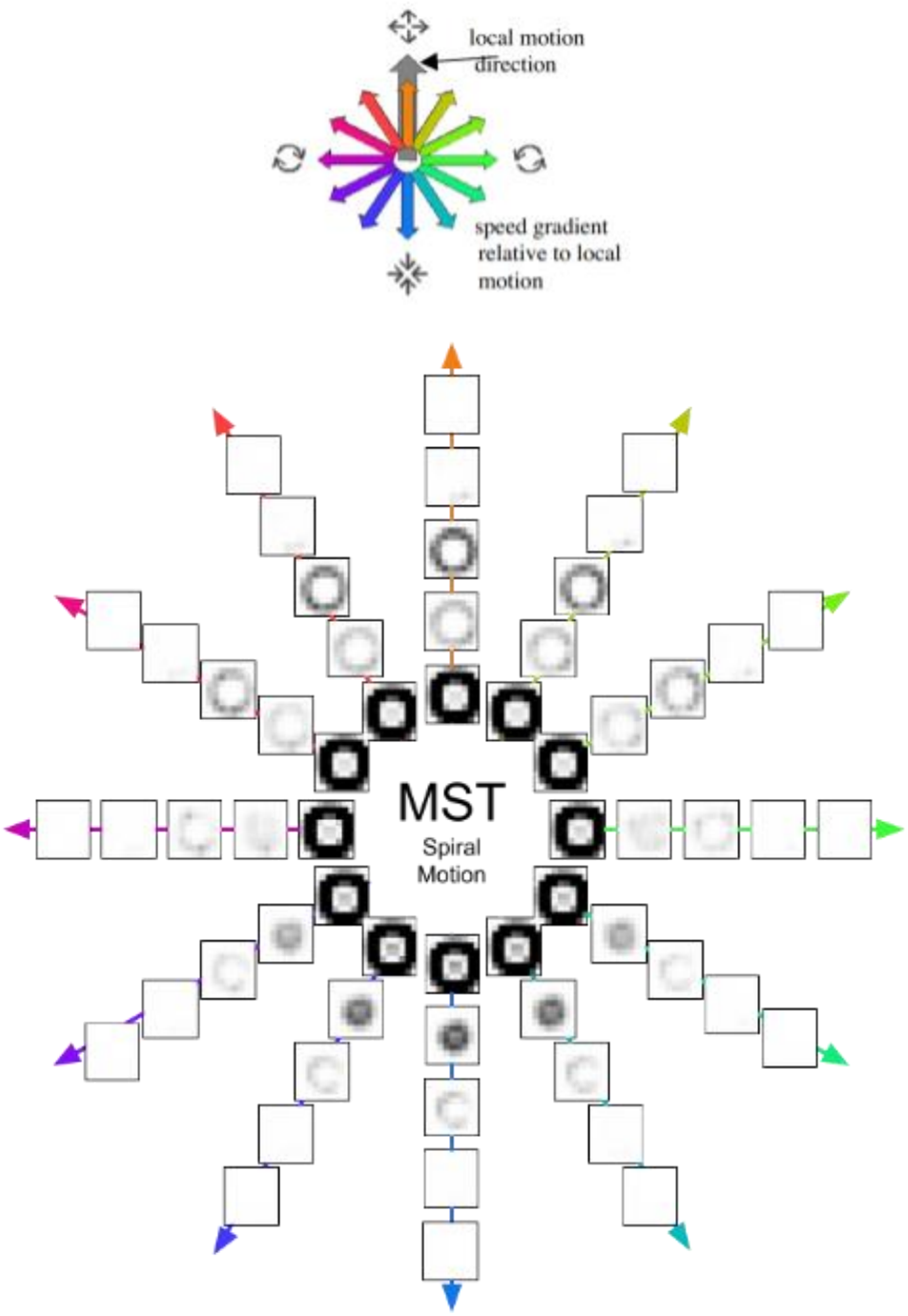

Figure A.1 (continued): MST Spiral Motion shows the generalized spiral output in Area MST. Coloured arrows indicate the inner angle δ between motion and speed gradient (See legend). This shows a close-up view of the 2-Frame model output for MST Spiral Motion.

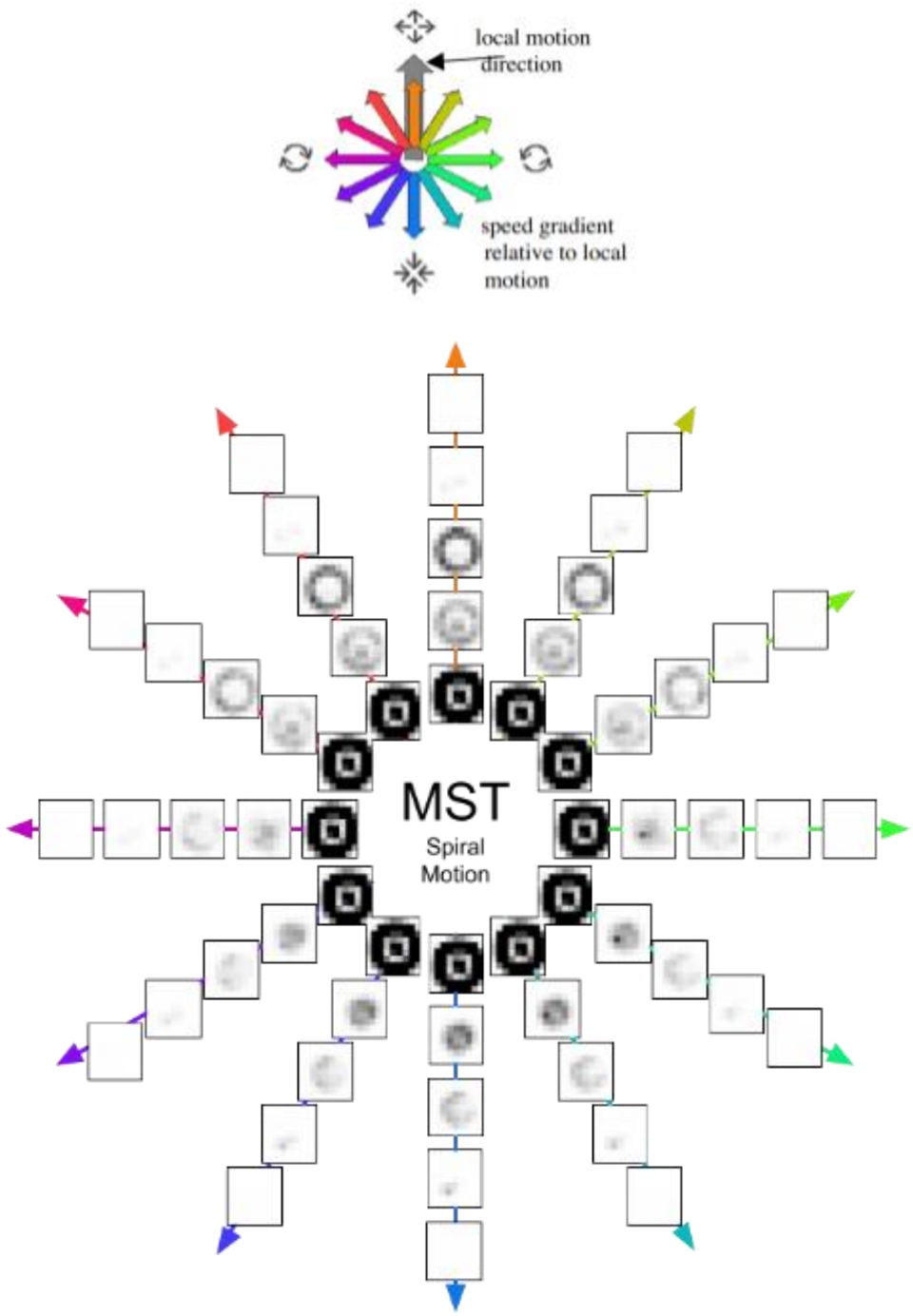

Figure A.1 (continued): This shows a close-up view of the 6-Frame model output for MST Spiral Motion.

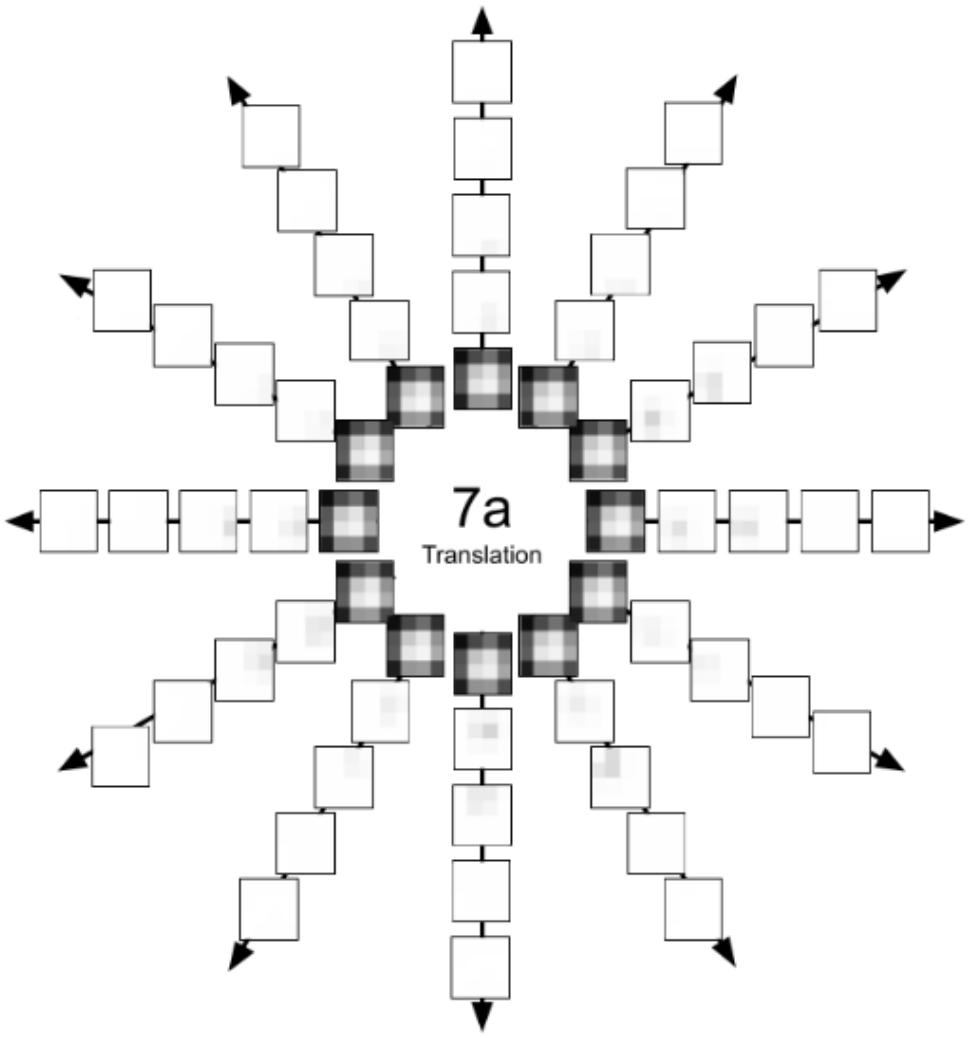

Figure A.1 (continued): 7a Translation shows the translation output in Area 7a. This shows a close-up view of the 2-Frame model output for 7a Translation.

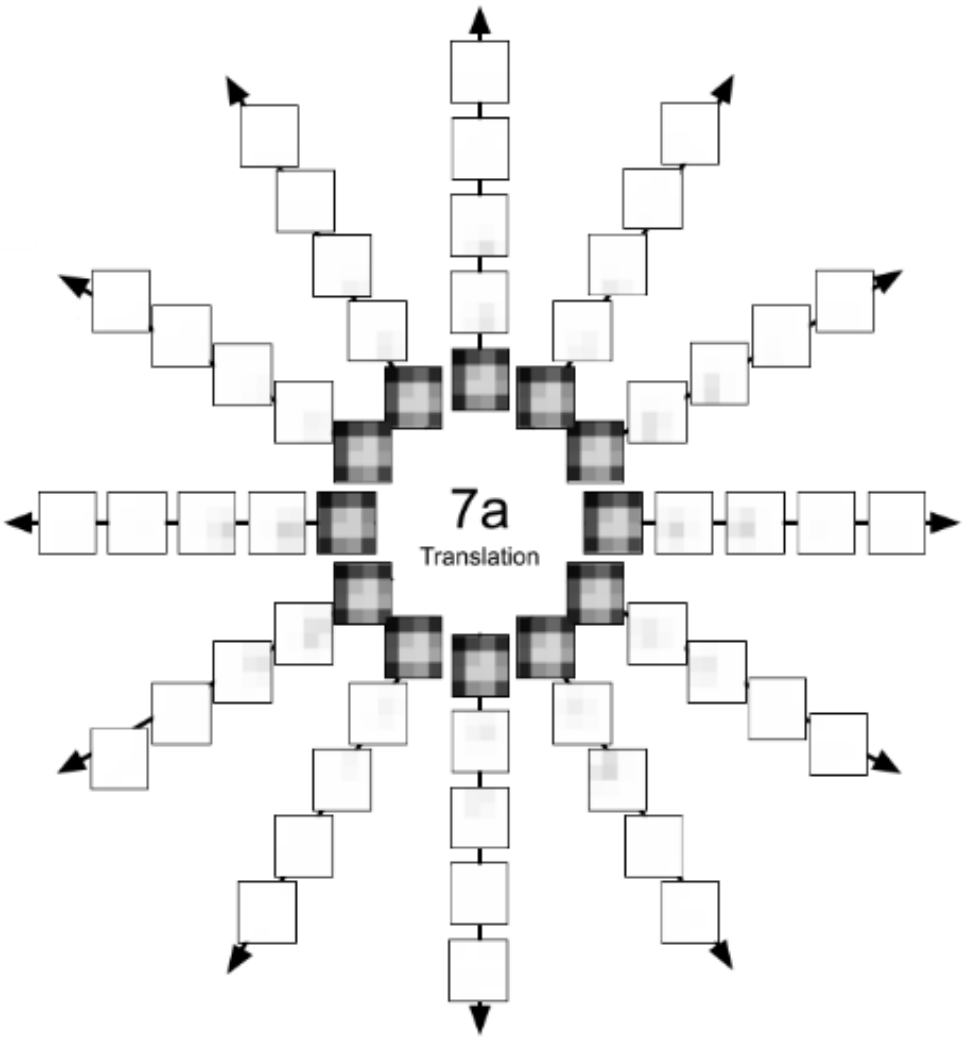

Figure A.1 (continued): This shows a close-up view of the 6-Frame model output for 7a Translation.

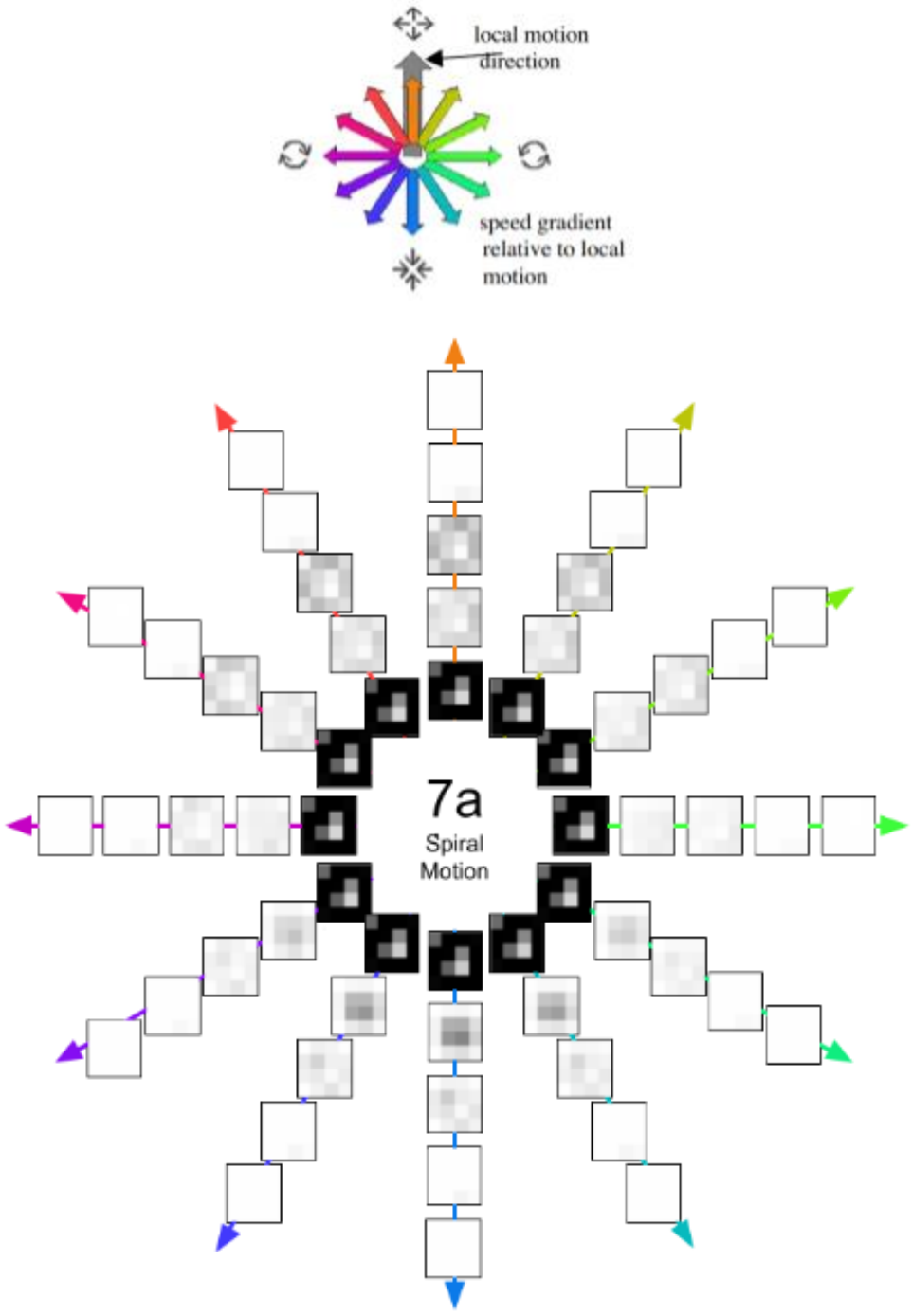

Figure A.1 (continued): 7a Spiral Motion shows the generalized spiral output in Area 7a. Coloured arrows indicate the inner angle δ between motion and speed gradient (See legend). This shows a close-up view of the 2-Frame model output for 7a Spiral Motion.

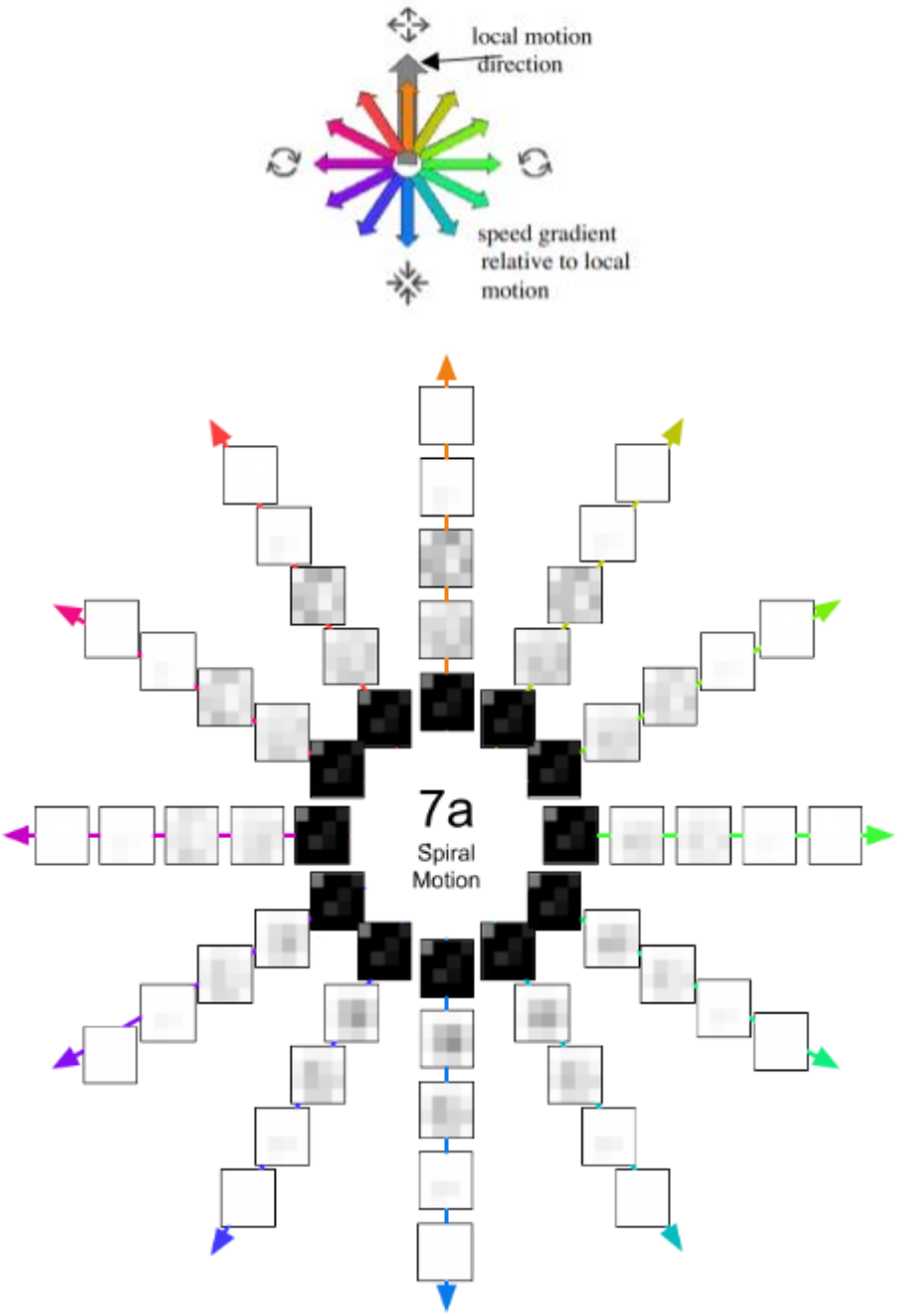

Figure A.1 (continued): This shows a close-up view of the 6-Frame model output for 7a Spiral Motion.

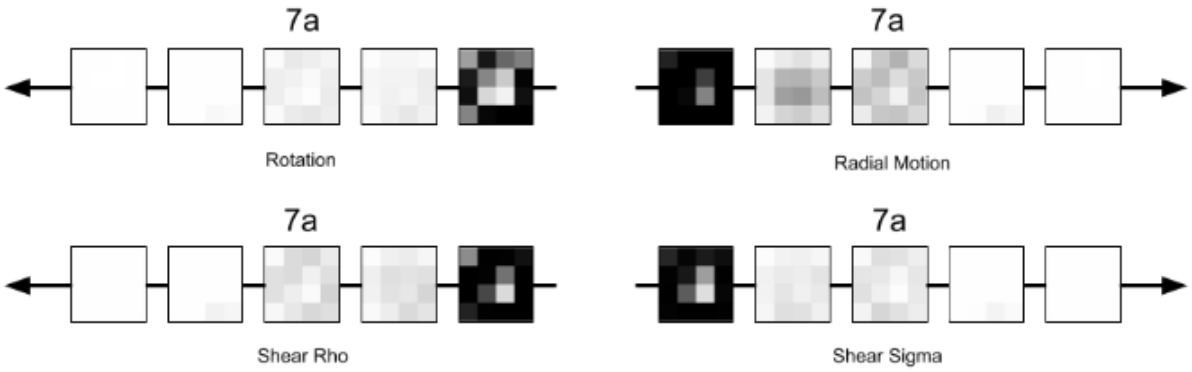

Figure A.1 (continued): Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the 2-Frame model outputs for AP.

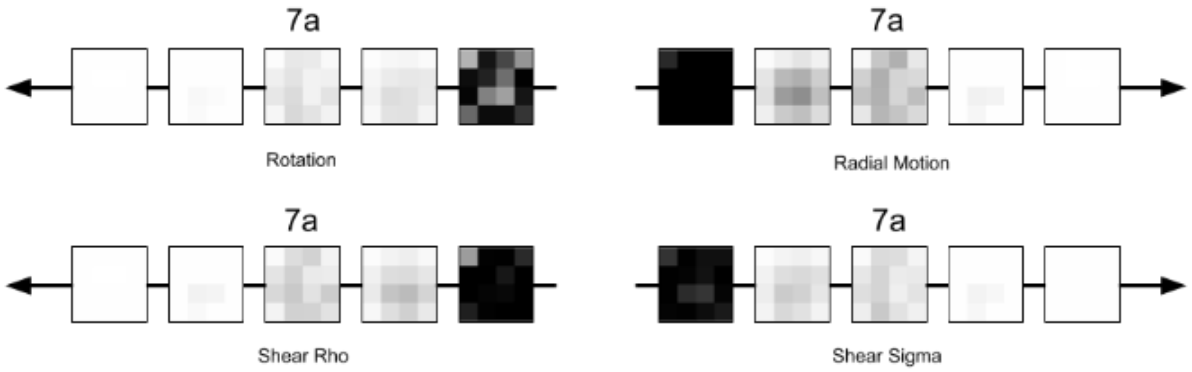

Figure A.1 (continued): This shows a close-up view of the 6-Frame model outputs for AP.

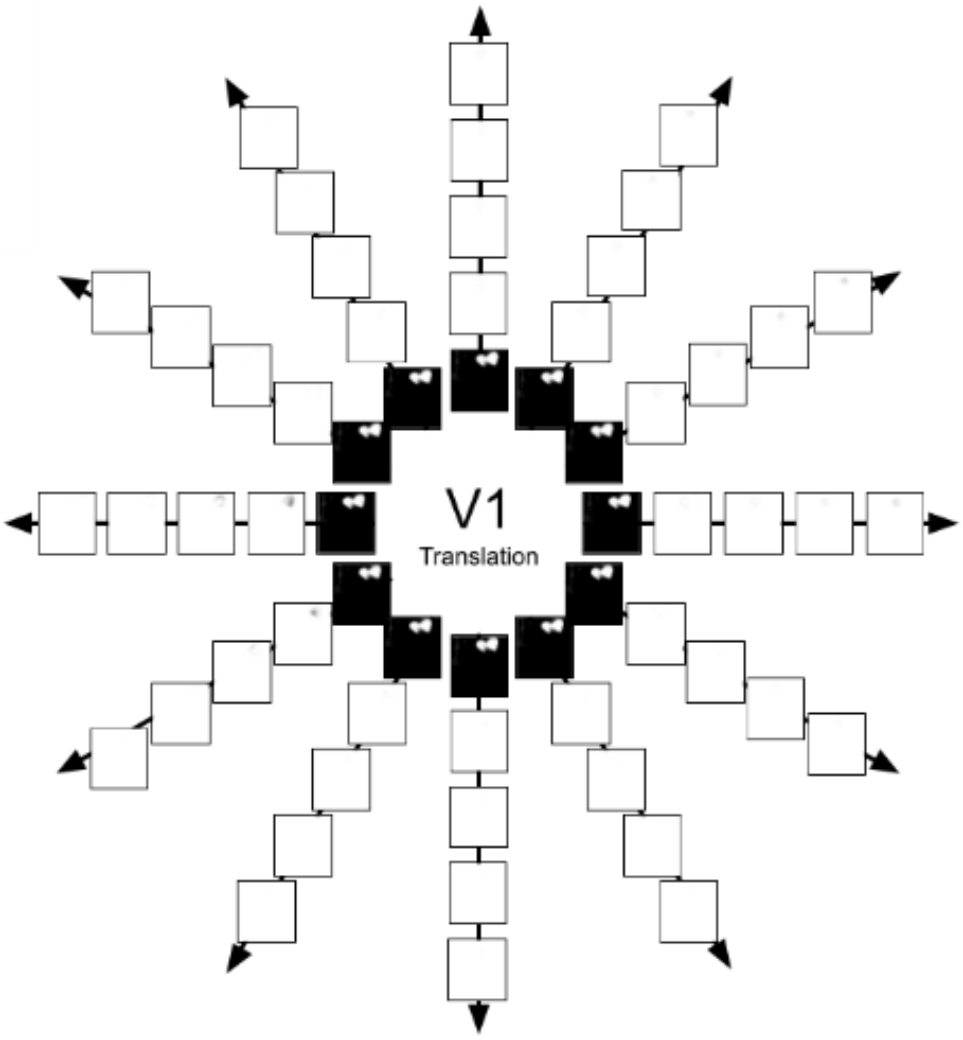

Figure A.2: Close-up view of the output of area computations of ST-Motion-Net for sequence in Figure 3.6. V1 Translation shows the translation output of Area V1. This shows a close-up view of the 2-Frame model output for V1 Translation.

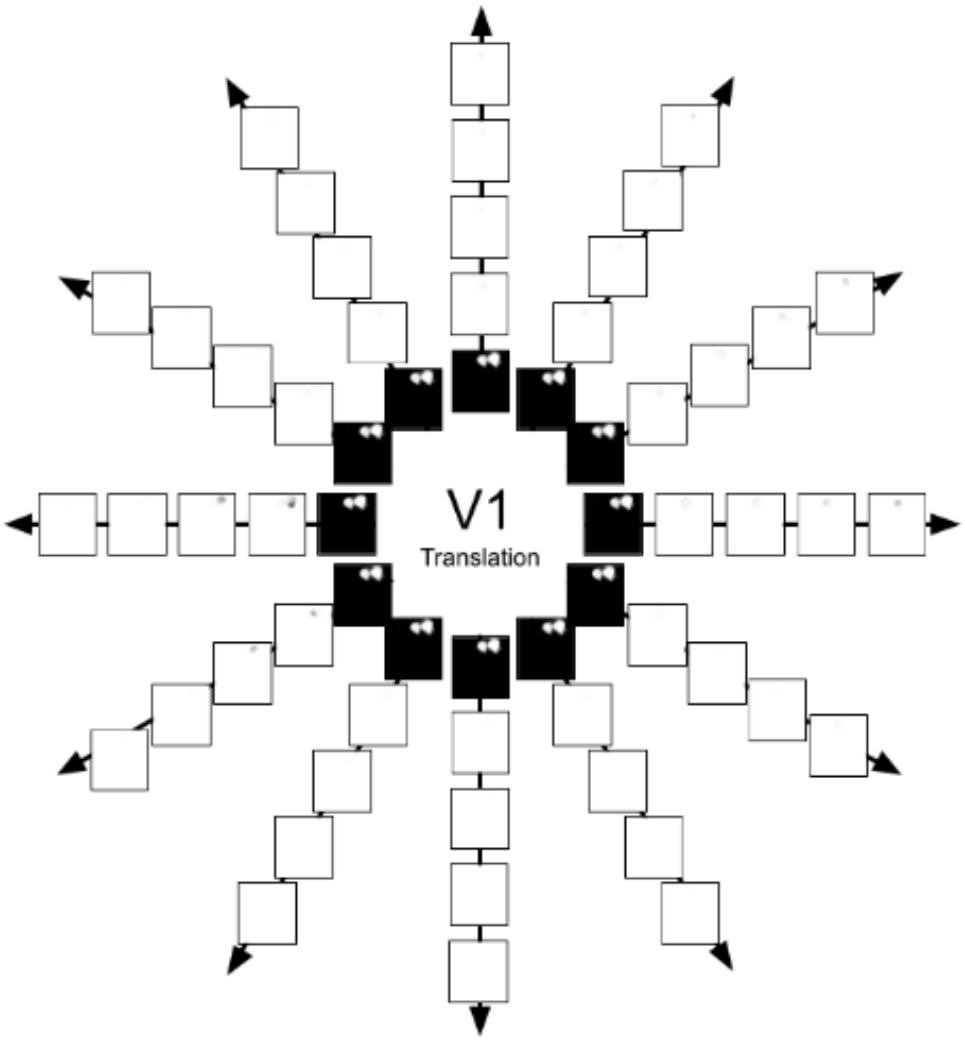

Figure A.2 (continued): This shows a close-up view of the 6-Frame model output for V1 Translation.

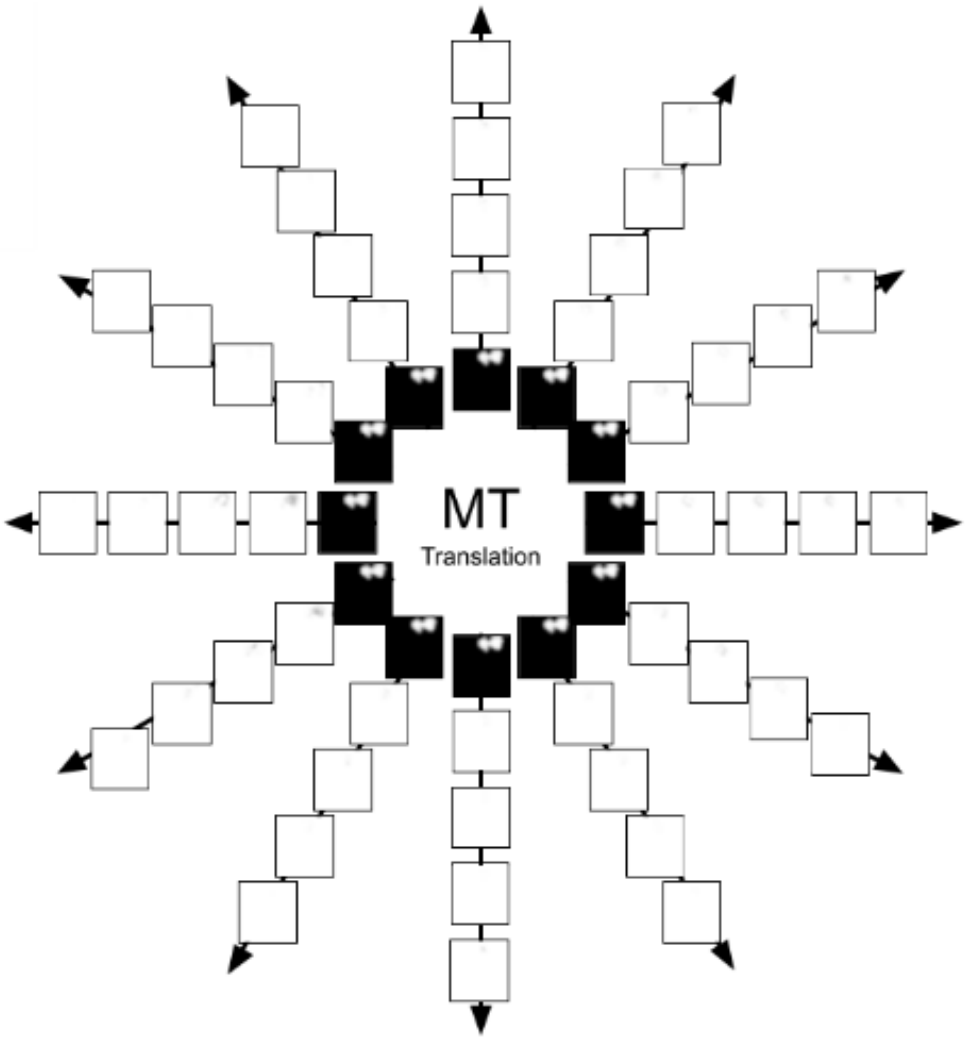

Figure A.2 (continued): MT Translation shows the translation output of Area MT. This shows a close-up view of the 2-Frame model output for MT Translation.

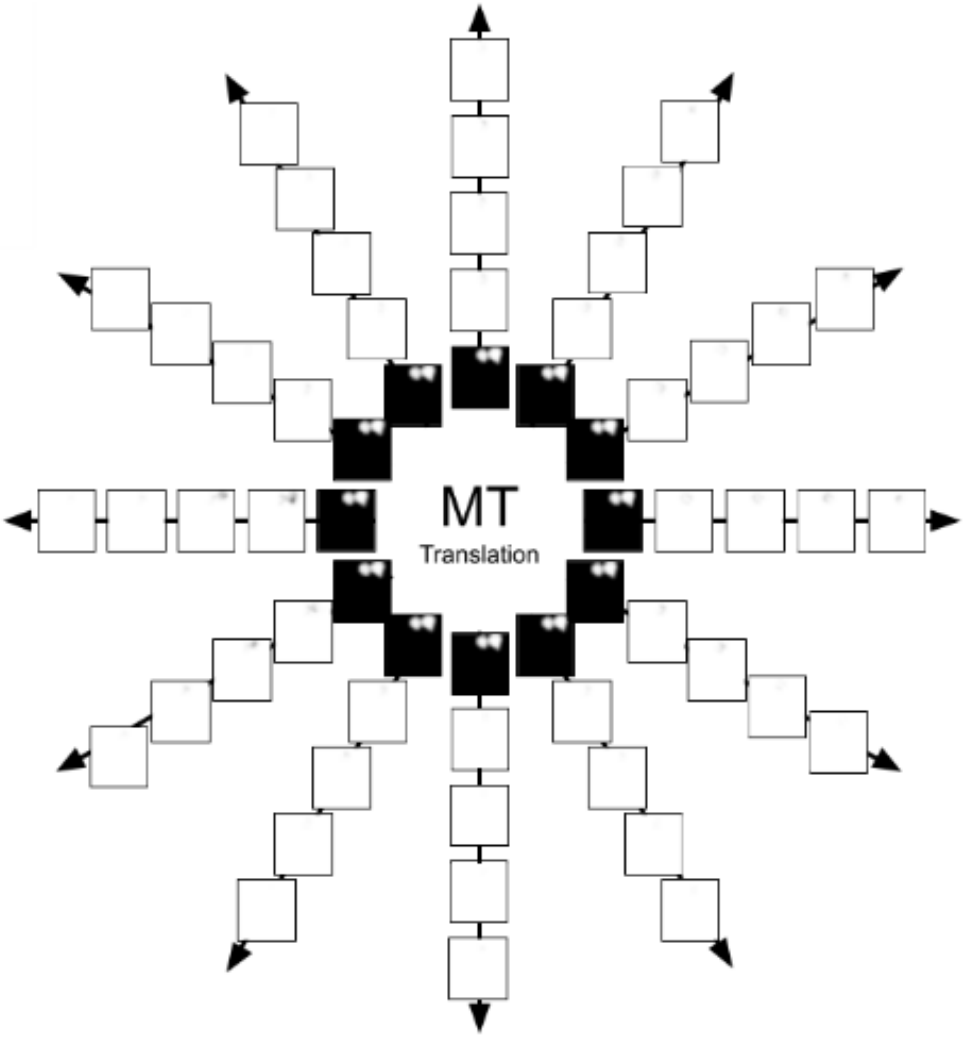

Figure A.2 (continued): This shows a close-up view of the 6-Frame model output for MT Translation.

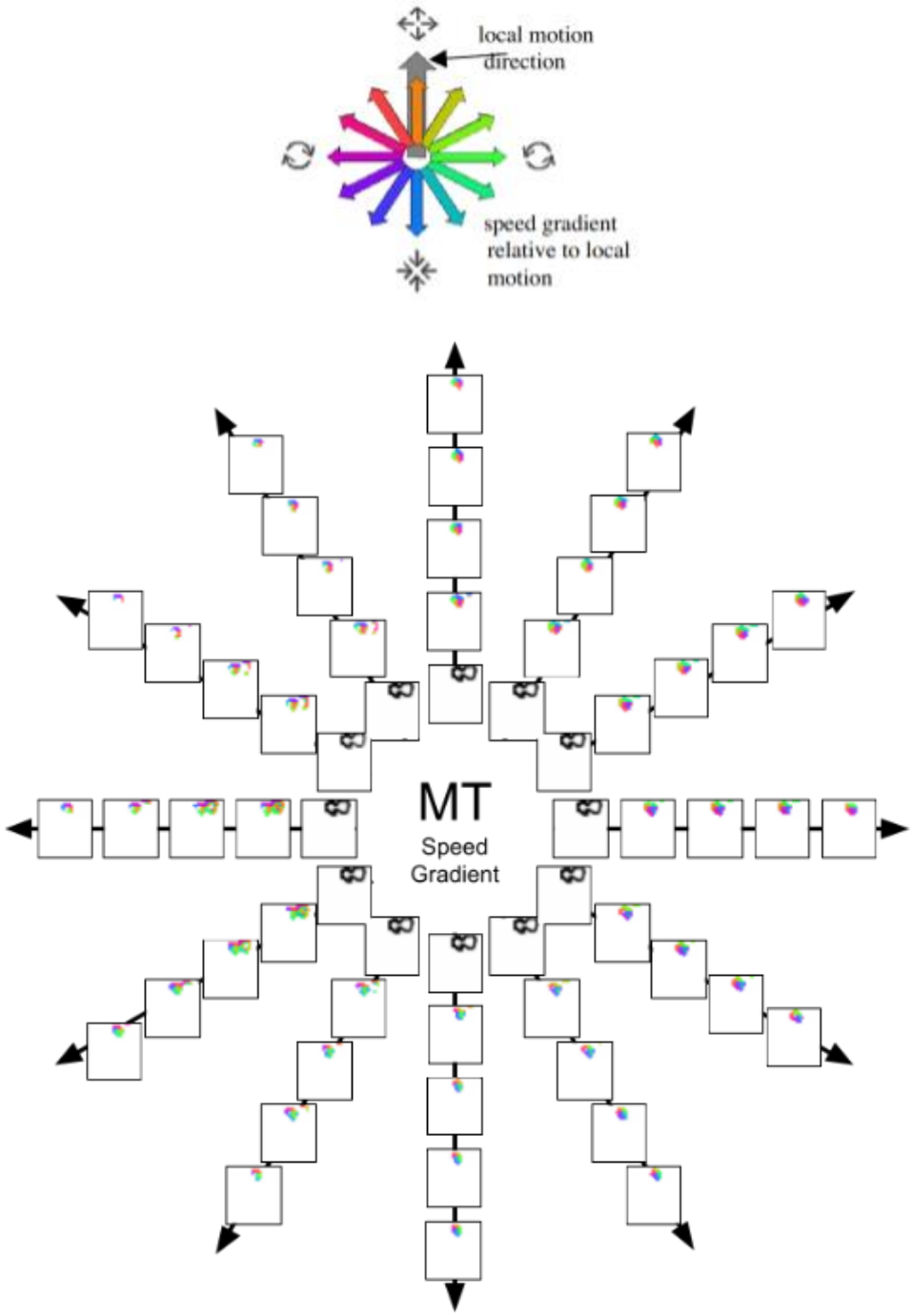

Figure A.2 (continued): Speed gradient outputs in Area MT are shown in a summary representation of the 12 speed gradients for each local direction and speed. Each coloured dot is the maximum value across the 12 representations. This shows a close-up view of the 2-Frame model output for MT Speed Gradients in the summary representation.

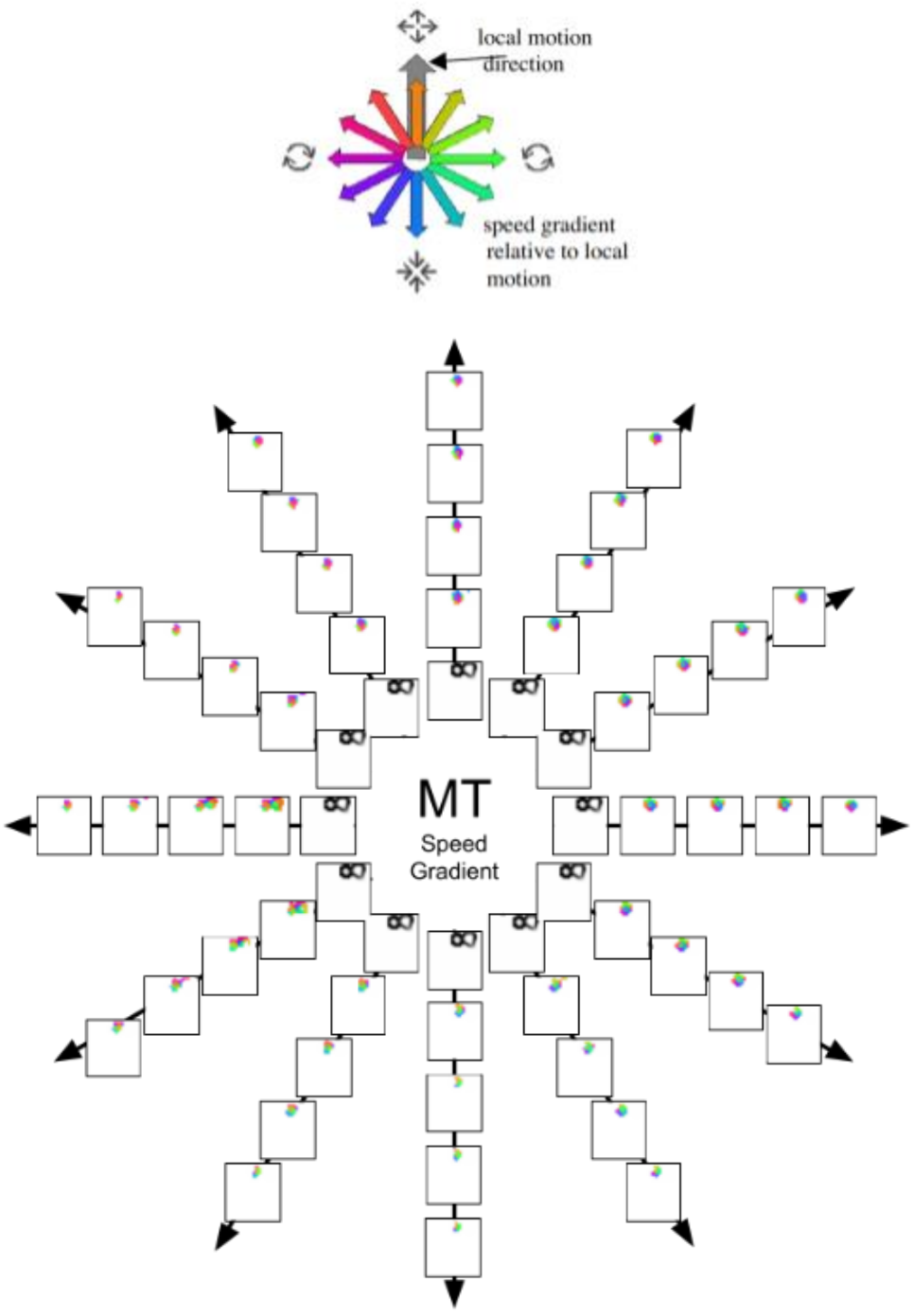

Figure A.2 (continued): This shows a close-up view of the 6-Frame model output for MT Speed Gradients in the summary representation.

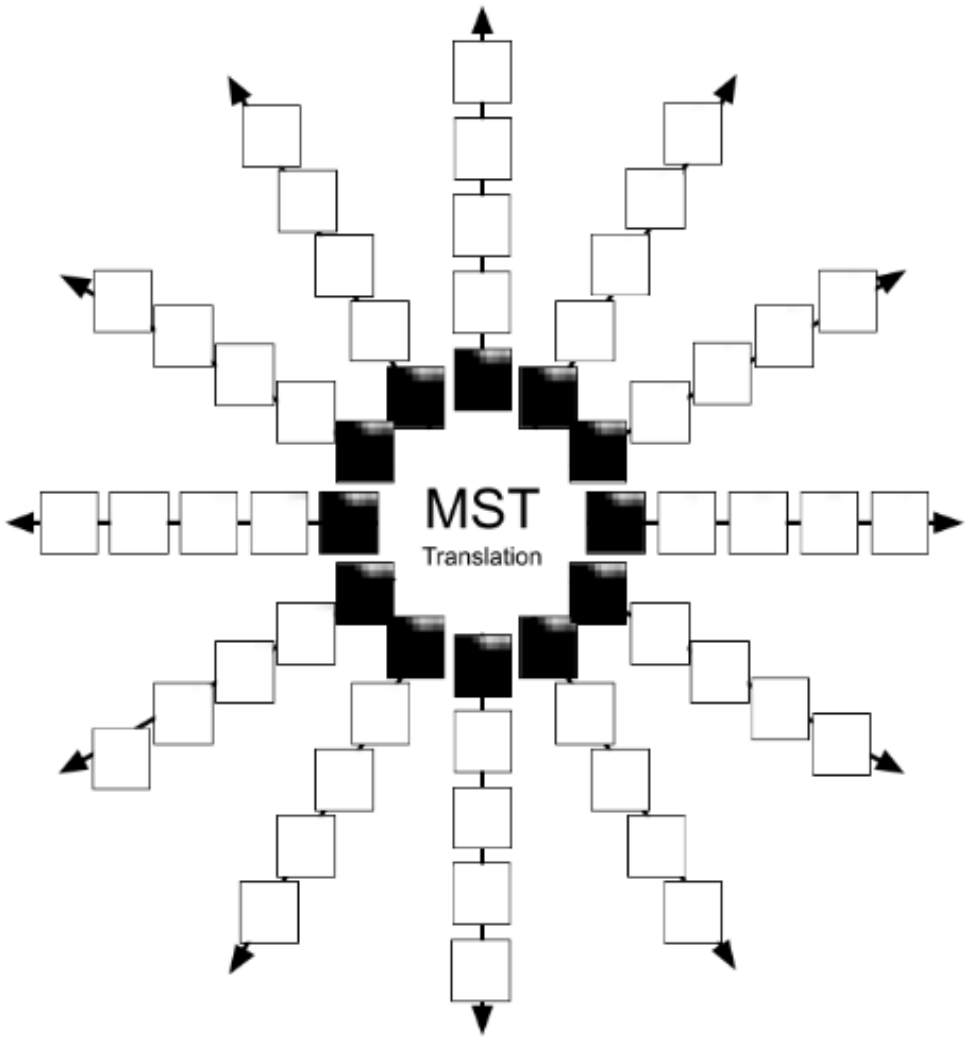

Figure A.2 (continued): MST Translation shows the translation output in Area MST. This shows a close-up view of the 2-Frame model output for MST Translation.

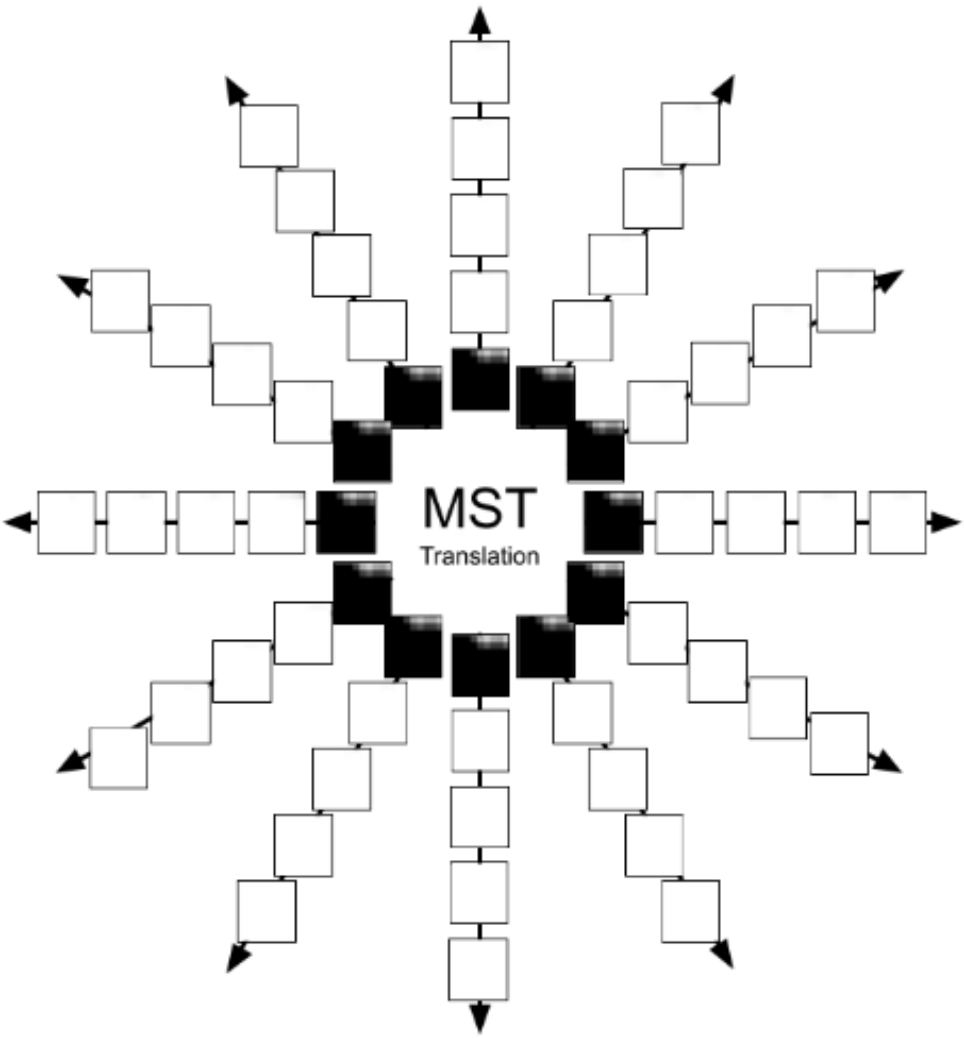

Figure A.2 (continued): This shows a close-up view of the 6-Frame model output for MST Translation.

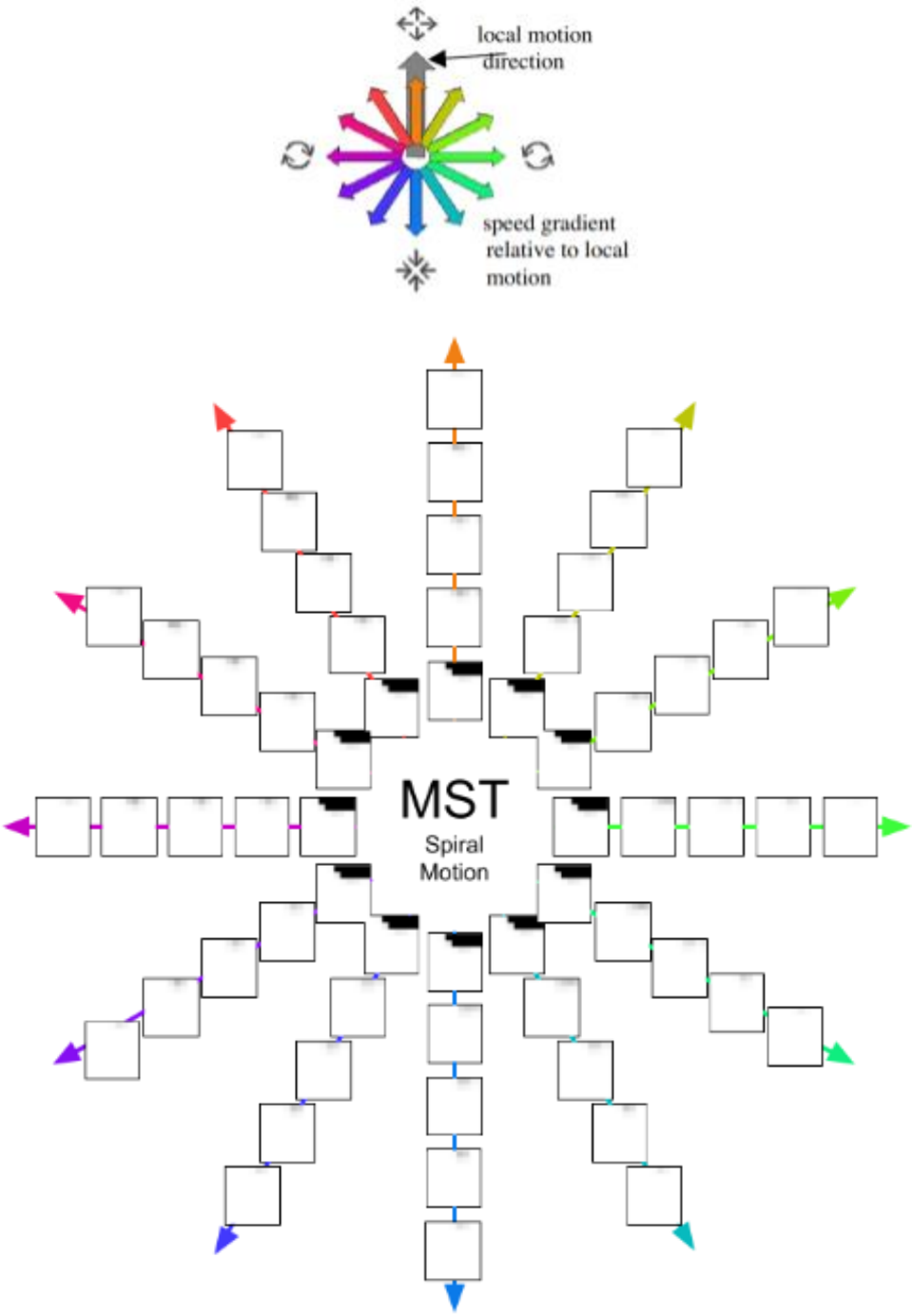

Figure A.2 (continued): MST Spiral Motion shows the generalized spiral output in Area MST. Coloured arrows indicate the inner angle δ between motion and speed gradient (See legend). This shows a close-up view of the 2-Frame model output for MST Spiral Motion.

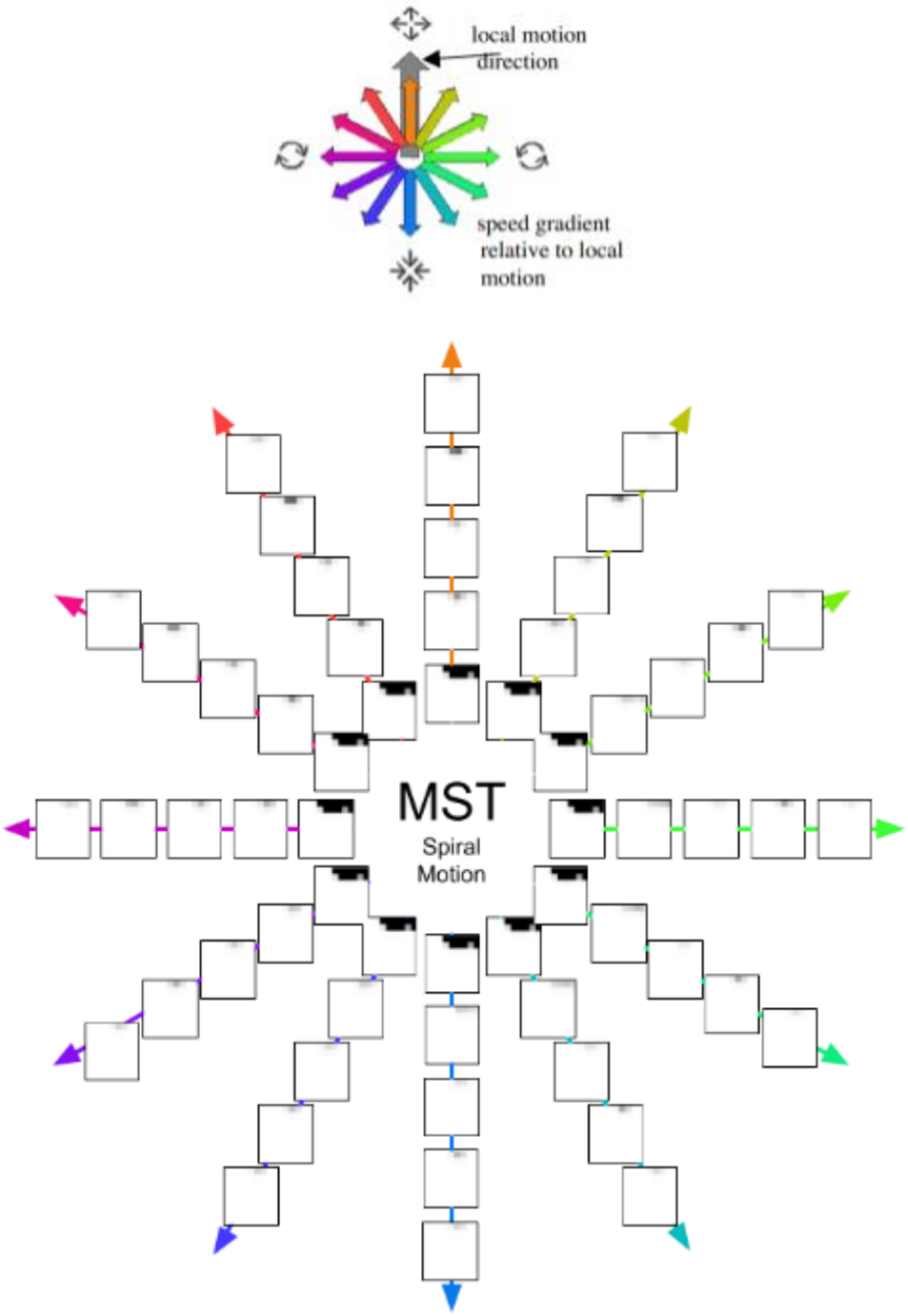

Figure A.2 (continued): This shows a close-up view of the 6-Frame model output for MST Spiral Motion.

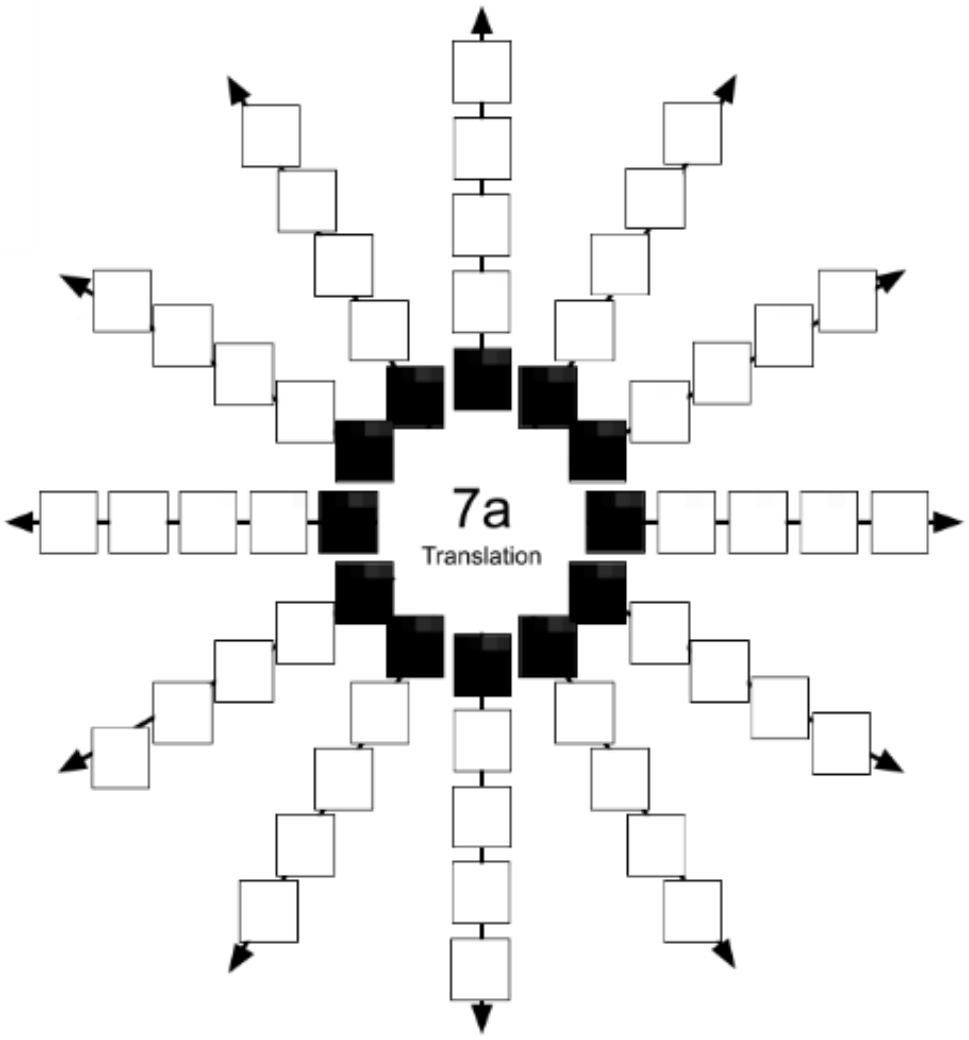

Figure A.2 (continued): 7a Translation shows the translation output in Area 7a. This shows a close-up view of the 2-Frame model output for 7a Translation.

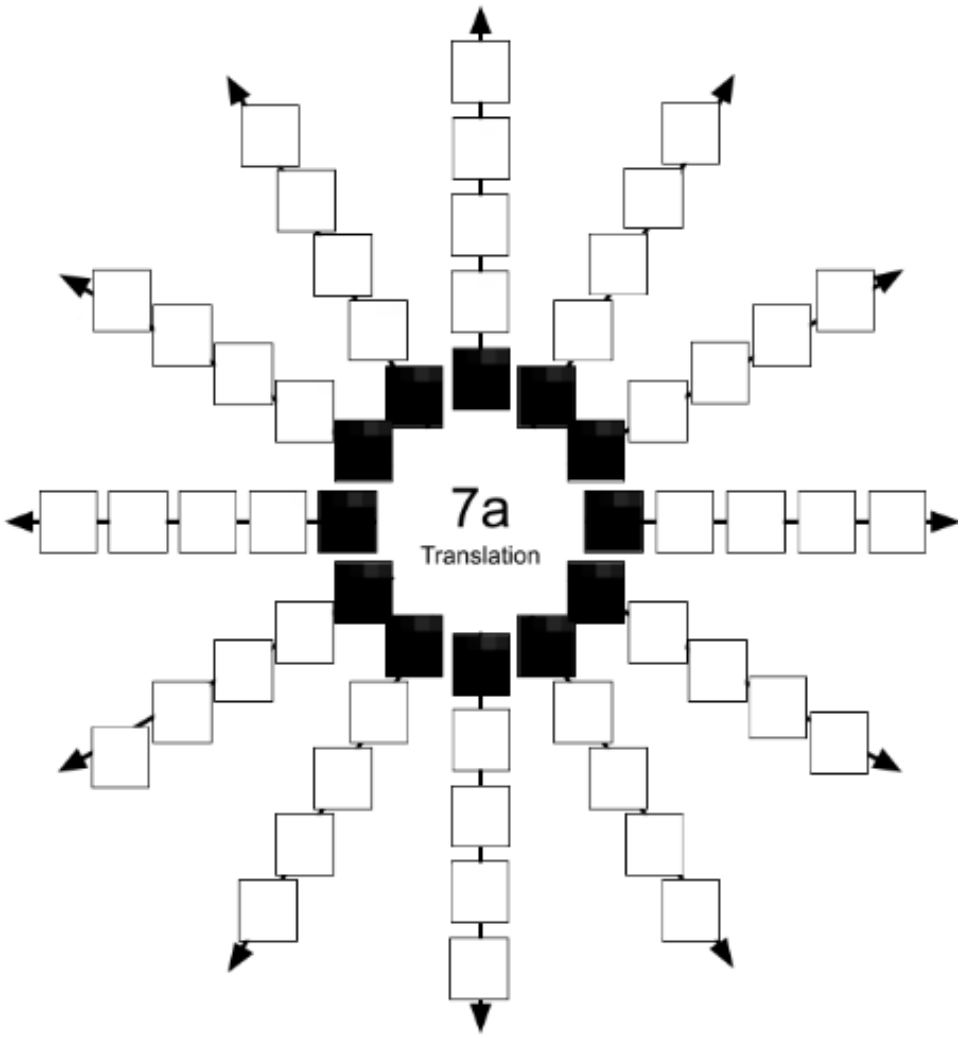

Figure A.2 (continued): This shows a close-up view of the 6-Frame model output for 7a Translation.

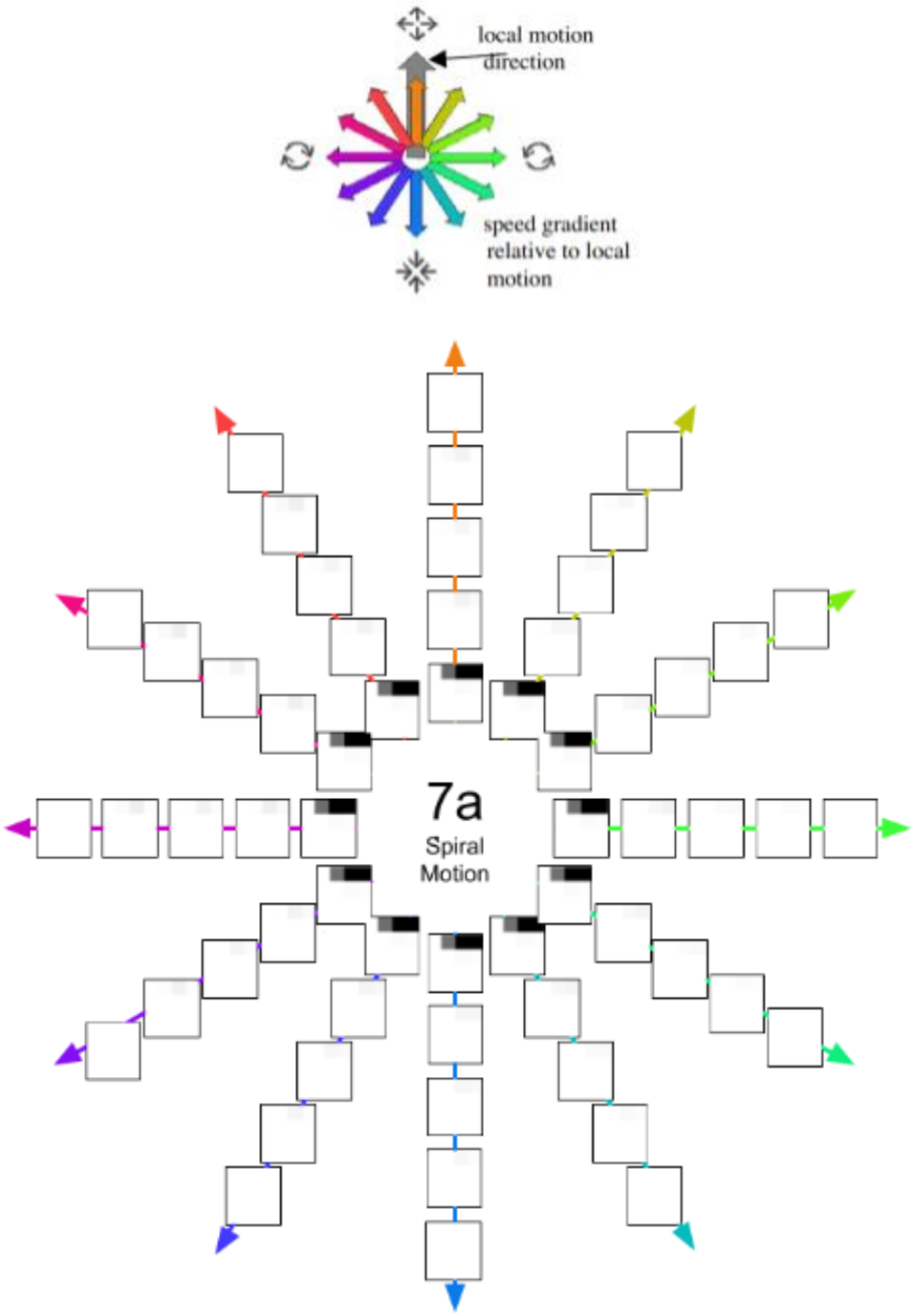

Figure A.2 (continued): 7a Spiral Motion shows the generalized spiral output in Area 7a. Coloured arrows indicate the inner angle δ between motion and speed gradient (See legend). This shows a close-up view of the 2-Frame model output for 7a Spiral Motion.

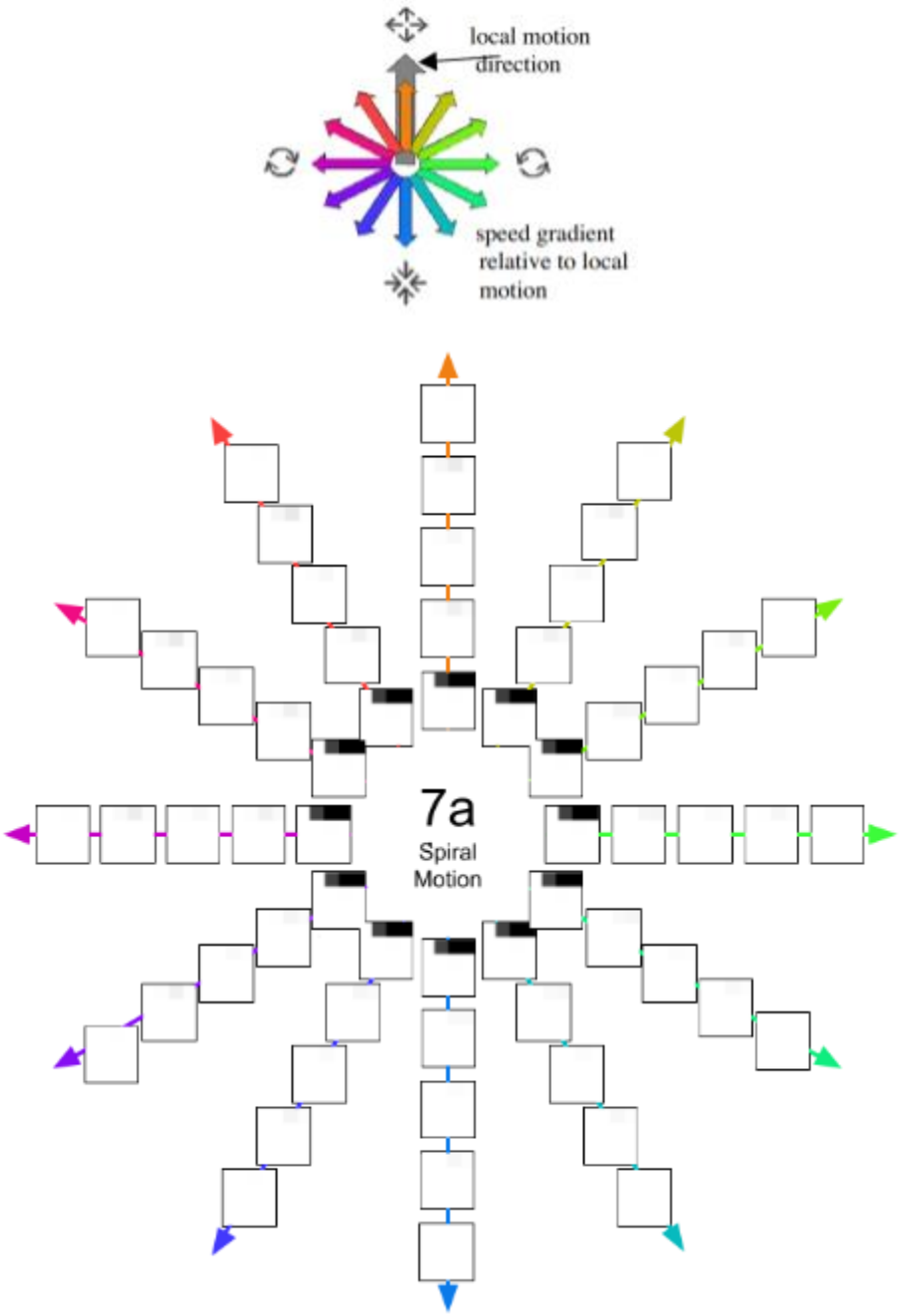

Figure A.2 (continued): This shows a close-up view of the 6-Frame model output for 7a Spiral Motion.

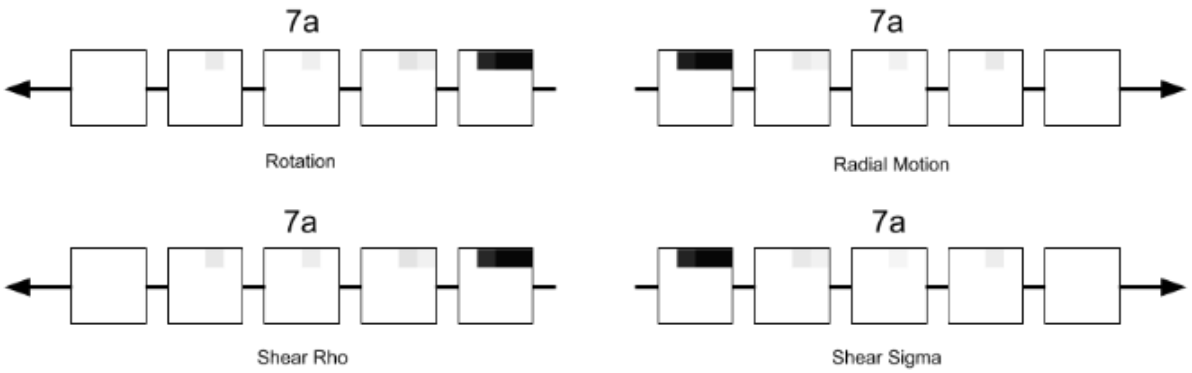

Figure A.2 (continued): Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the 2-Frame model outputs for AP.

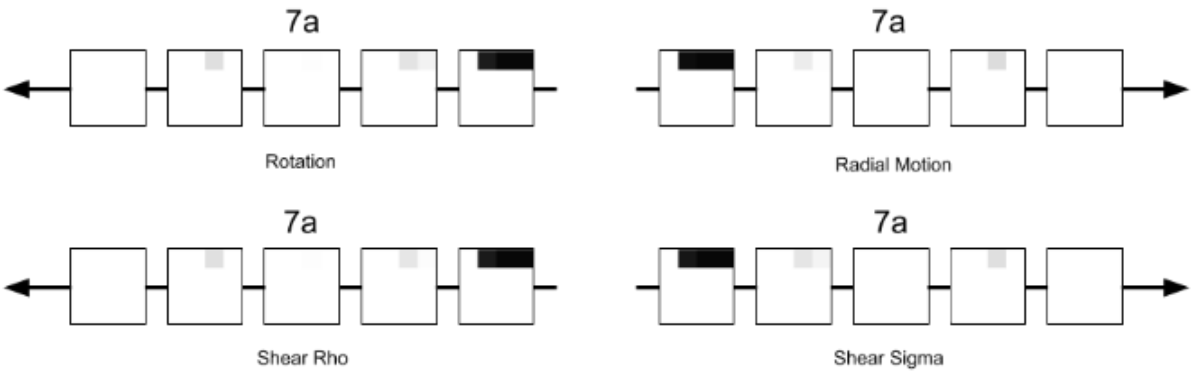

Figure A.2 (continued): This shows a close-up view of the 6-Frame model outputs for AP.

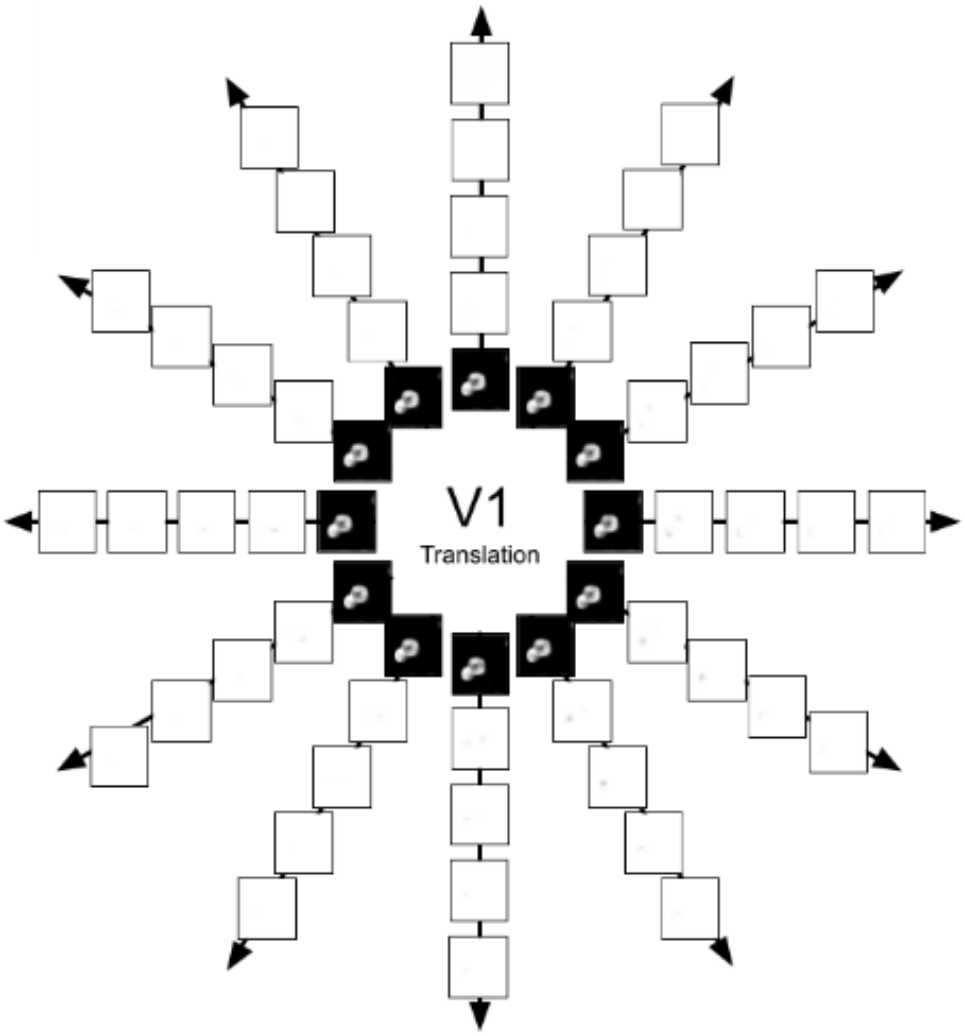

Figure A.3: Close-up view of the output of area computations of ST-Motion-Net for sequence in Figure 3.8. V1 Translation shows the translation output of Area V1. This shows a close-up view of the 2-Frame model output for V1 Translation.

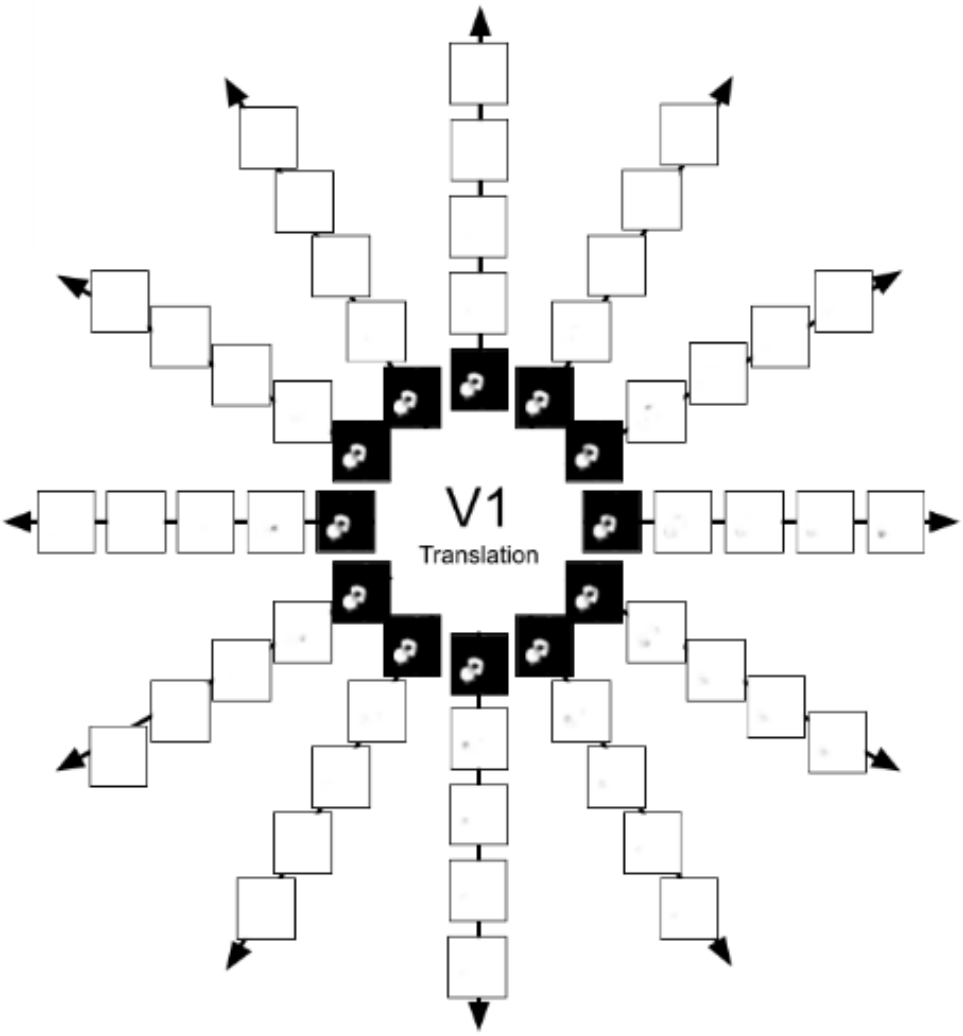

Figure A.3 (continued): This shows a close-up view of the 6-Frame model output for V1 Translation.

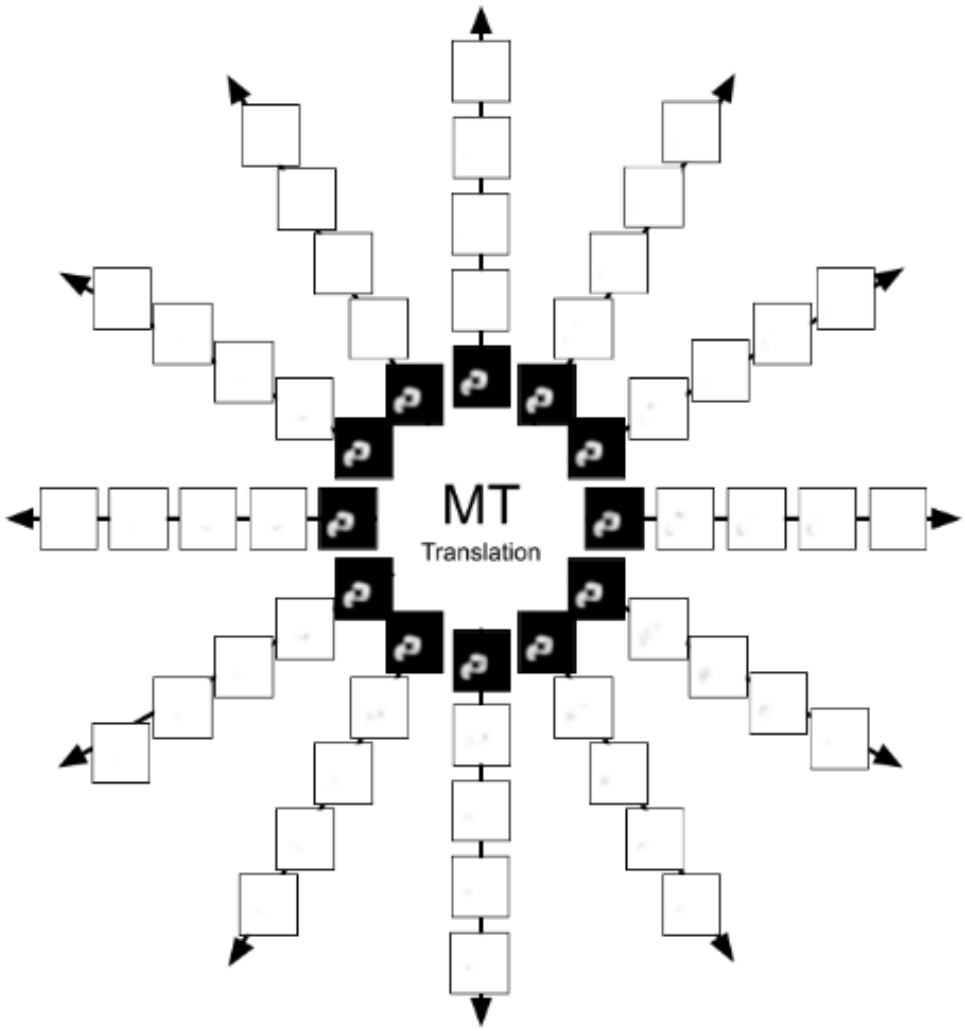

Figure A.3 (continued): MT Translation shows the translation output of Area MT. This shows a close-up view of the 2-Frame model output for MT Translation.

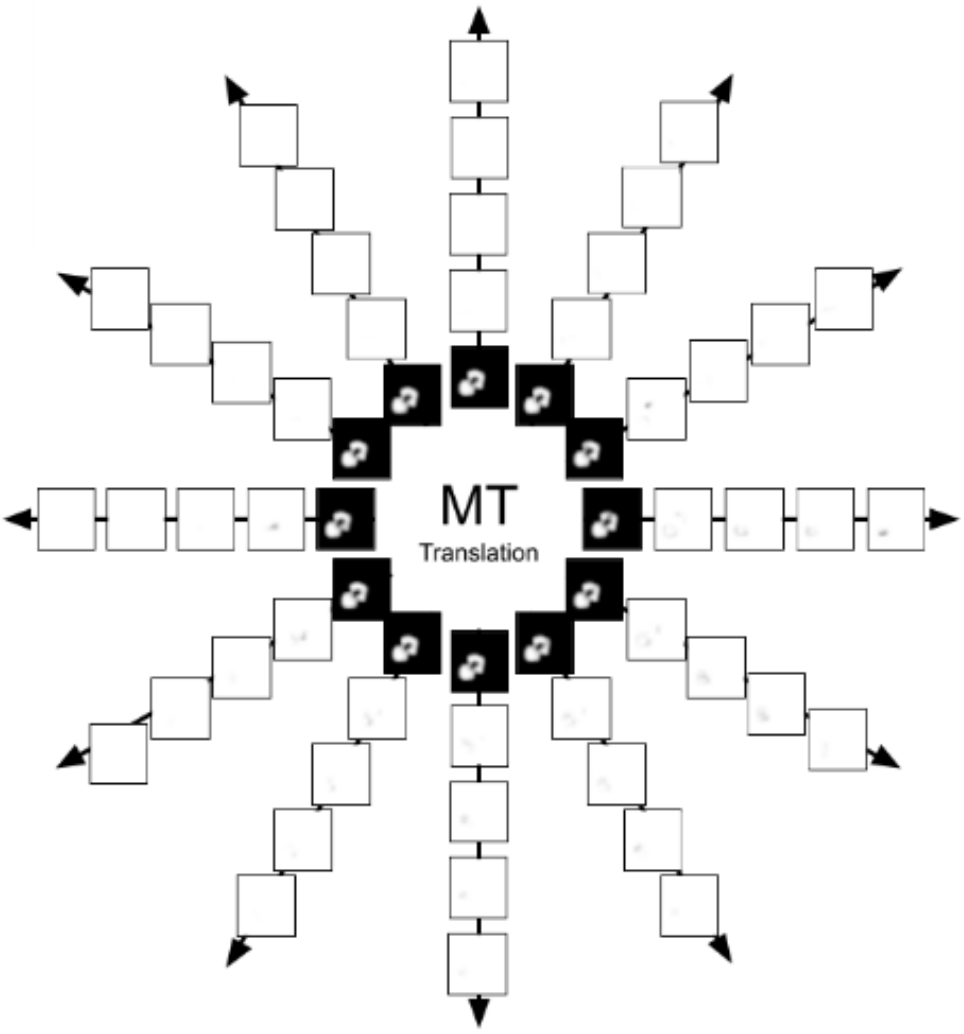

Figure A.3 (continued): This shows a close-up view of the 6-Frame model output for MT Translation.

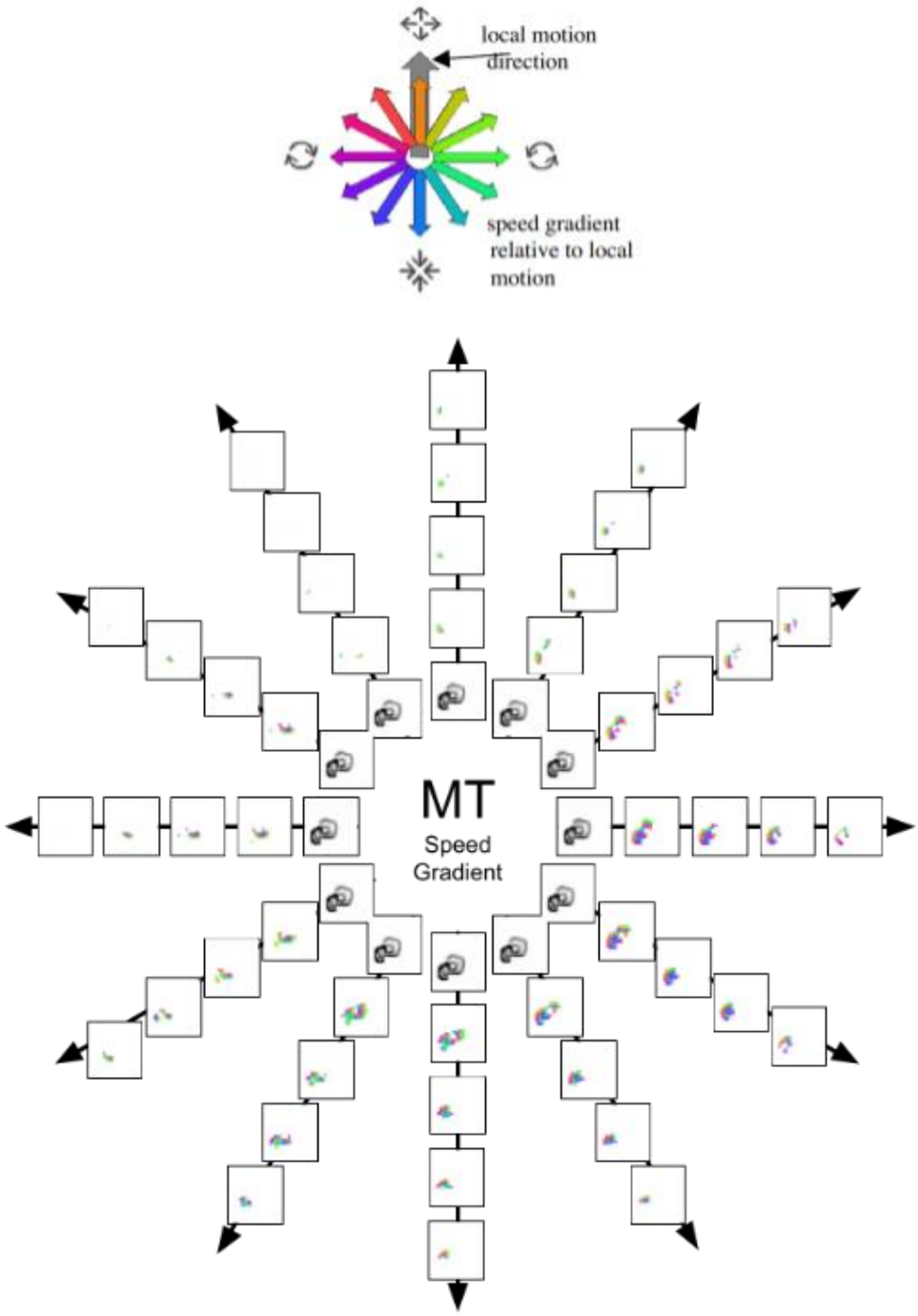

Figure A.3 (continued): Speed gradient outputs in Area MT are shown in a summary representation of the 12 speed gradients for each local direction and speed. Each coloured dot is the maximum value across the 12 representations. This shows a close-up view of the 2-Frame model output for MT Speed Gradients in the summary representation.

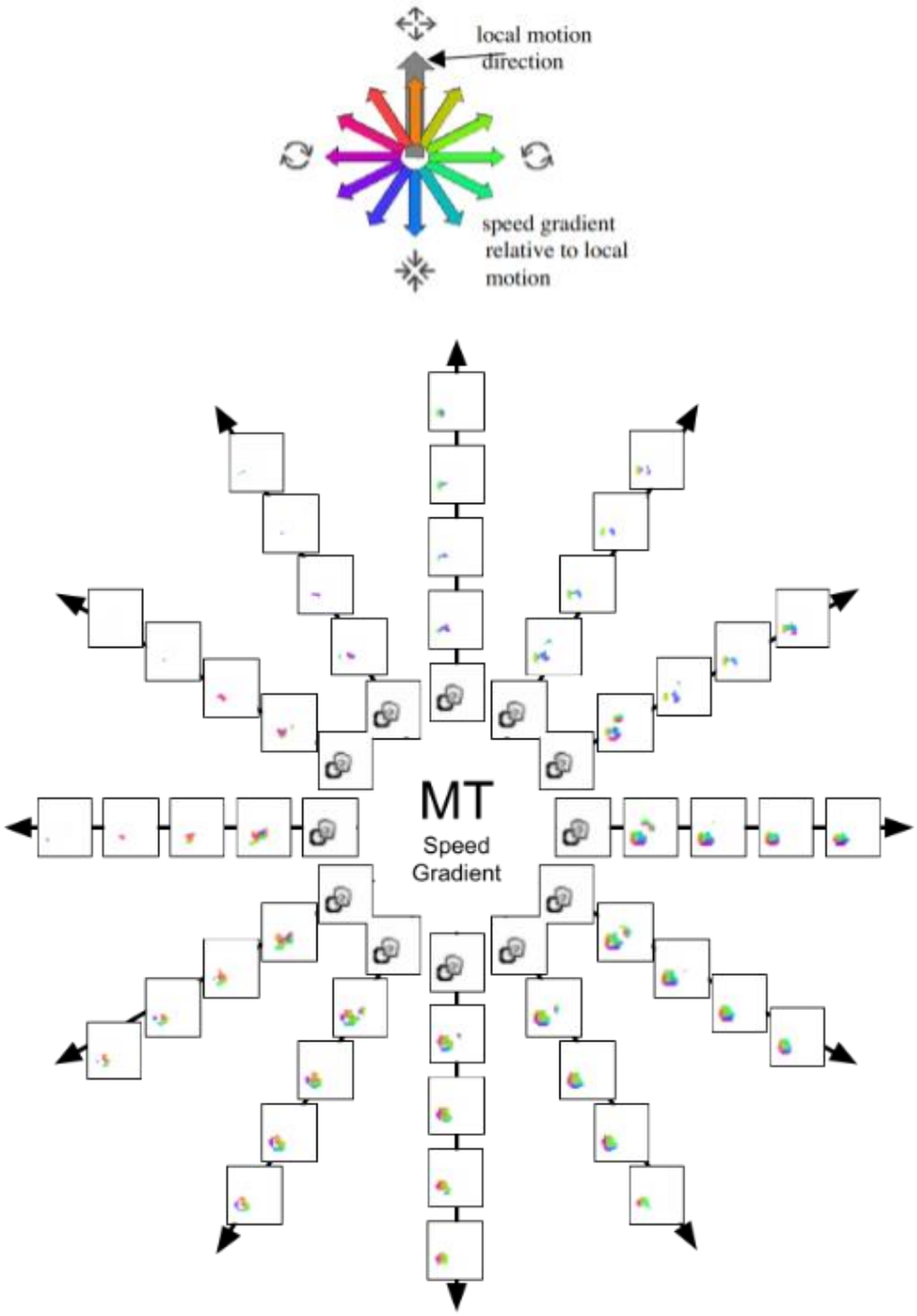

Figure A.3 (continued): This shows a close-up view of the 6-Frame model output for MT Speed Gradients in the summary representation.

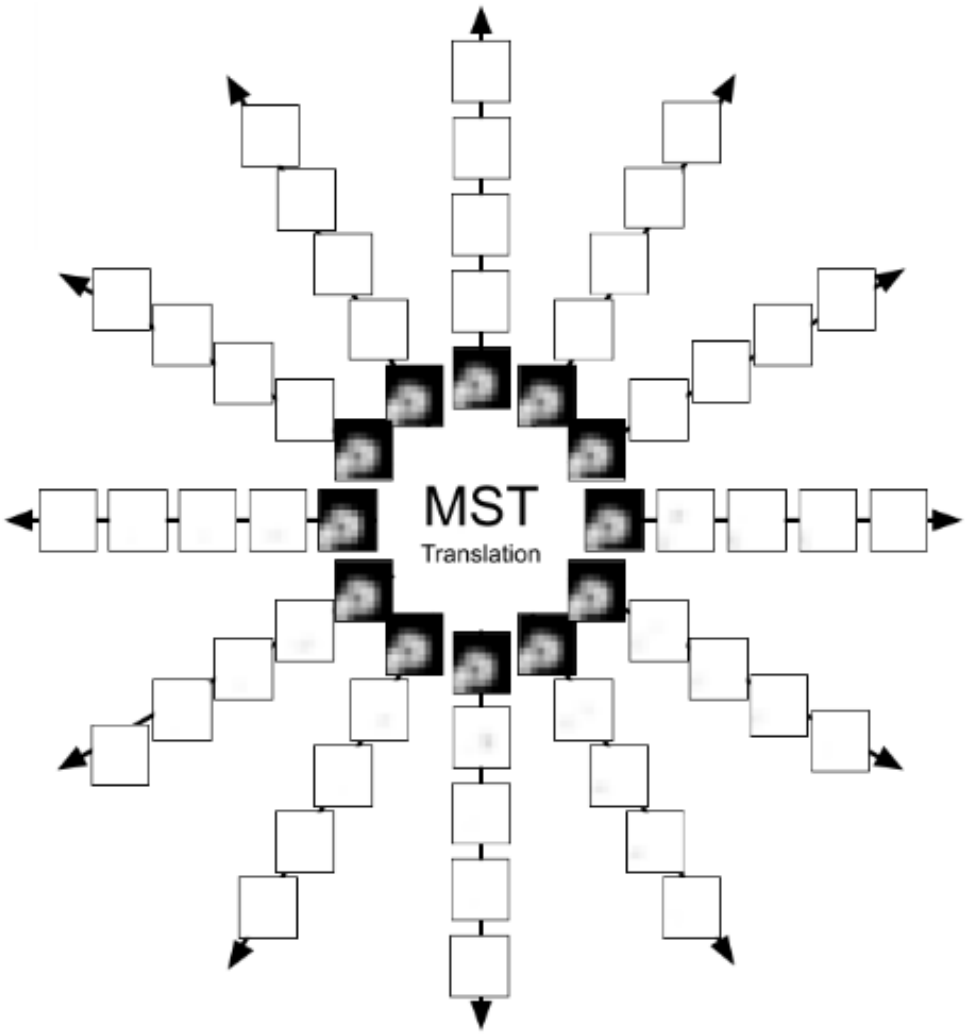

Figure A.3 (continued): MST Translation shows the translation output in Area MST. This shows a close-up view of the 2-Frame model output for MST Translation.

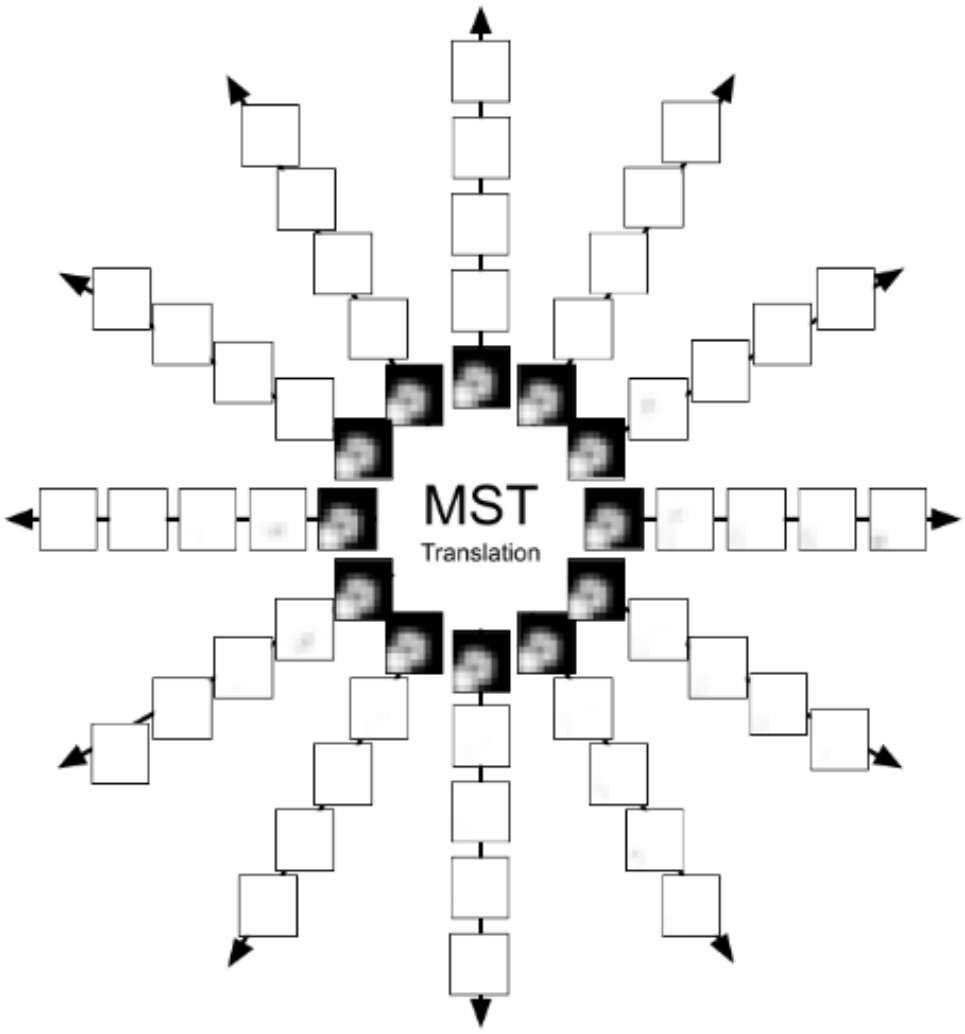

Figure A.3 (continued): This shows a close-up view of the 6-Frame model output for MST Translation.

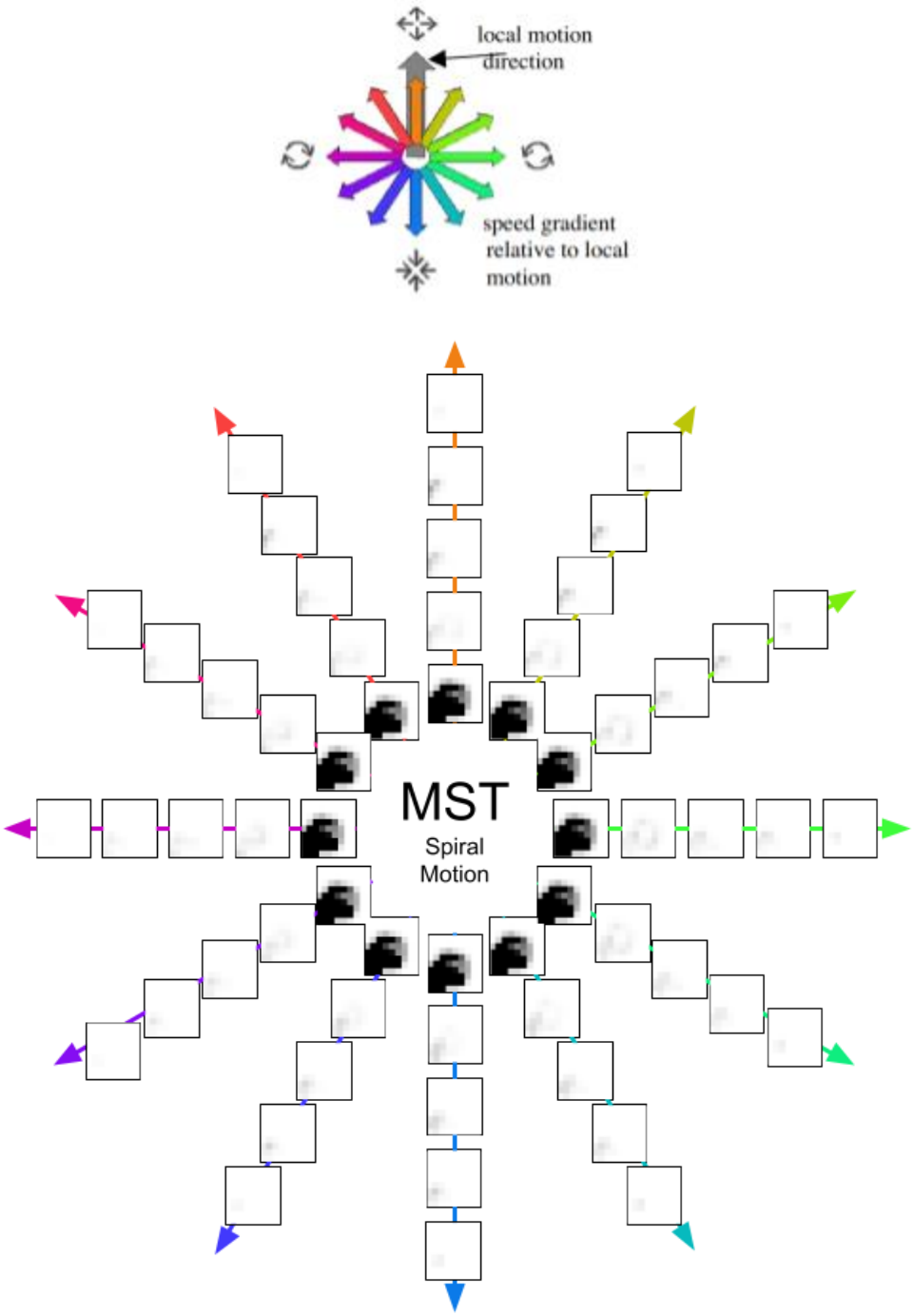

Figure A.3 (continued): MST Spiral Motion shows the generalized spiral output in Area MST. Coloured arrows indicate the inner angle δ between motion and speed gradient (See legend). This shows a close-up view of the 2-Frame model output for MST Spiral Motion.

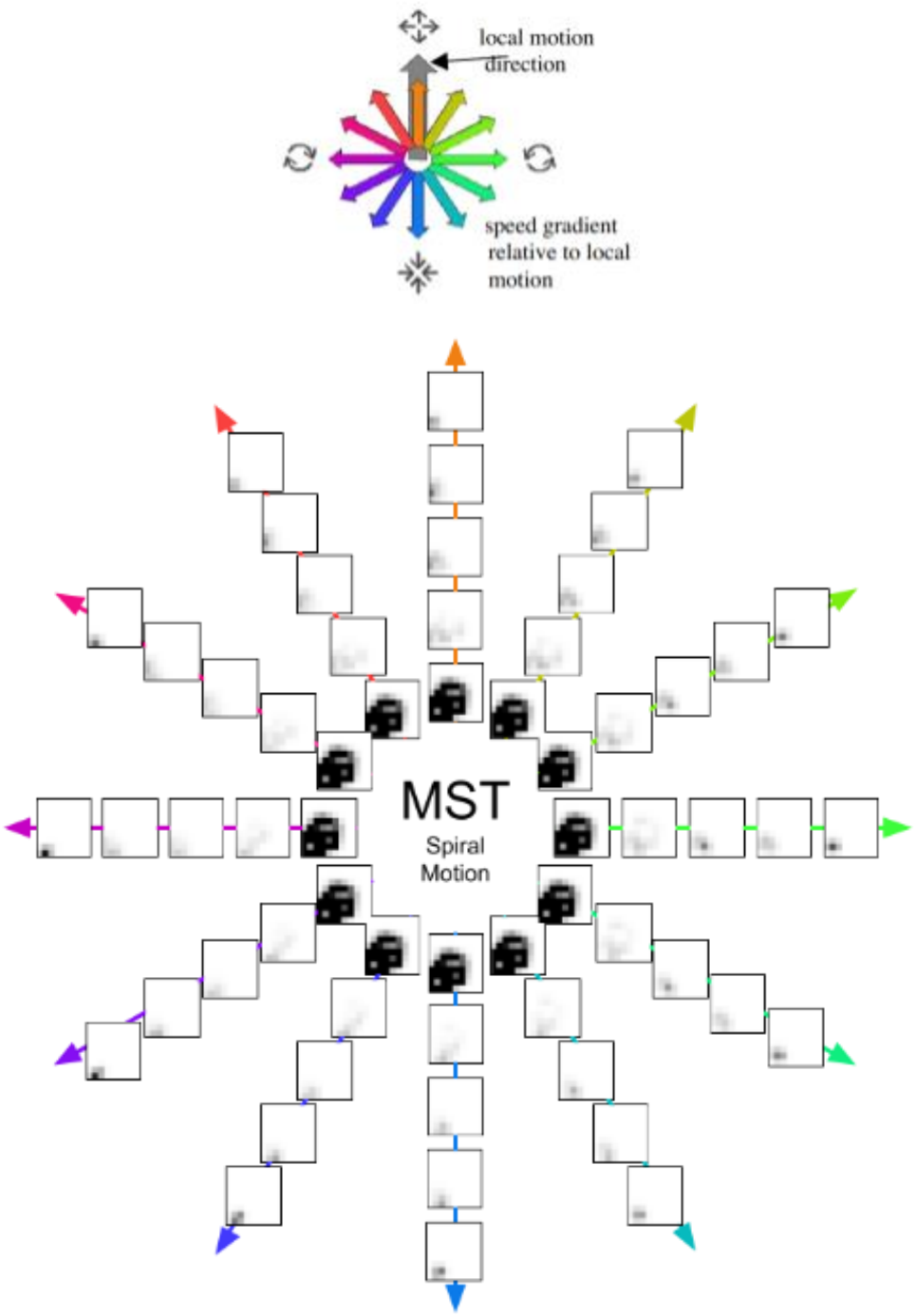

Figure A.3 (continued): This shows a close-up view of the 6-Frame model output for MST Spiral Motion.

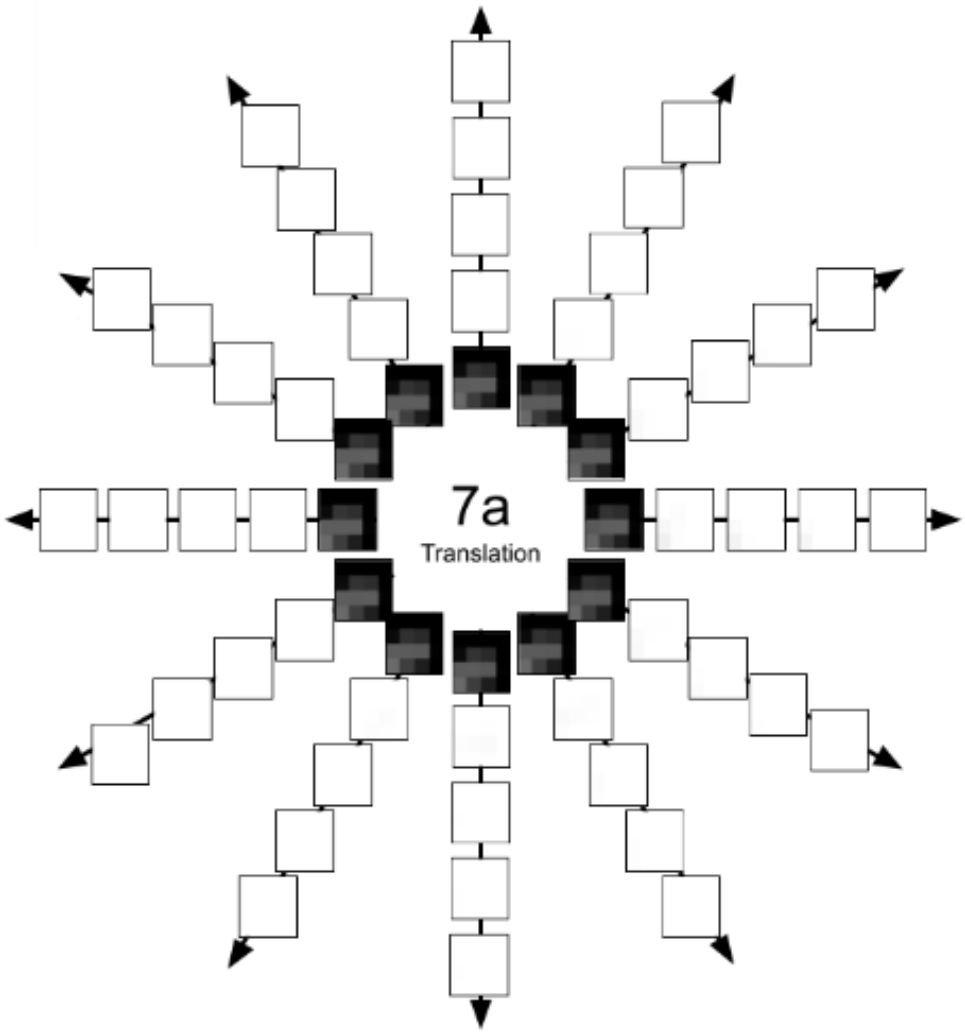

Figure A.3 (continued): 7a Translation shows the translation output in Area 7a. This shows a close-up view of the 2-Frame model output for 7a Translation.

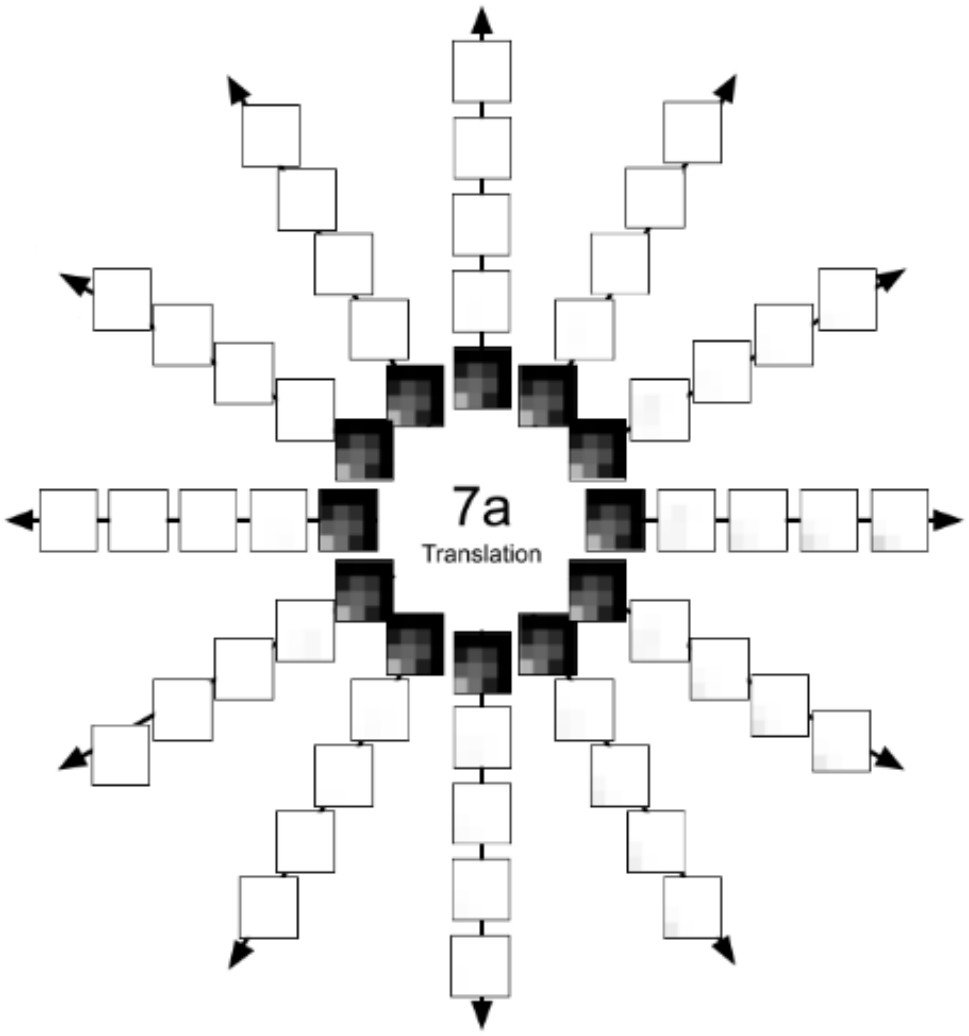

Figure A.3 (continued): This shows a close-up view of the 6-Frame model output for 7a Translation.

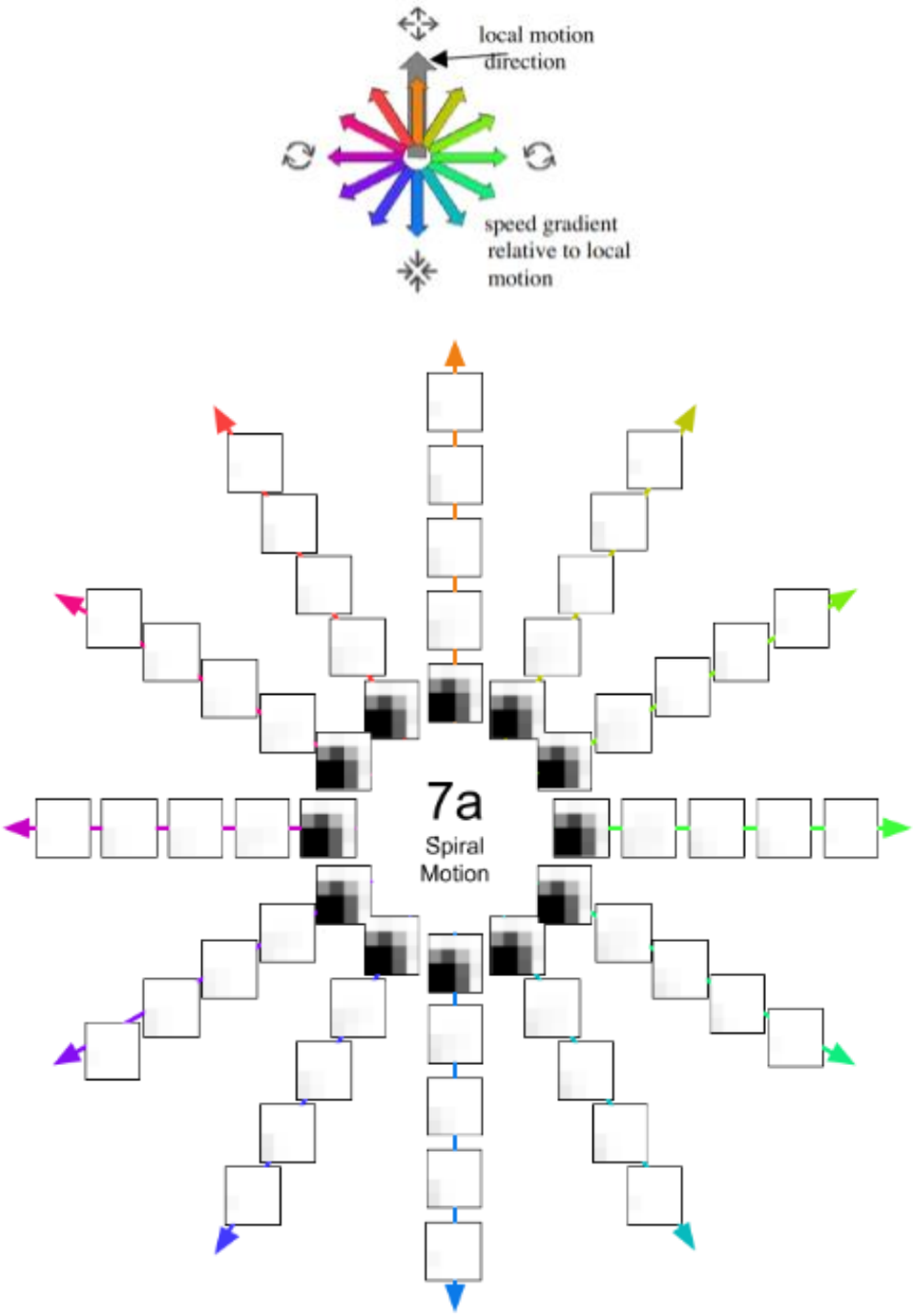

Figure A.3 (continued): 7a Spiral Motion shows the generalized spiral output in Area 7a. Coloured arrows indicate the inner angle δ between motion and speed gradient (See legend). This shows a close-up view of the 2-Frame model output for 7a Spiral Motion.

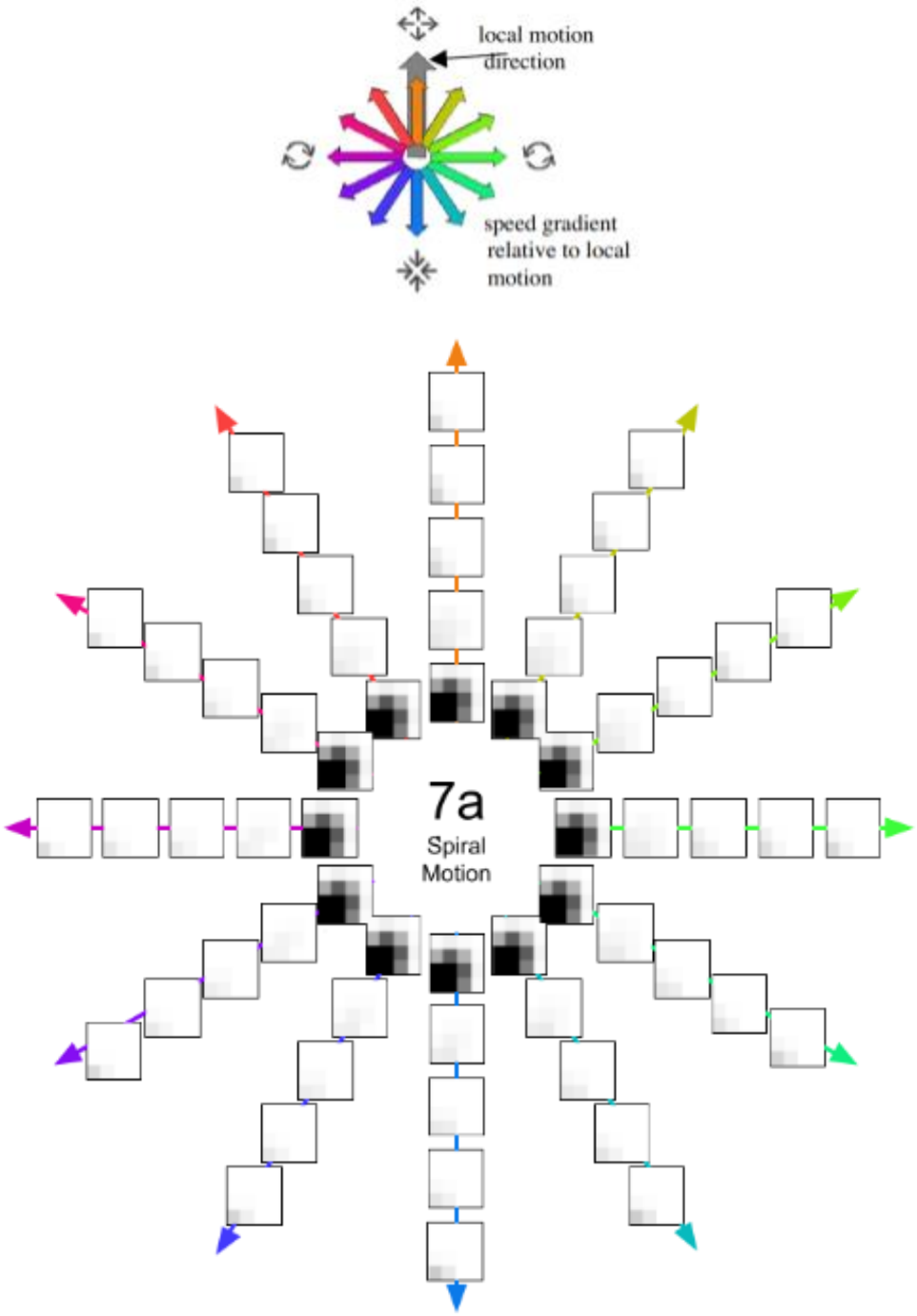

Figure A.3 (continued): This shows a close-up view of the 6-Frame model output for 7a Spiral Motion.

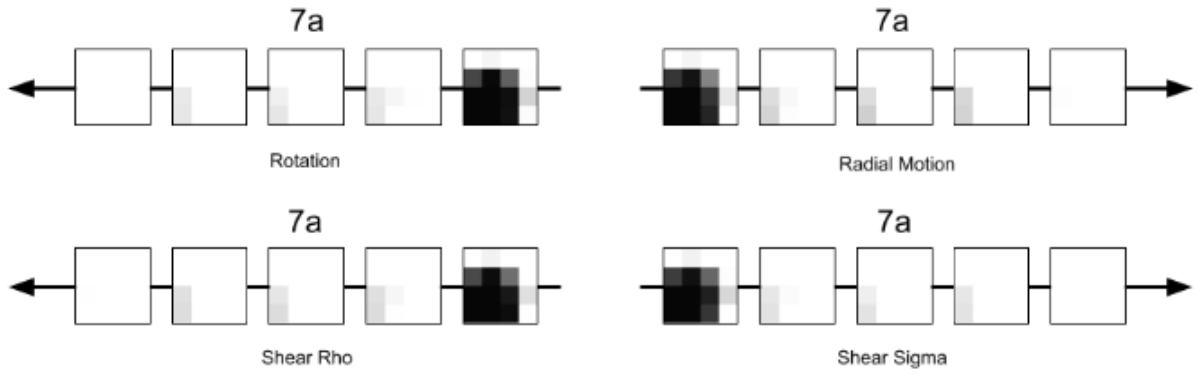

Figure A.3 (continued): Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the 2-Frame model outputs for AP.

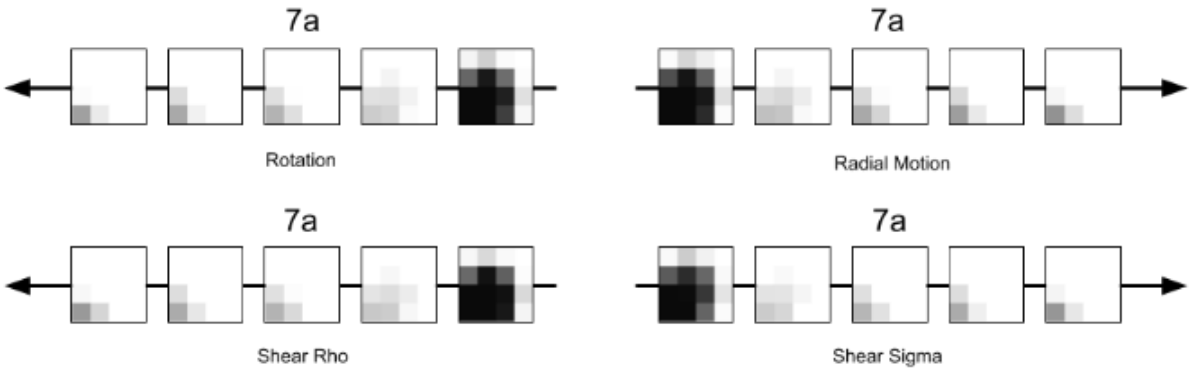

Figure A.3 (continued): This shows a close-up view of the 6-Frame model outputs for AP.

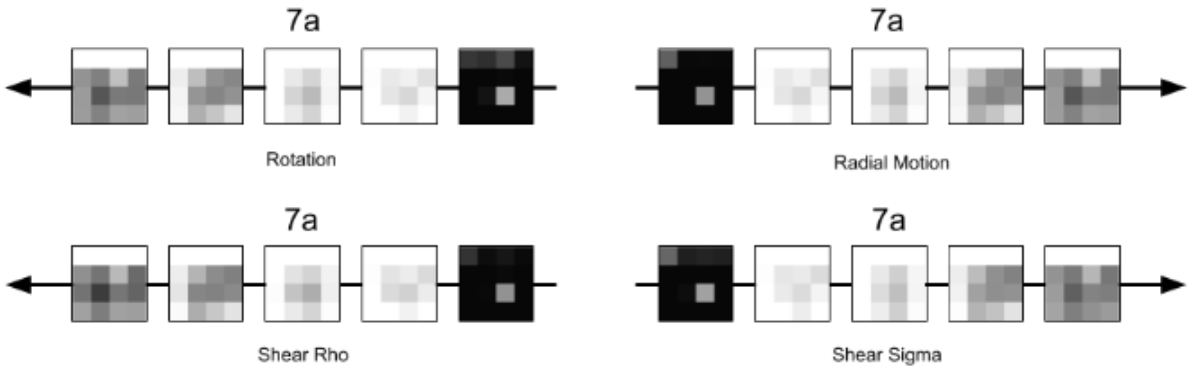

Figure A.4: Close-up view of the output of area computations of AP of ST-Motion-Net for sequence in Figure 3.10. Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the 2-Frame model outputs for AP.

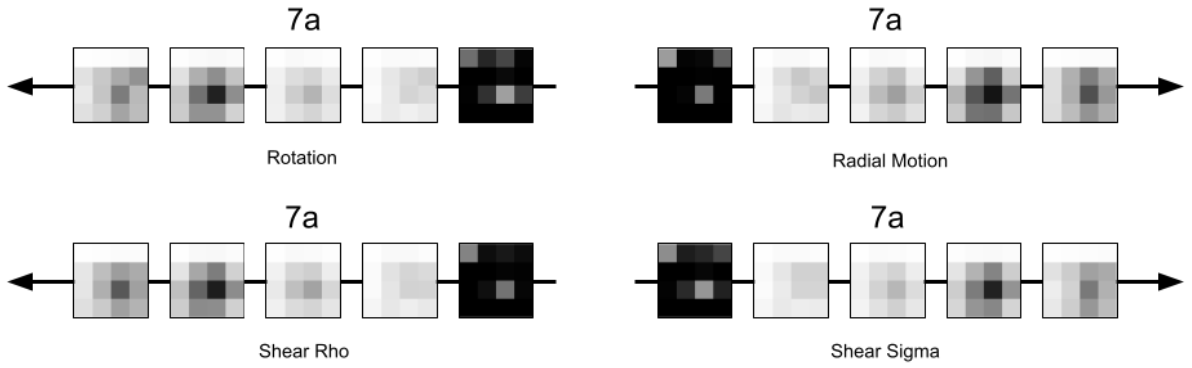

Figure A.4 (continued): This shows a close-up view of the 6-Frame model outputs for AP.

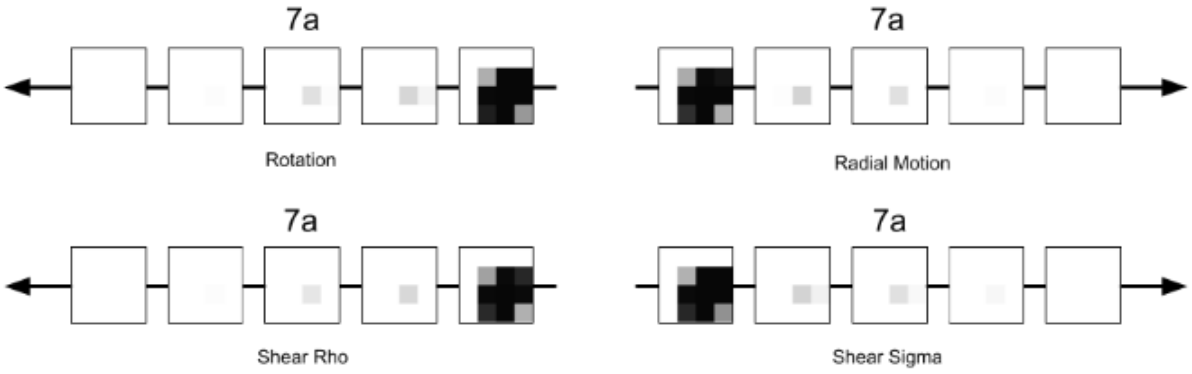

Figure A.5: Close-up view of the output of area computations of AP of ST-Motion-Net for sequence in Figure 3.12. Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the 2-Frame model outputs for AP.

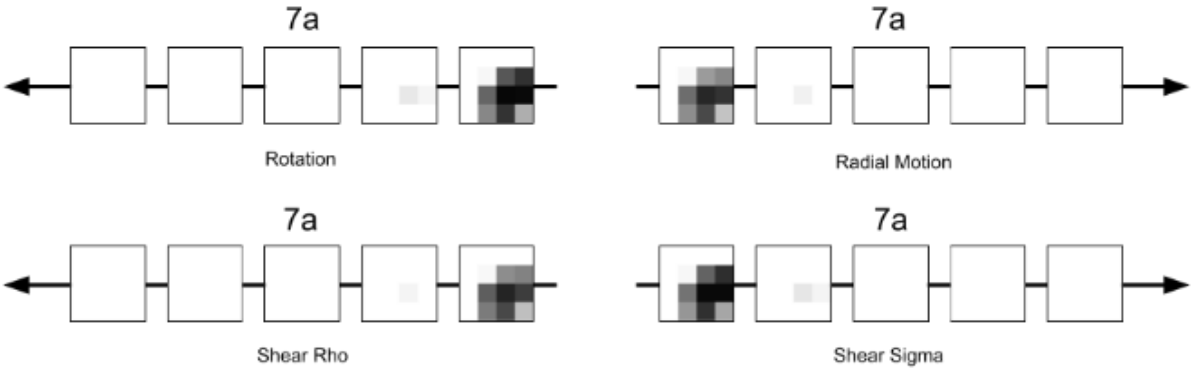

Figure A.5 (continued): This shows a close-up view of the 6-Frame model outputs for AP.

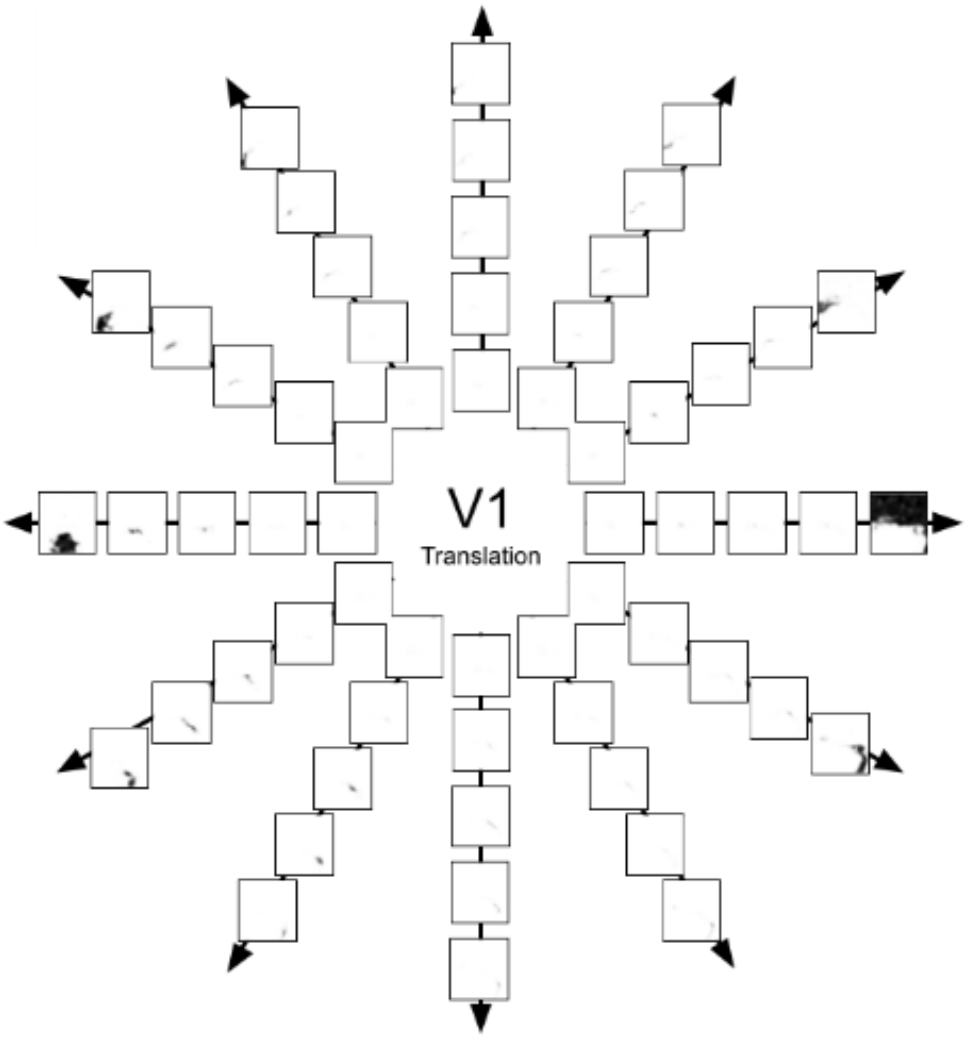

Figure A.6: Close-up view of the output of area computations of ST-Motion-Net for sequence in Figure 6.1. V1 Translation shows the translation output of Area V1. This shows a close-up view of the 2-Frame model output for V1 Translation.

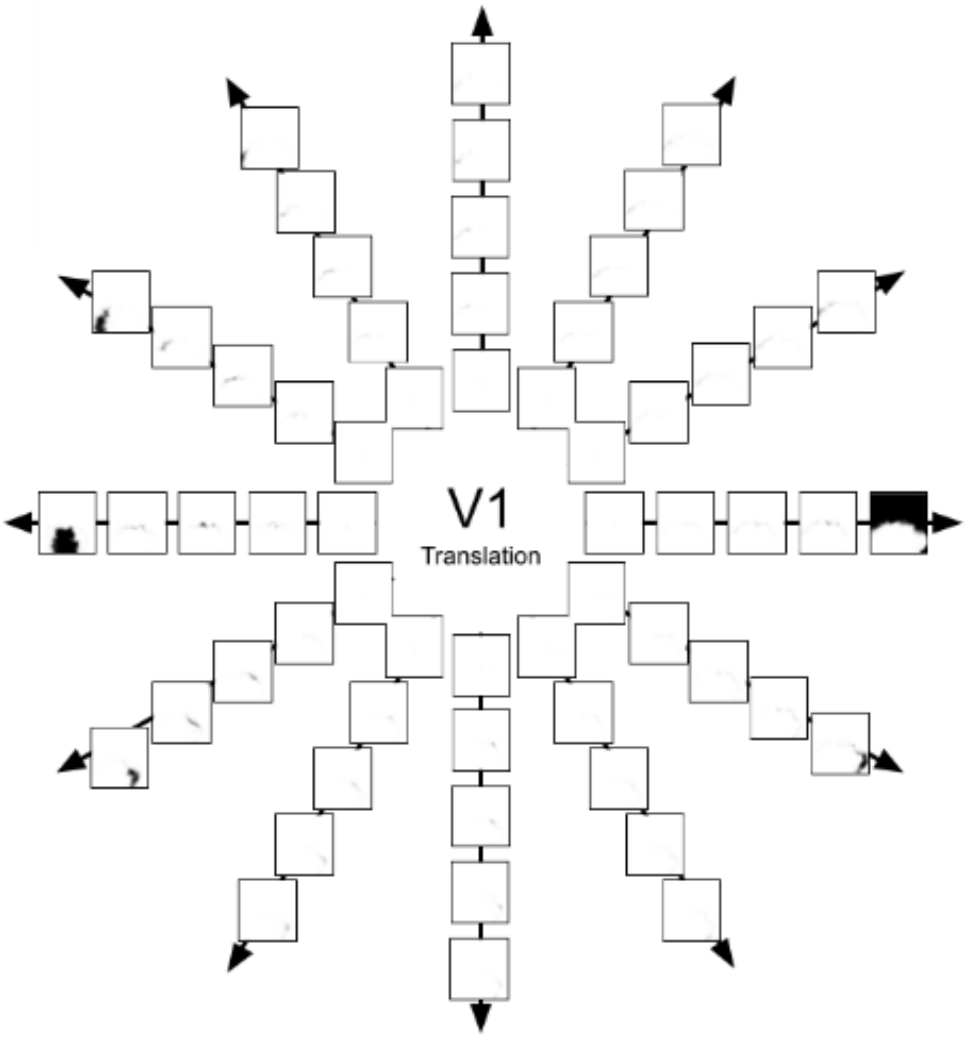

Figure A.6 (continued): This shows a close-up view of the 6-Frame model output for V1 Translation.

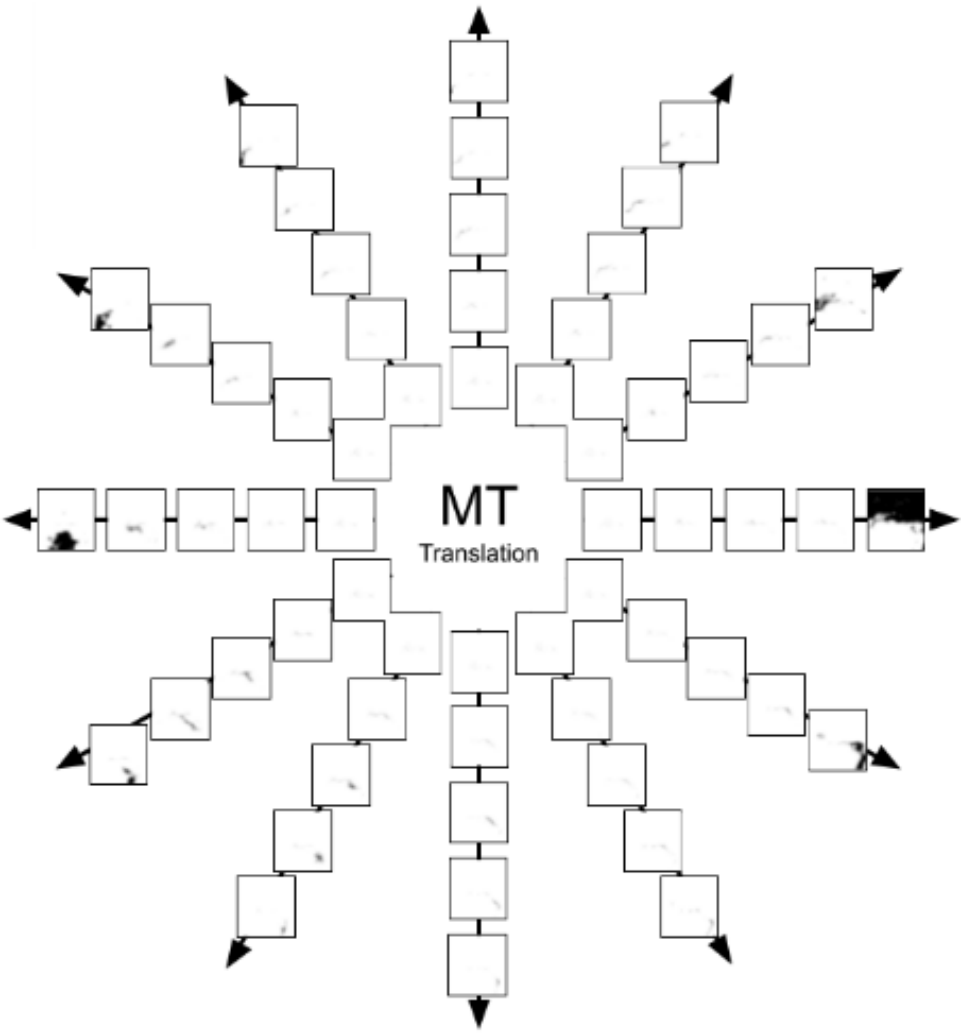

Figure A.6 (continued): MT Translation shows the translation output of Area MT. This shows a close-up view of the 2-Frame model output for MT Translation.

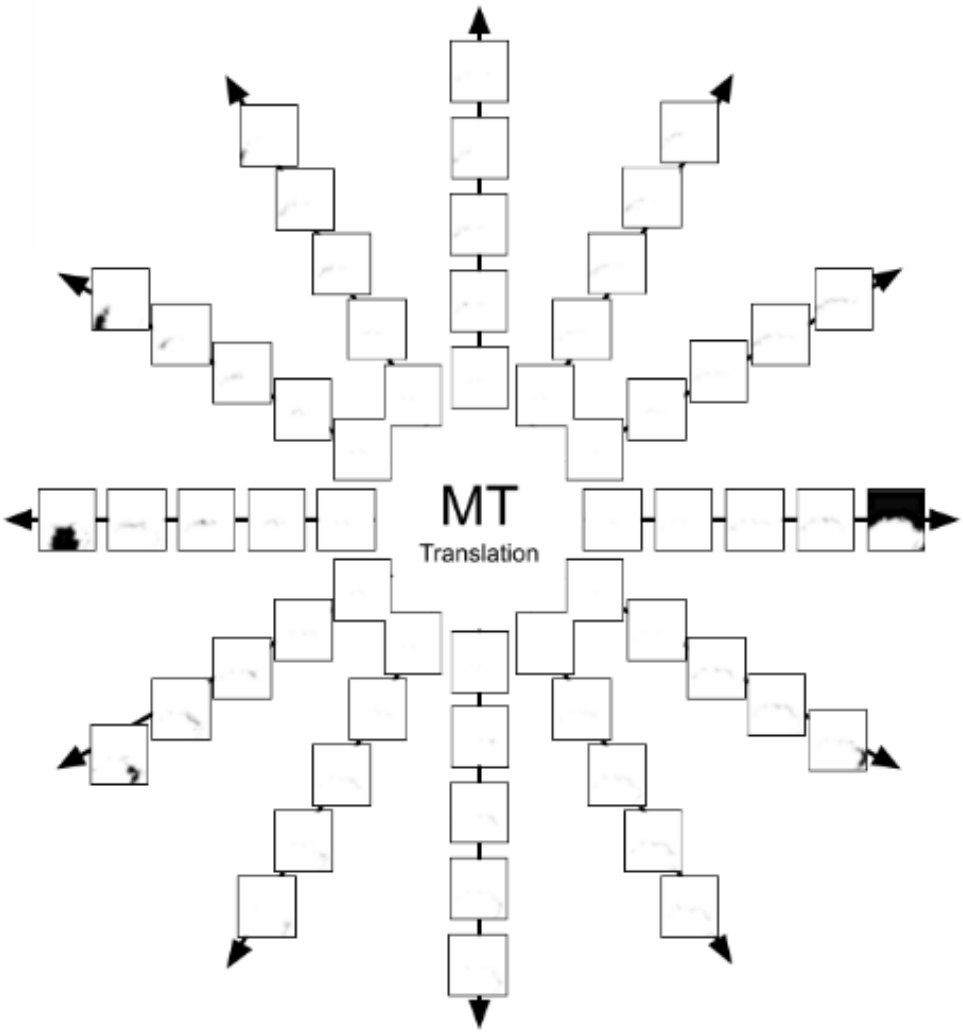

Figure A.6 (continued): This shows a close-up view of the 6-Frame model output for MT Translation.

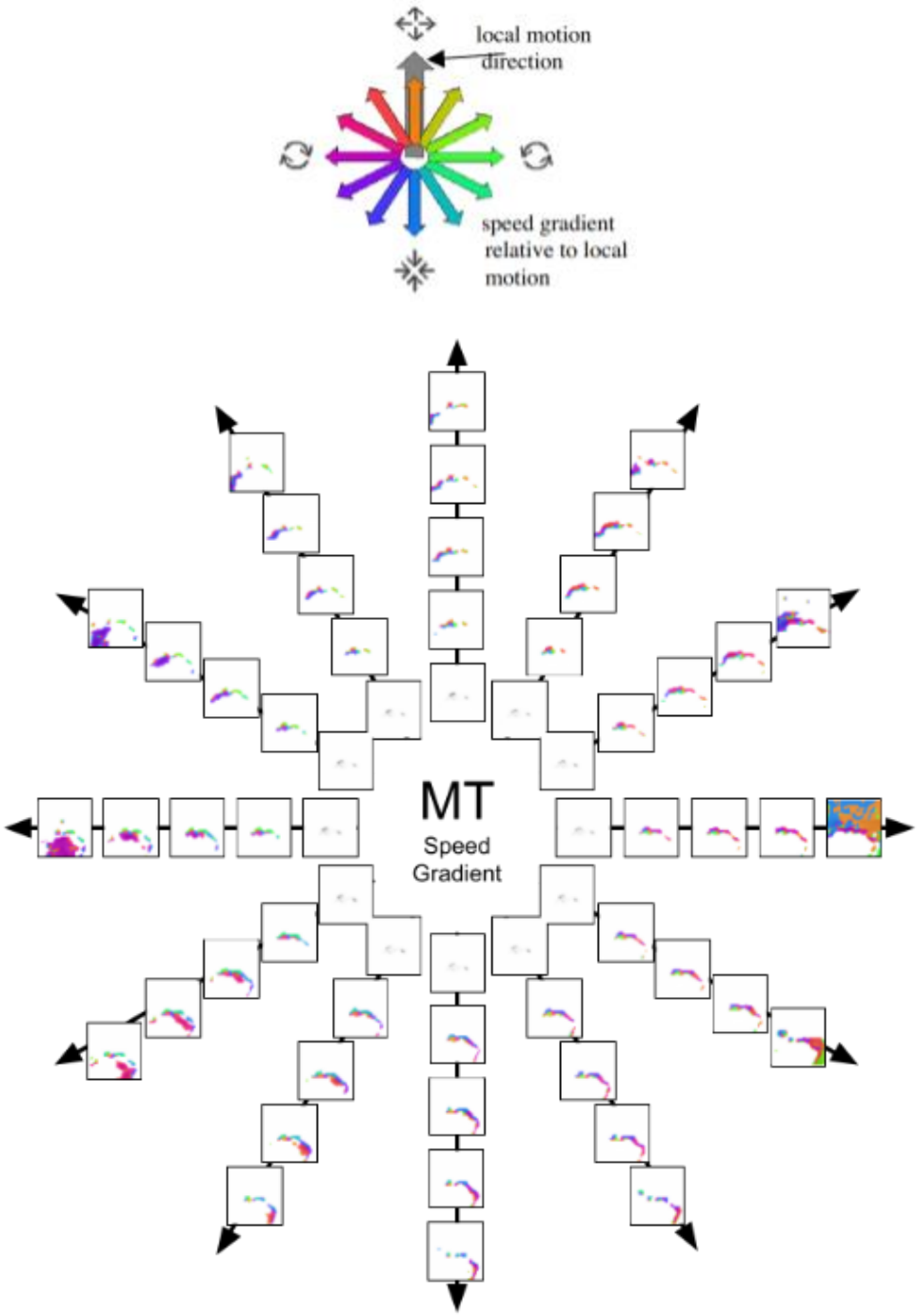

Figure A.6 (continued): Speed gradient outputs in Area MT are shown in a summary representation of the 12 speed gradients for each local direction and speed. Each coloured dot is the maximum value across the 12 representations. This shows a close-up view of the 2-Frame model output for MT Speed Gradients in the summary representation.

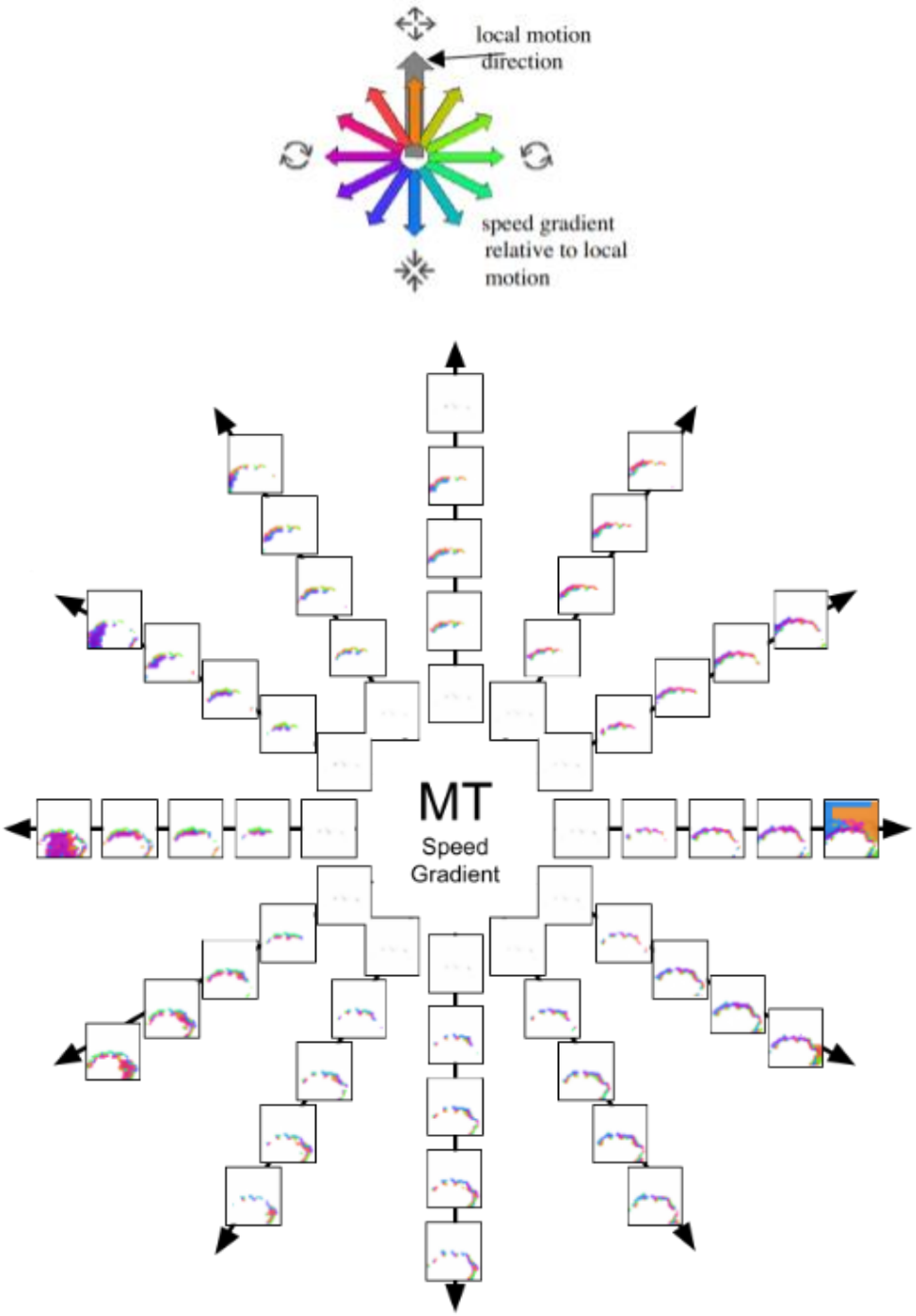

Figure A.6 (continued): This shows a close-up view of the 6-Frame model output for MT Speed Gradients in the summary representation.

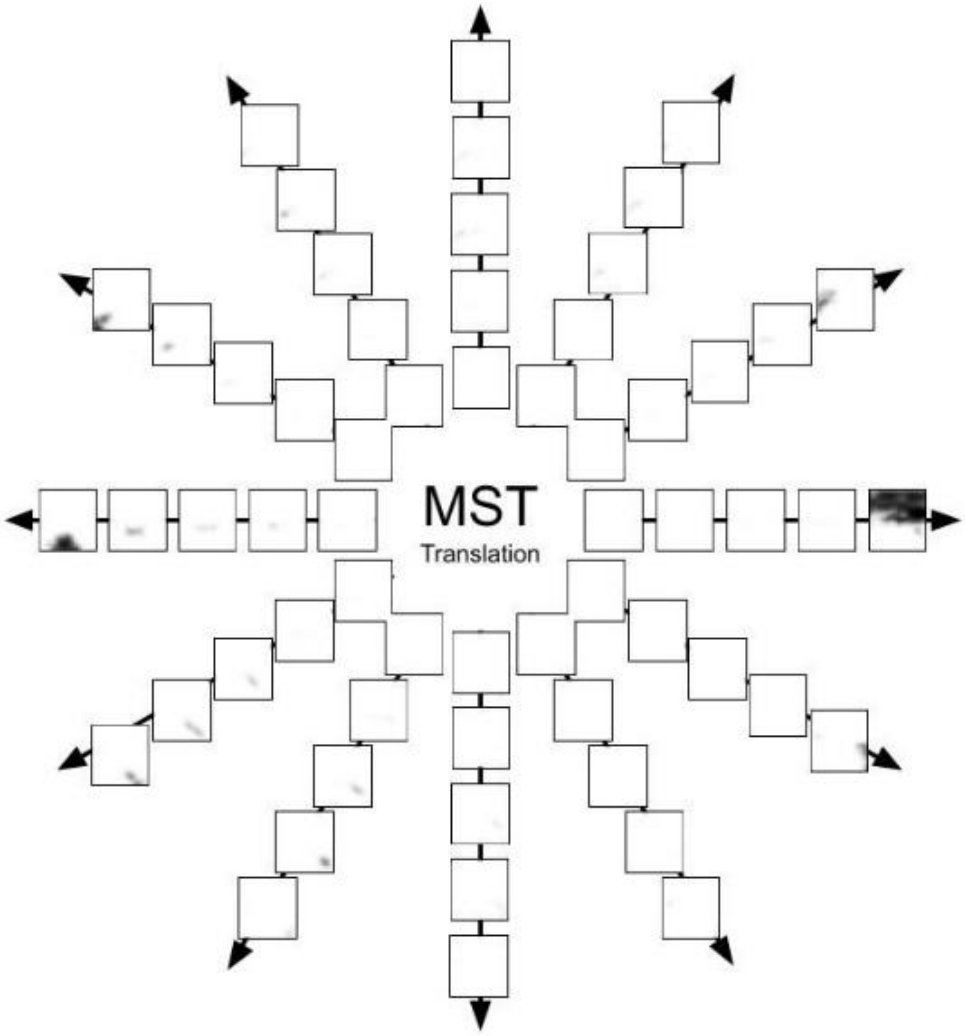

Figure A.6 (continued): MST Translation shows the translation output in Area MST. This shows a close-up view of the 2-Frame model output for MST Translation.

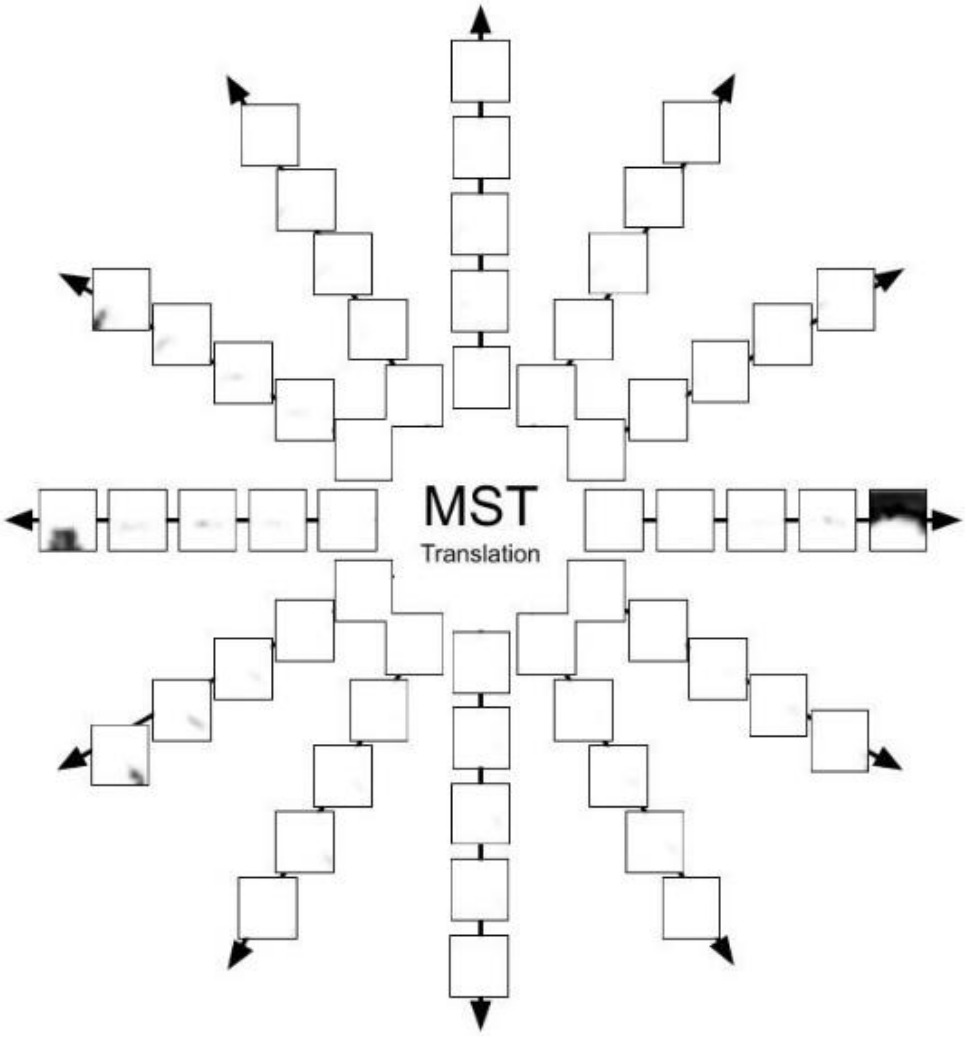

Figure A.6 (continued): This shows a close-up view of the 6-Frame model output for MST Translation.

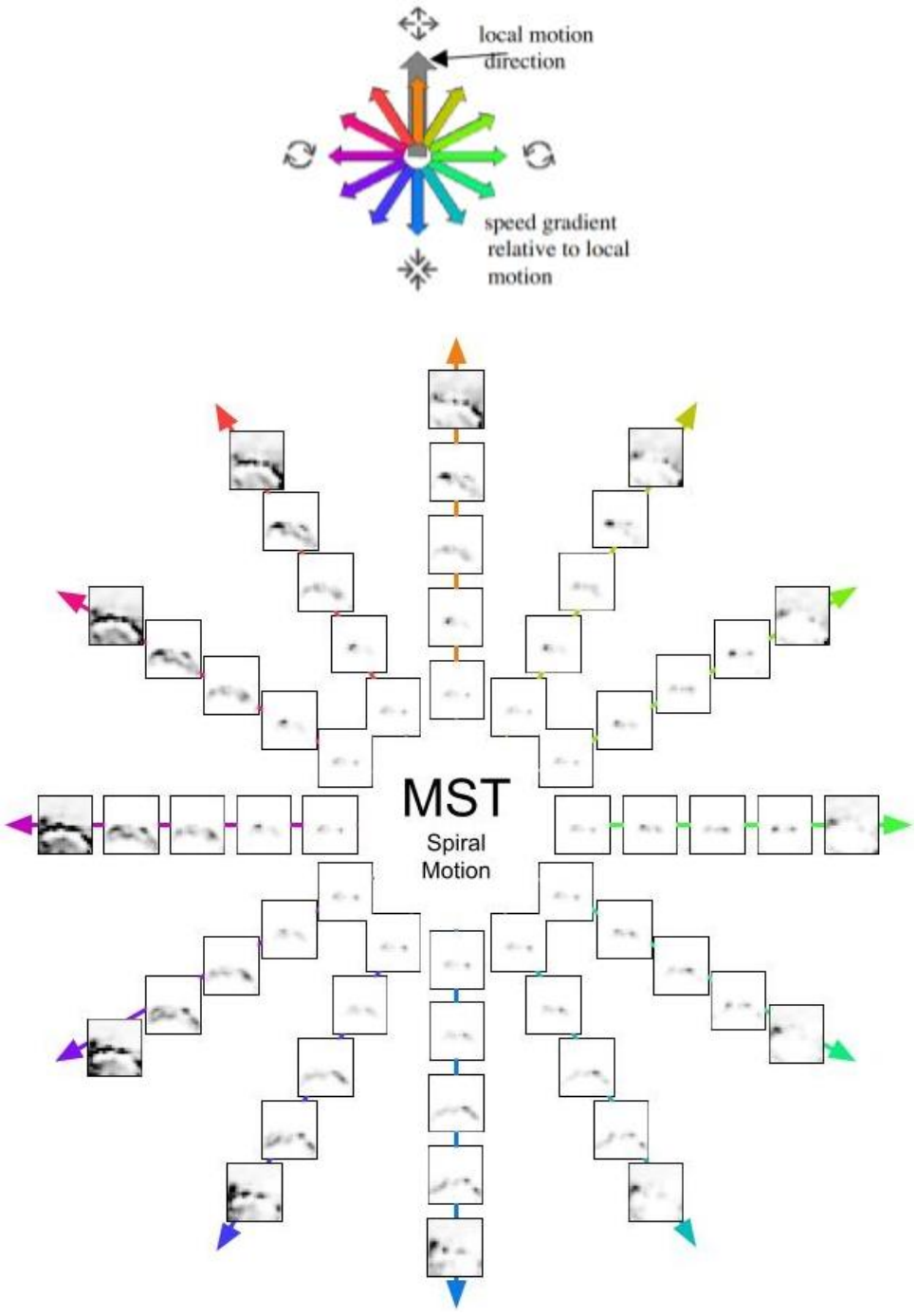

Figure A.6 (continued): MST Spiral Motion shows the generalized spiral output in Area MST. Coloured arrows indicate the inner angle δ between motion and speed gradient (See legend). This shows a close-up view of the 2-Frame model output for MST Spiral Motion.

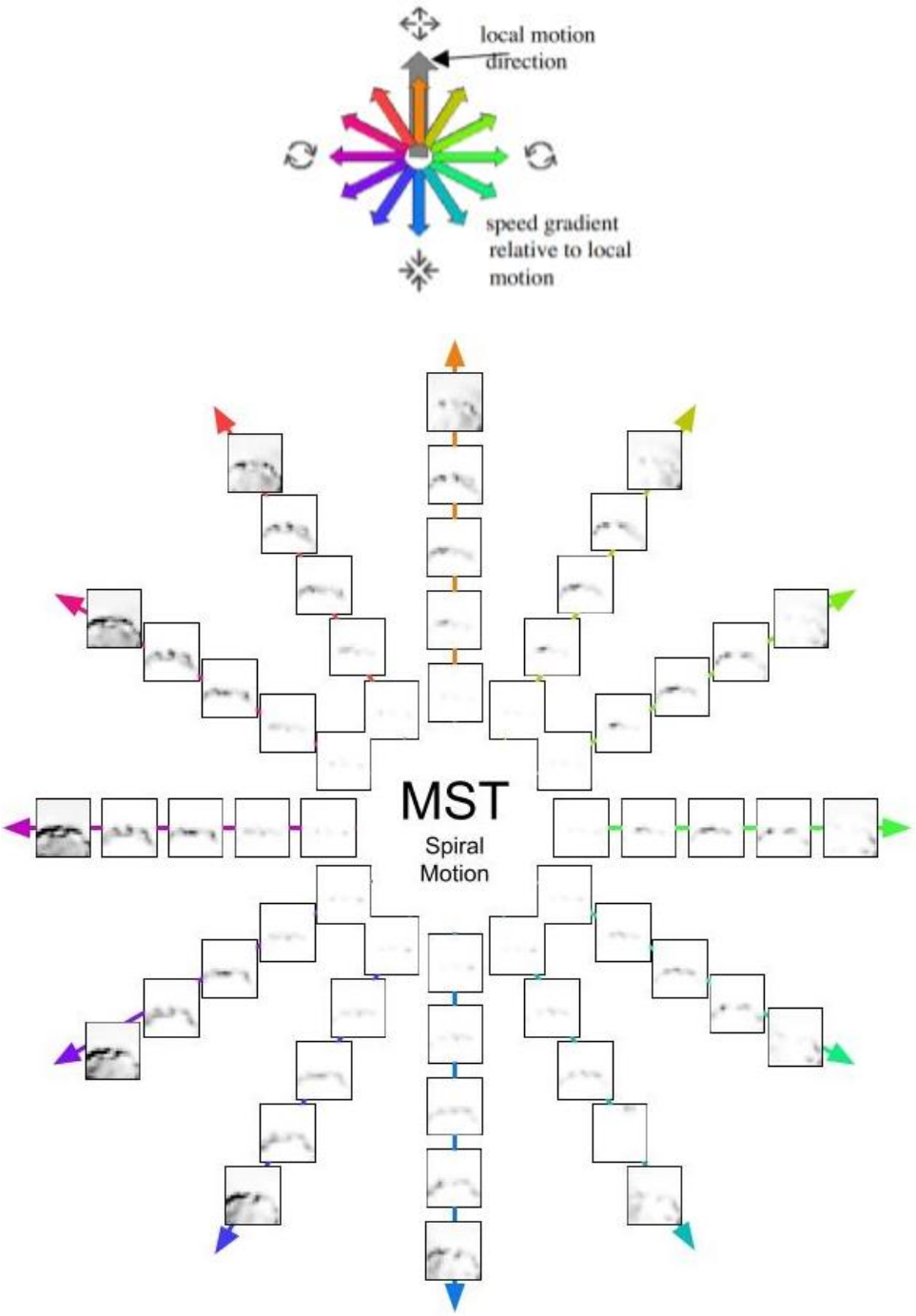

Figure A.6 (continued): This shows a close-up view of the 6-Frame model output for MST Spiral Motion.

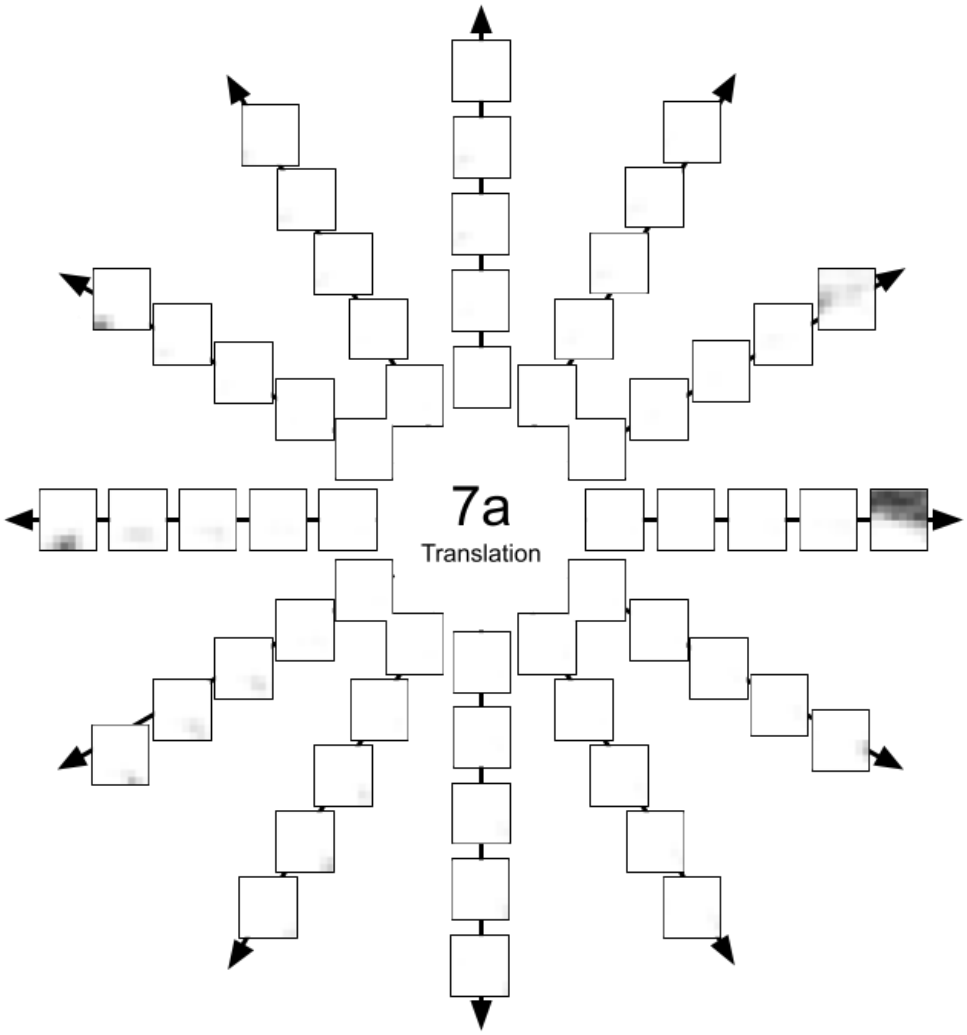

Figure A.6 (continued): 7a Translation shows the translation output in Area 7a. This shows a close-up view of the 2-Frame model output for 7a Translation.

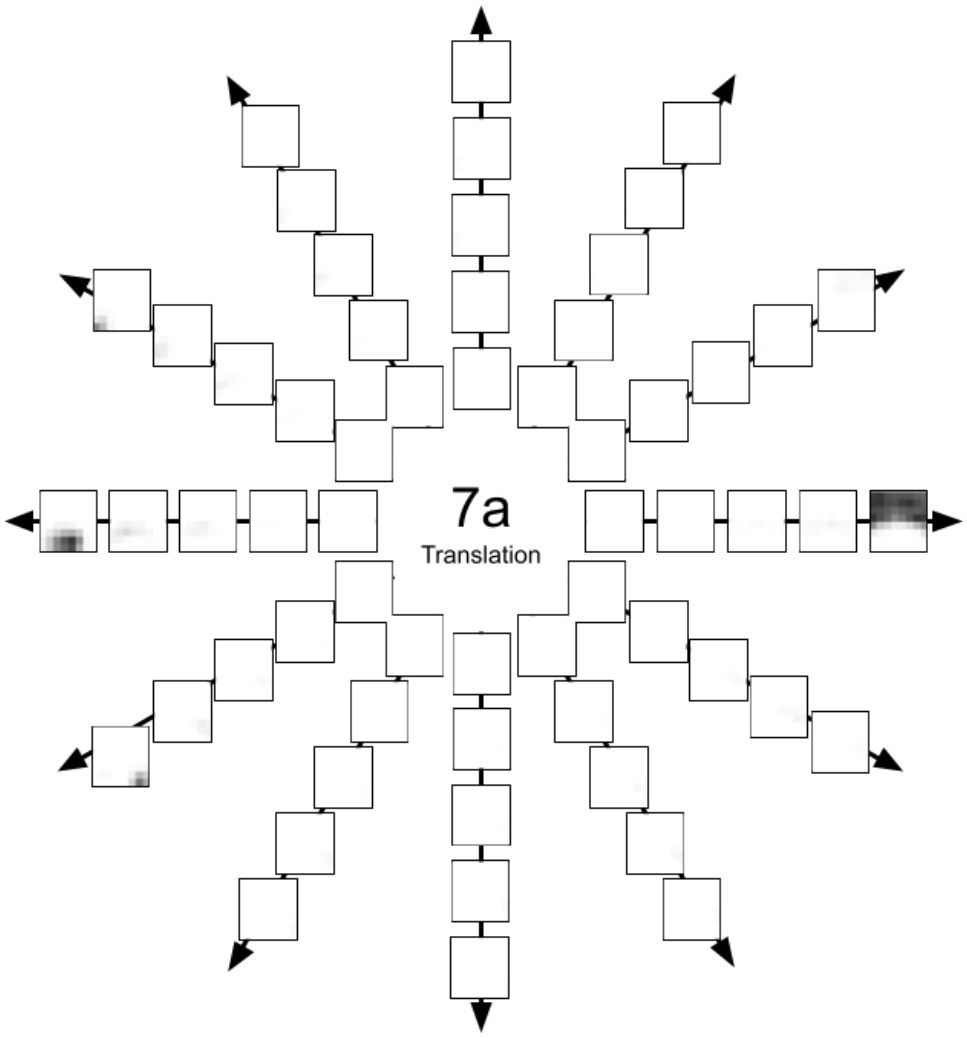

Figure A.6 (continued): This shows a close-up view of the 6-Frame model output for 7a Translation.

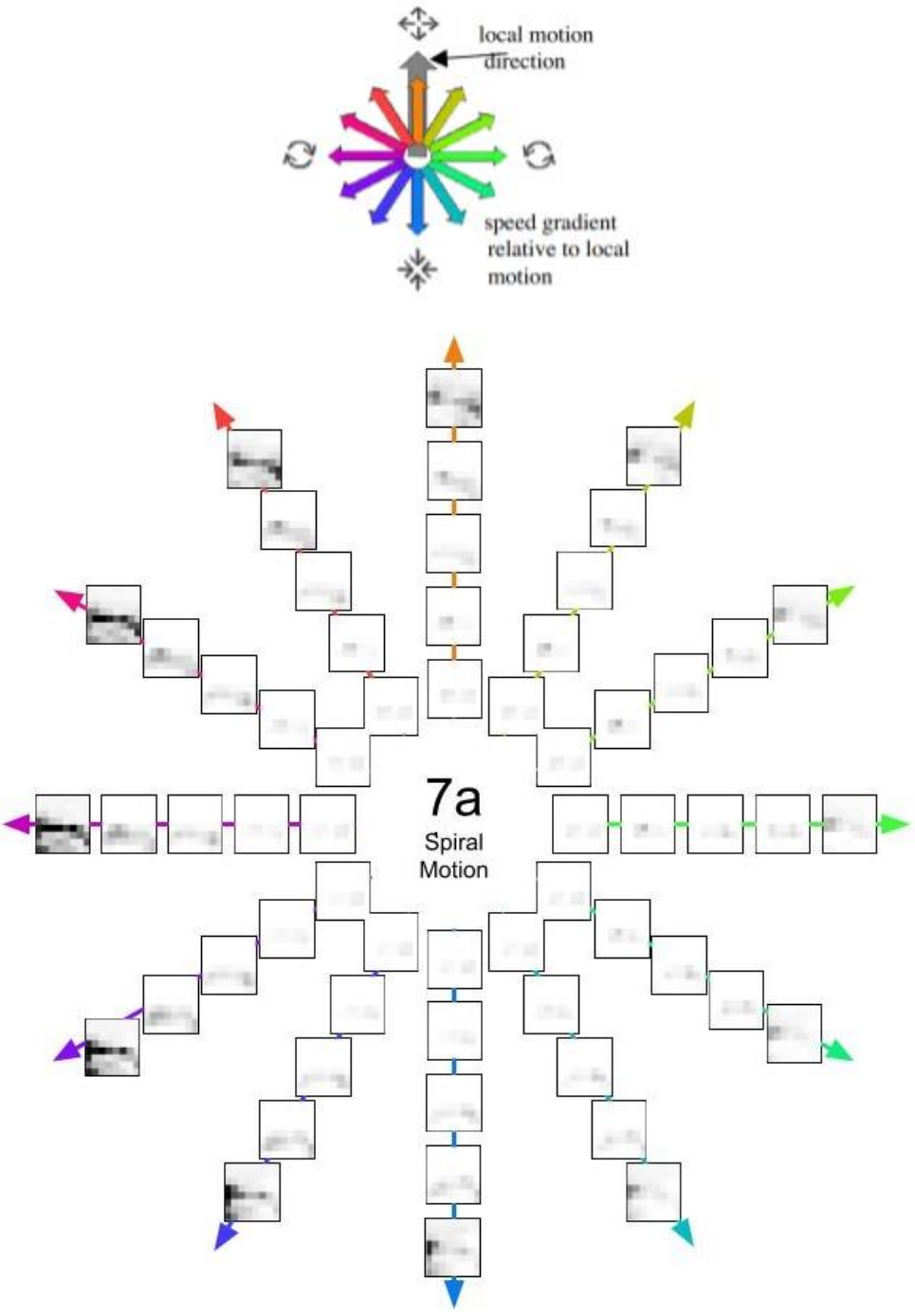

Figure A.6 (continued): 7a Spiral Motion shows the generalized spiral output in Area 7a. Coloured arrows indicate the inner angle δ between motion and speed gradient (See legend). This shows a close-up view of the 2-Frame model output for 7a Spiral Motion.

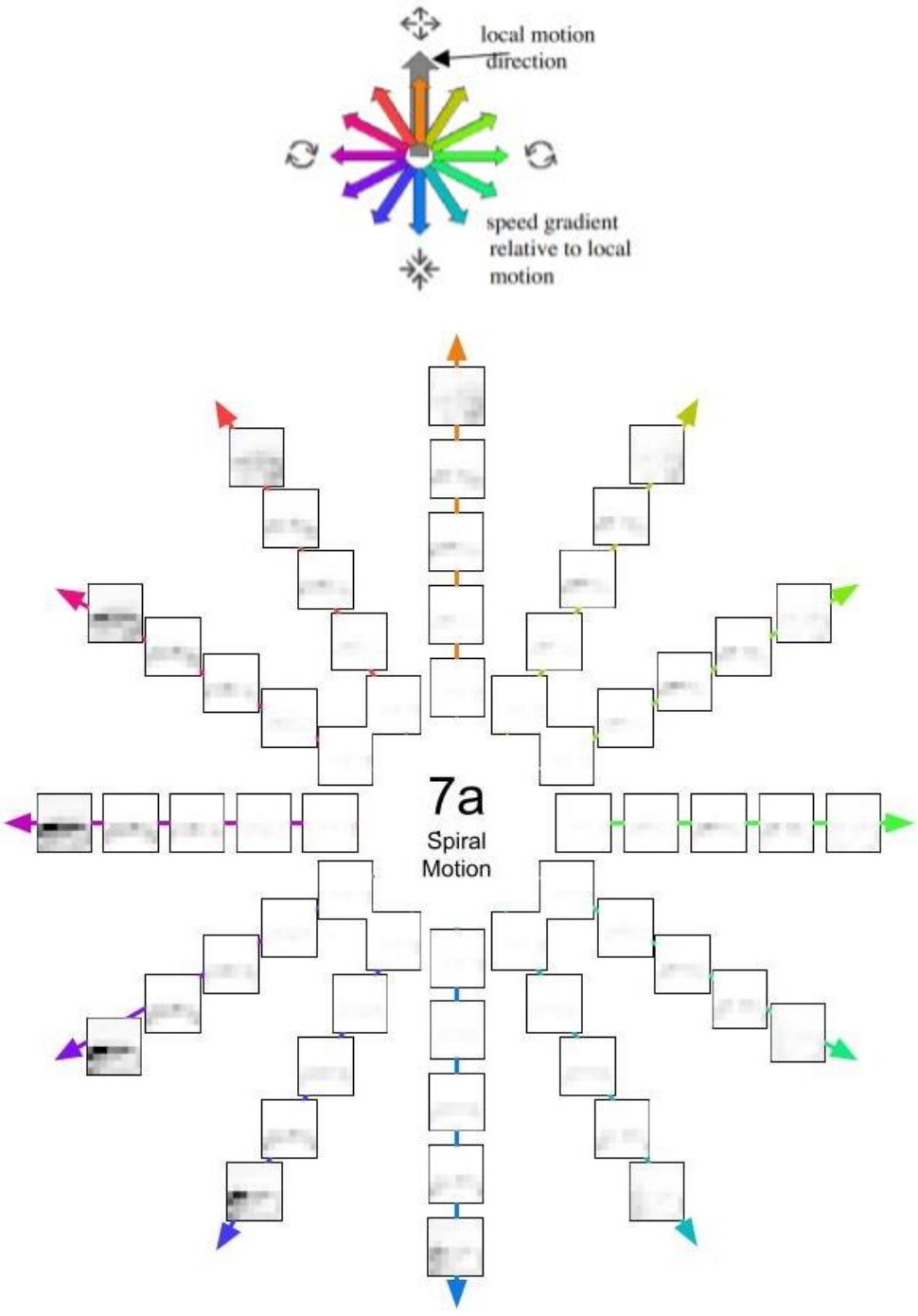

Figure A.6 (continued): This shows a close-up view of the 6-Frame model output for 7a Spiral Motion.

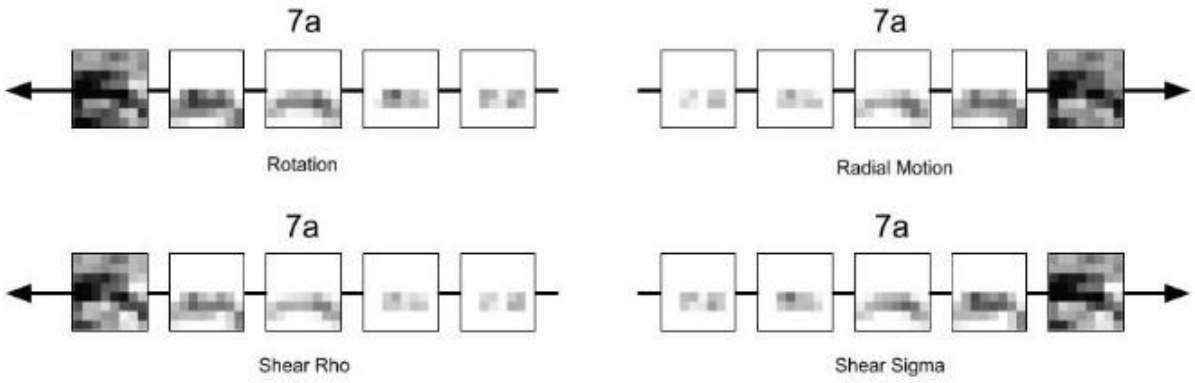

Figure A.6 (continued): Rotation, Radial Motion, Shear Rho, and Shear Sigma show the outputs for ART, ARD, ADR, and ADS in AP of Area 7a. This shows a close-up view of the 2-Frame model outputs for AP.

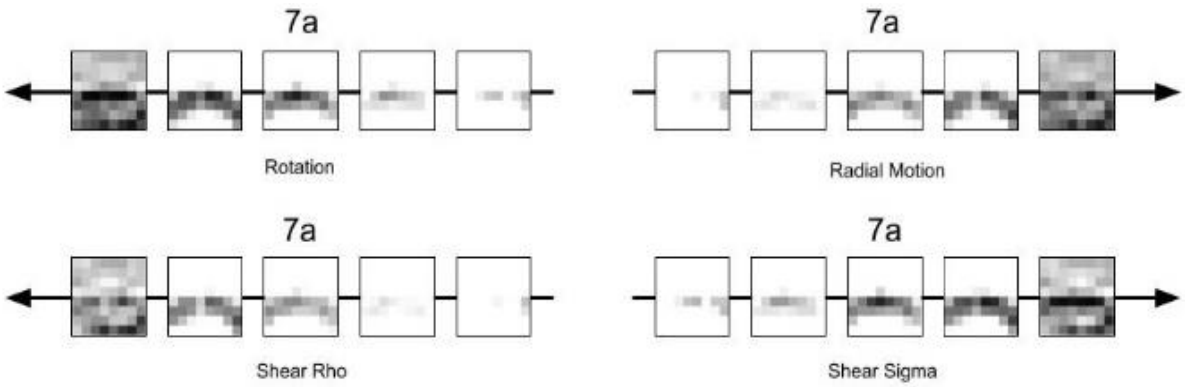

Figure A.6 (continued): This shows a close-up view of the 6-Frame model outputs for AP.