**Towards Temporal Semantic Scene Understanding**

by

**Mahsa Ghafarianzadeh**

M.Sc., The George Washington University, 2011

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2017

This thesis entitled:
Towards Temporal Semantic Scene Understanding
written by Mahsa Ghafarianzadeh
has been approved for the Department of Computer Science

_____

Professor Gabe Sibley

_____

Professor Christoffer Heckman

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Ghafarianzadeh, Mahsa (Ph.D., Computer Science)

Towards Temporal Semantic Scene Understanding

Thesis directed by Professor Gabe Sibley

The ability to quickly and accurately understand pixel-level scene semantics is a key capability required for various robotics applications such as autonomous driving. Until now, the temporal aspect of this problem has been largely overlooked. Therefore, the focus of research in this dissertation is to study the impact of temporal information in perception-related tasks and investigate whether it is useful to be included, more specifically for semantic scene understanding.

In this thesis, we first propose a set of novel techniques for unsupervised spatio-temporal segmentation in video sequences to obtain regions that are coherent in space and time. We then extend our method to exploit other strong cues present in the scene such as the depth signal or object parts to further improve the accuracy. The bottleneck in studying the temporal data is caused by both the limitations in computing resources and/or the lack of existing comprehensive labeled data. We tackle these issues by introducing a simple and efficient unsupervised label propagation algorithm that transfers the pixel-wise semantic labels from a groundtruth frame to its adjacent neighbor frames and produces auxiliary temporal groundtruth. Finally we take a further step towards the ultimate goal of holistic scene understanding and present a deep, recurrent multi-scale network that is capable of leveraging the temporal information present in the video data. We show that our model can be easily extended to the related problem of prediction to estimate the expected semantics of the scene a small number of frames into the future. We achieve promising state-of-the-art results on various datasets and prove that our temporal approach is superior to the non-temporal baseline.

## Dedication

To mom & dad.

## Acknowledgements

I am very grateful to my advisor, Gabe Sibley, for his guidance, support, patience and most importantly, encouragement. He never stopped believing in me and always presented me with more opportunities, challenges and turned me into an independent researcher. I always admire his enthusiasm and endless energy towards research. This dissertation would not have been possible without him.

I would like to especially thank Matthew Blaschko whom I had the pleasure of collaborating with and learning from. Even though we were working remotely throughout these years, he has been always very responsive. His insightful comments, feedback and advice on my research has shaped my thinking process and approach to research.

A very special thank you to my thesis committee members: Christoffer Heckman, Nikolaus Correll and James Martin for their support and encouragement. I would also like to thank my wonderful colleagues at Autonomous Robotics and Perception Group (ARPG) whom I have had the great fortune of interacting with over the past few years and becoming good friends.

I spent the last year of my PhD at Zoox Inc, which was a wonderful learning experience and had a lot of impact on my research. I would like to thank James Philbin for being an amazing mentor during my internship.

Lastly, I would like to thank my family for their continuous love and support over the years. My mom, Farideh, for being the strongest woman I know who inspired me to always reach higher. My dad, Zia, for being the most caring man I know who taught me to work hard. And my brothers, Mehdi and Reza, for being my first and forever friends whom I cannot imagine my life without.

# Contents

**Chapter**

# Tables

**Table**

# Figures

**Figure**

# Chapter 1

# Introduction

## 1.1 Objective

A significant part of humans cerebral cortex is devoted primarily to processing visual information which enables us to infer the meaning of what we see without the need for proximity involved in other senses. Humans learn to perceive the world around them by observing different cues such as motion, color and depth from very early during their perceptual and cognitive development. Also our capacity to comprehend the dynamics of visual scenes is continuously improved over time. As humans, we have a stunning ability to understand complex scenes in details and fairly quickly. We are able to accurately identify the objects, interact with them, understand the scene structures, navigate ourselves in the places that we have seen before, recognize and describe actions. However, this very basic set of skills still poses a lot of questions and challenges for machines. It is interesting to note that the supervision involved in the human learning process is rarely given on still images. We are extremely good at processing large amount of data and utilizing our prior knowledge about the world to reason about the unseen. In contrast, current computer vision systems are limited in their ability to process large amount of complex data and in their ability to leverage the temporal coherency present in the video data to build more accurate semantic models of the environment.

This thesis studies the impact of using temporal information for the task of holistic scene understanding. The goal is to develop an efficient and simple method that can be used in a wide variety of applications such as autonomous driving, robotic exploration of environments, video

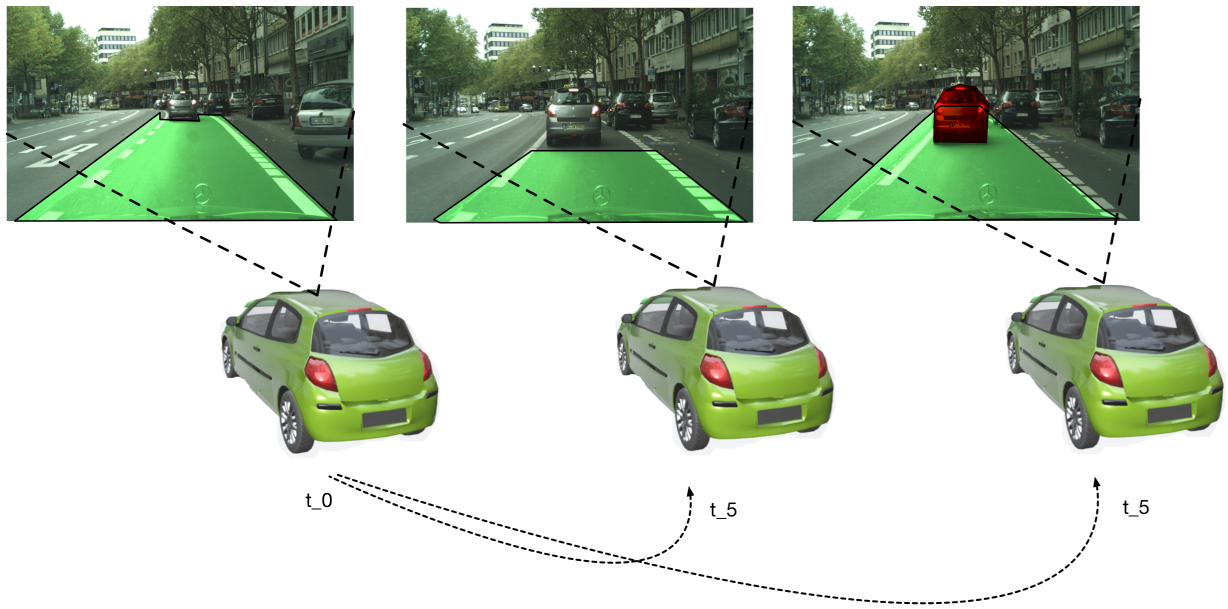


Figure 1.1: Illustration of free-space estimation.

analysis and surveillance and augmented or virtual reality. Figure 1.1 shows an application of semantic understanding in autonomous driving. Here, the hero is estimating the free-space for each frame where the planner is always ensuring that the estimated trajectory is lying within the green area. The free-space is obtained by processing different semantic categories in the scene such as the road, sidewalk, lane markings and etc.

## 1.2 Challenges

Nevertheless, the task of scene understanding in videos faces several challenges that needs to be overcome. Below is a list of few important challenges:

**Temporal Coherency.** Image segmentation techniques applied to each frame of a video sequence independently produce unstable segmentation results due to the fact that the scene structure might drastically change from one frame to another. Therefore, posing temporal segmentation as

spatial region matching between individual frames does not enforce temporal consistency of regions over time. Therefore, it is important to model the temporal variations of sequential data for scene understanding in videos.

**Causality.** Often image based segmentation techniques ignore the causality and redundancy available in the sequential data when being applied on each frame individually. The fact that each frame in a video is temporally related to its neighboring frames should be efficiently exploited in video based algorithms.

**Object Appearance.** The appearance of objects in the scene can drastically change due to several challenging conditions. Changes in camera viewpoint, illumination and lighting, scale and rotation preserve the visible parts of the objects in the scene. However, the actual appearance of the scene in terms of its pixels, can be changed quite significantly from one frame to another frame.

**Scene Dynamics.** Temporal scene understanding methods need to take into consideration the dynamics of the scene. In addition to the changes in the appearance of objects, the underlying motion in the video scene is of enormous importance mainly due to changes in object visibility. Occlusions and re-appearance can be caused by moving objects quite often in video sequences. Also changes in the camera view point might reveal parts of objects that were not previously visible in the scene.

**Memory and speed Constraints.** Given the large amount of pixels in a video, temporal segmentation approaches tend to be very slow and often require a large memory footprint. Hence, most of existing approaches focus on sequences that are short to reduce the complexity of the algorithm and also the computation time. For this reason, developing reliable and scalable algorithms that are memory efficient and produce accurate segmentation in videos is very valuable and challenging.

**Supervision.** Although huge amount of video data is easily accessible, they are mostly unlabeled due to the laborious task of human annotation. Therefore, it is necessary to design efficient unsupervised or semi-supervised algorithms that are able to benefit from processing large unlabeled video datasets.

**Lack of large scale annotated datasets.** The performance of deep learning algorithms depends greatly on the breadth and diversity of the examples in the training data. Image scene understanding has been widely studied in the literature but temporal algorithms are rare. This is mainly due to the fact that there exists no public dataset with labellings for all the frames in videos.

## 1.3 Thesis outline

This dissertation attempts to address the challenges mentioned in the previous section and is structured as the following:

In Chapter 2, we adapt an existing graph-based image segmentation and present an efficient and simple unsupervised technique for spatio-temporal segmentation. Our method is based on a low-rank spectral clustering algorithm. The complexity of graph-based spatio-temporal segmentation is dominated by the size of the graph, which is proportional to the number of pixels in a video sequence. In contrast to other works, we avoid over-segmenting the figures into super-pixels and instead generalize a simple graph based image segmentation. Our graph construction encodes appearance and motion information with temporal links based on optical flow. For large scale data sets naïve graph construction is computationally and memory intensive, and has only been achieved previously using a high power compute cluster. We make feasible for the first time large scale graph-based spatio-temporal segmentation on a single core by exploiting the sparsity structure of the problem and a low rank factorization that has strong approximation guarantees. We empirically demonstrate that constructing the low rank approximation using a subset of pixels (30%-50%) achieves performance exceeding the state-of-the-art on the Hopkins 155 dataset, while enabling the graph to fit in core memory. Figure 1.2 shows an illustration of our proposed method for spatio-temporal segmentation.

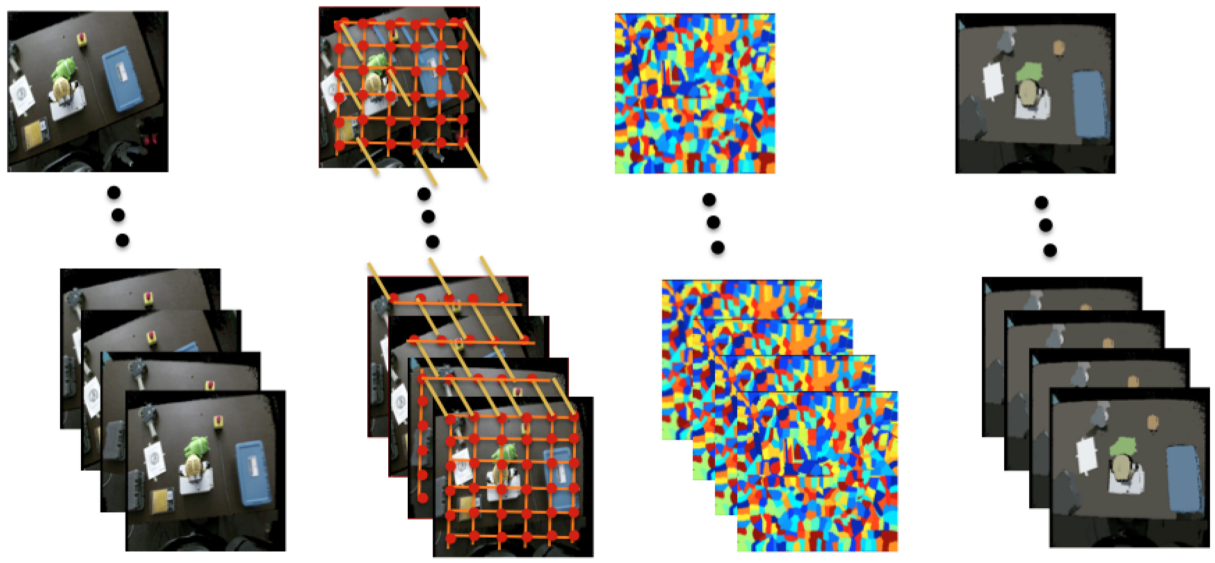

Figure 1.2: The overview of our spatio-temporal segmentation algorithm

In Chapter 3, we present a novel framework for spatio-temporal segmentation from RGBD video. The method employs both low-level (intensity, color) and high-level (deformable parts model) appearance features. Depth cues are incorporated in the similarity metric to provide an informative cue for object boundaries at depth disparities. We further explore structured local connectivity patterns to give high performance at low computational cost. Also we propose a novel context-aware aggregation method that uses a deformable parts model to group the detected parts of the object as a single segment with an accurate boundary. Detailed experiments on the NYU Depth Dataset and TUM RGBD Dataset is performed to compare against previous large-scale graph-based spatio-temporal segmentation techniques which shows the substantial performance advantages of our framework. Figure 1.3 shows an illustration of our object-centric segmentation method that uses Deformable Parts Model to segment out the background of chair in the detected bounding box.

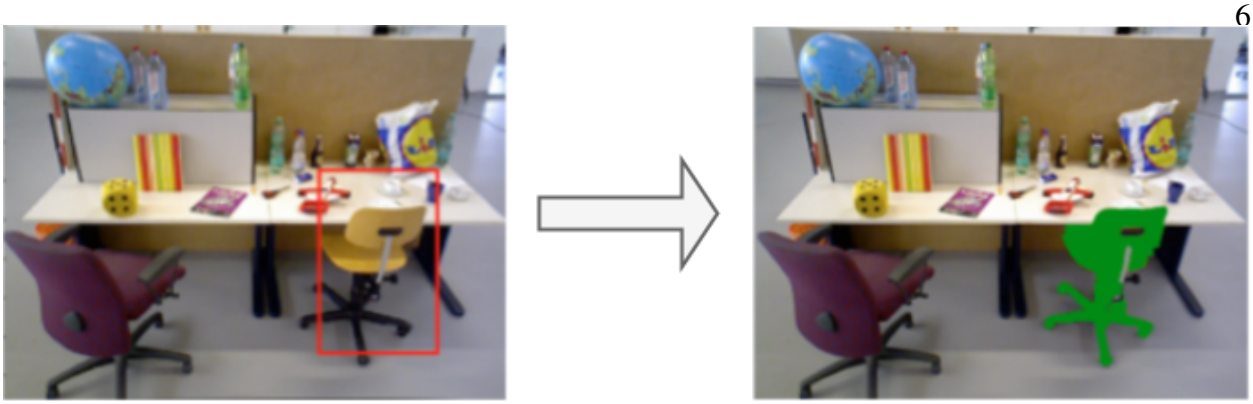

Figure 1.3: An example of our object-centric segmentation method

In Chapter 4, we present an unsupervised framework for simultaneous appearance-based object discovery, detection, tracking and reconstruction using RGBD cameras and a robot manipulator. The system performs dense 3D simultaneous localization and mapping concurrently with unsupervised object discovery. Putative objects that are spatially and visually coherent are manipulated by the robot to gain additional motion-cues. The robot uses appearance alone, followed by structure and motion cues, to jointly discover, verify, learn and improve models of objects. Induced motion segmentation reinforces learned models which are represented implicitly as 2D and 3D level sets to capture both shape and appearance. We compare three different approaches for appearance-based object discovery and find that a novel form of spatio-temporal super-pixels gives the highest quality candidate object models in terms of precision and recall. Live experiments with a Baxter robot demonstrate a holistic pipeline capable of automatic discovery, verification, detection, tracking and reconstruction of unknown objects. Figure 1.4 shows the result of tracking and reconstruction of a verified object using our method. In this scenario, the robot grasps an object on the desk and moves it around the scene (for 500 frames). A complete 3D model of the object is learned after the robot has enough observation of the object.

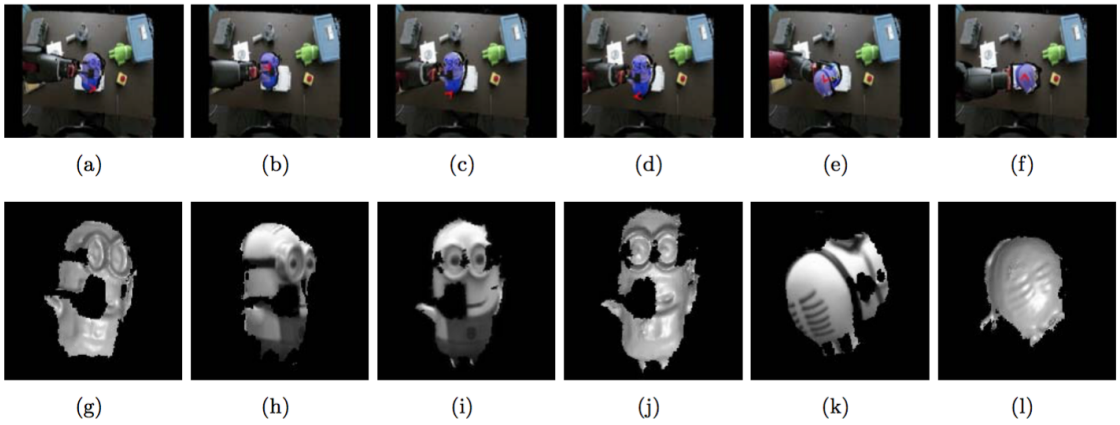


Figure 1.4: An example of our unsupervised object discovery, tracking and reconstruction technique

In Chapter 5, we propose a simple unsupervised method for transferring pixel-wise dense labels from finely annotated frames of video sequences to the rest of unlabeled frames. We are motivated by the fact that recent progress in computer vision has been driven by high-capacity models which are trained on large datasets. However to our knowledge, there is no public dataset that has groundtruth annotation for all the frames in a video sequence. Unfortunately, creating large datasets with pixel-level labels has been extremely costly and expensive due to the amount of human effort required. Our algorithm provides an efficient tool to reduce the cost of mundane manual labeling, therefore producing abundant amounts of usable groundtruth data. A novel application of this algorithm is for semantic segmentation and scene parsing especially in temporal semantic segmentation where having groundtruth for each individual frame in the sequence is beneficial. We also introduce a simple web-based video annotation tool which is based on the techniques introduced in previous chapters. Using this tool, we are able to acquire a rapid propagation of semantic labels within and across images of a video sequence. We validate the presented approach by producing dense pixel-level semantic annotations for about 18K images extracted from CamVid dataset where there is only 701 groundtruth frames are available. We also demonstrate qualitative results for Cityscapes dataset for which we generated about 105K auxiliary groundtruth. Fig-

ure 1.5 shows several samples of auxiliary groundtruth generated by our method from cityscapes dataset[33]. The middle highlighted frame is the clean groundtruth and the rest of frames are the outputs of our method.

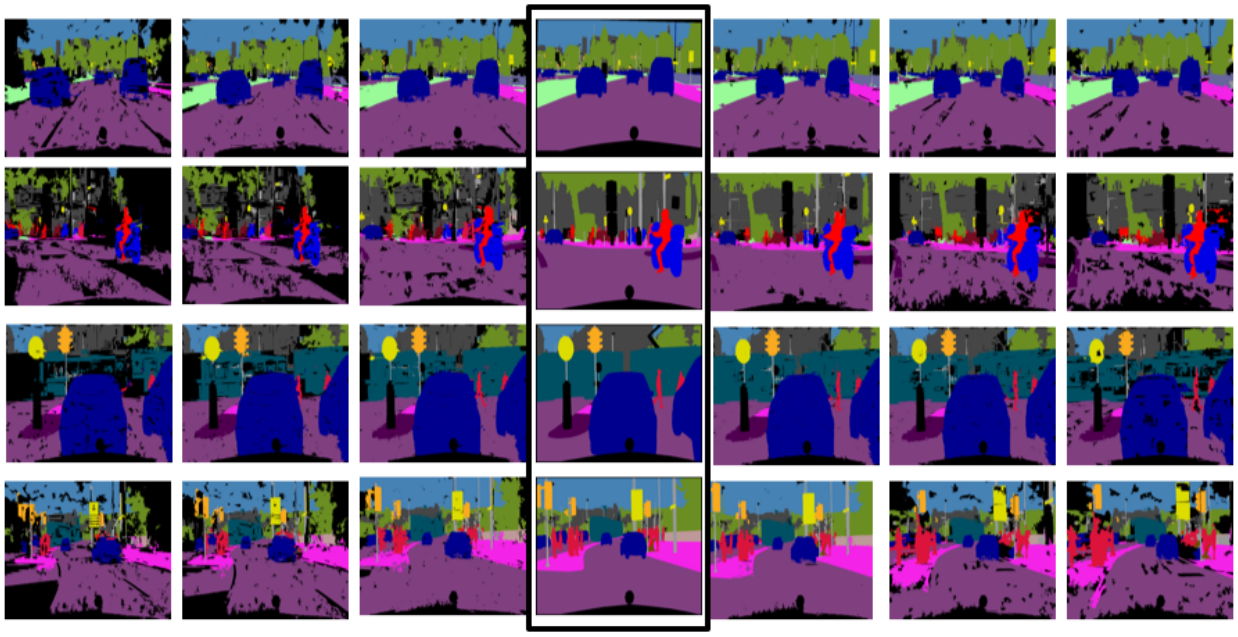

Figure 1.5: Examples of our label propagation algorithm

In Chapter 6, we present a deep, recurrent multi-scale network that is capable of leveraging information present in temporal data to improve the accuracy of semantic scene understanding. We propose using temporal information in two ways. Firstly, we formulate an approach that integrates semantic segmentation into a recurrent architecture where the model is given access to the previous frames in a sequence. We show that our model outperforms single frame segmentation when given access to this data even when groundtruth information is not provided for intermediate frames. Secondly, we show that our model can be easily extended to the related problem of prediction: estimating the expected semantics of the scene a small number of frames into the future. This is a crucial capability for many robotics applications, where planning needs knowledge of the probable near future as well as the present. Our architecture for semantic temporal segmentation has three

key components: 1) an encoder that extracts the spatial features of frames using CNNs; 2) an convolutional LSTM module that integrates temporal features over many frames while preserving the spatial correlation; and 3) a decoder that combines different scale feature maps and produces final pixel-wise predictions for either the input sequence or the predictive future sequence. Figure 1.6 shows the scheme of our network for temporal and predictive semantic segmentation.

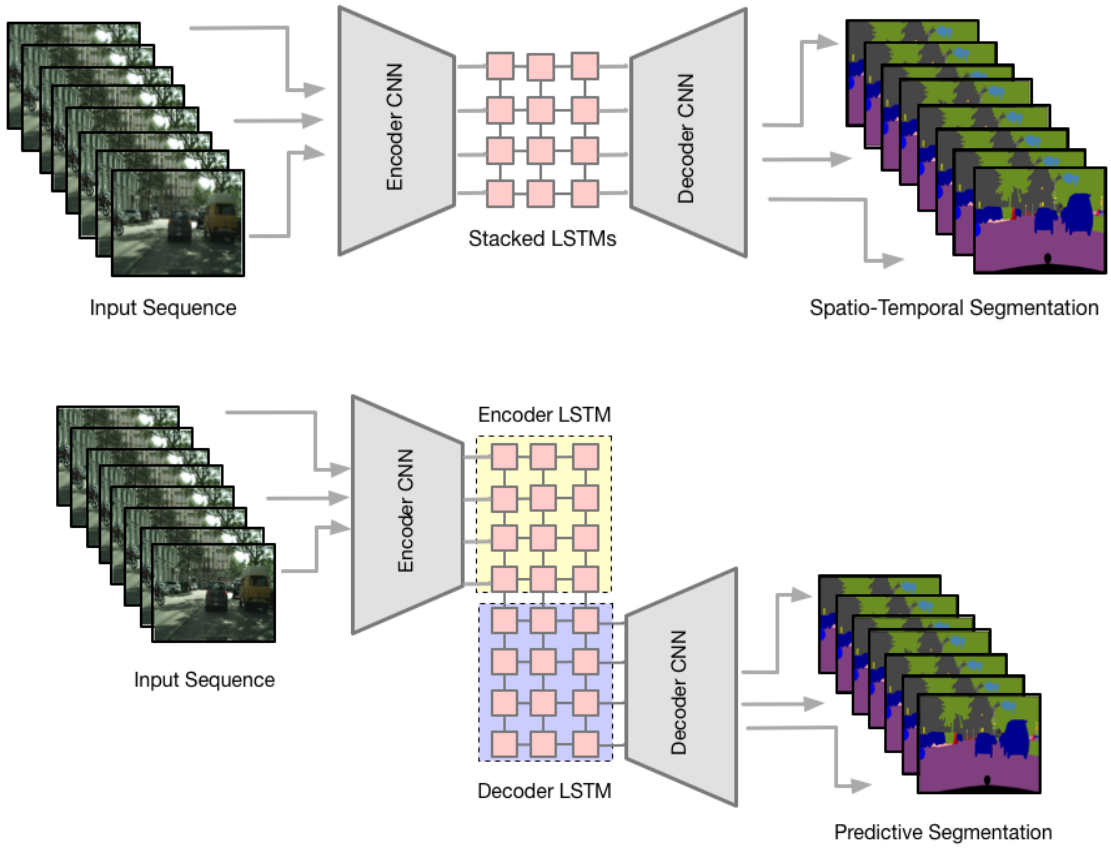


Figure 1.6: The network architecture of our temporal and predictive semantic segmentation model

Our spatio-temporal CNNs achieve promising result as demonstrated for the CamVid and Cityscapes datasets without resorting to more complex random-field inference or instance detection driven architectures. To our knowledge, our work is the first to exploit deep learning techniques for temporal semantic segmentation and prediction in image sequences. Our proposed module can be used in contemporary networks with little modification in order to exploit the temporal

information.

## 1.4 Publications

Most of the contributions described in this dissertation have first appeared as the following publications:

- **Chapter 2**: "Unsupervised spatio-temporal segmentation with sparse spectral clustering", which is presented in British Machine Vision Conference (BMVC), 2014. [53]
- **Chapter 3**: "Efficient, dense, object-based segmentation from RGBD video", which is presented in Robotics and Automation (ICRA), 2016. [54]
- **Chapter 4**: "Simultaneous localization, mapping, and manipulation for unsupervised object discovery", which is presented in Robotics and Automation (ICRA), 2015. [103]
- **Chapter 5**: "Generating Auxiliary Groundtruth via Label Propagation in Sequences", Under submission.
- **Chapter 6**: US Patent Application Serial Number 15/409,841 entitled "Spatial and Temporal Information for Semantic Segmentation", which is the work done during internship at Zoox Inc, 2016.
- **Chapter 6**: "Temporal pyramid Networks for Semantic Scene Understanding", Under submission.

# Chapter 2

## Unsupervised spatio-temporal segmentation with sparse spectral clustering

Spatio-temporal cues are powerful sources of information for segmentation in videos. In this work we present an efficient and simple technique for spatio-temporal segmentation that is based on a low-rank spectral clustering algorithm. The complexity of graph-based spatio-temporal segmentation is dominated by the size of the graph, which is proportional to the number of pixels in a video sequence. In contrast to other works, we avoid oversegmenting the figures into super-pixels and instead generalize a simple graph based image segmentation. Our graph construction encodes appearance and motion information with temporal links based on optical flow. For large scale data sets naïve graph construction is computationally and memory intensive, and has only been achieved previously using a high power compute cluster. We make feasible for the first time large scale graph-based spatio-temporal segmentation on a **single core** by exploiting the sparsity structure of the problem and a low rank factorization that has strong approximation guarantees. We empirically demonstrate that constructing the low rank approximation using a subset of pixels (30%-50%) achieves performance exceeding the state-of-the-art on the Hopkins 155 dataset, while enabling the graph to fit in core memory.

### 2.1 Introduction

The analysis of videos is a central task in computer vision. Videos encode useful information in appearance, motion cues, and temporal continuity, which can be leveraged to improve performance on a wide range of applications. Spatio-temporal segmentation is defined as the prob-

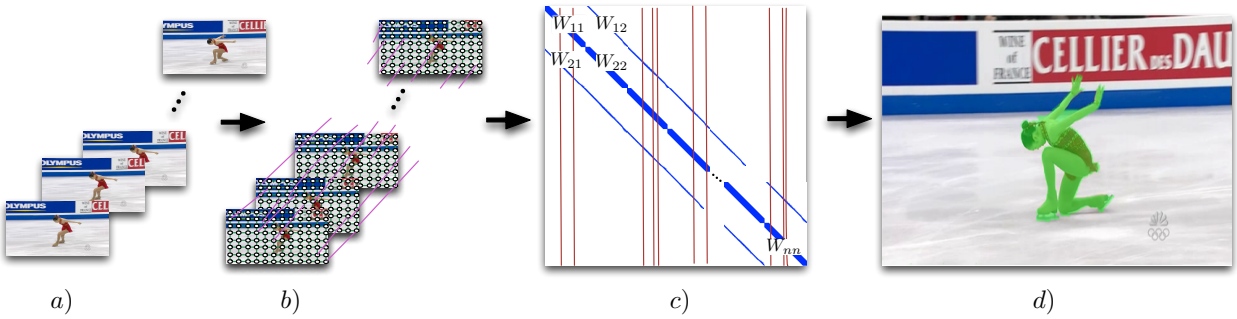


Figure 2.1: Approach overview: given a sequence of figures a) we construct a graph b) connecting pixels in a neighborhood and also to their temporal correspondences. This is represented as a very large sparse matrix, from which we select a random subset of columns (which correspond to pixels) – only the randomly selected pixels are used for graph and matrix construction. d) we employ spectral clustering segmentation based on efficient and accurate low rank factorization based on the Nyström method to approximate the graph Laplacian.

lem of partitioning a sequence of figures into coherent regions using motion and appearance cues. This is an essential step for different tasks such as tracking, object detection and video analysis. Spatio-temporal segmentation is a challenging problem because there are different factors that may change throughout the sequence such as camera motion, scale, illumination, viewpoint, deformation of objects, occlusion, etc.

In this paper, we improve on a family of methods that generalize graph based image segmentation to the spatio-temporal case. Such methods build a graph based on an affinity matrix with connectivity between neighboring pixels, as well as temporal links computed using optical flow. While such techniques have been applied previously, they have either been based on a sparse set of pixels, super-pixel segmentation, or have made use of extensive computing power to exhaustively compute a dense graph based segmentation. The central contribution of this paper is to introduce a set of strategies that enable us to compute a dense graph based segmentation using a single processor and fitting in core memory. This is done primarily through two innovations: (i) we exploit the sparsity structure of the spatio-temporal graph, and (ii) we make use of an efficient and accurate low rank factorization based on the Nyström method to approximate the graph Laplacian in a spectral clustering approach.

We randomly sample a subset of pixels in the sequence of figures and build a sparse approximation of the full affinity matrix. The sampled pixels are the nodes of the similarity matrix and edges encode appearance and motion cues. Next we perform spectral clustering on this subset similarity matrix as it provides optimal global solutions and use the solution to segment all the pixels. Our results show that not all of the pixels contain meaningful information about figures, and just a subset of pixels can be a good representation of the entire scene.

## 2.2 Related work

A number of video segmentation techniques has been proposed in recent years. Existing video segmentation approaches may differ in the assumptions they make and can be divided into different categories. One category include layered motion segmentation techniques [162, 141, 119] which assume that a scene can be represented with a number of overlapping layers, and each layer has a motion model that can be obtained by some reasoning process and therefore is capable of capturing moving objects in the scene. One drawback is that they may not perform very well on non-textured regions. However, they are capable of segmenting objects when there is occlusion in the scene due to their global motion model.

Another category of video segmentation techniques which is based on sparse number of locations with point trajectories [126, 17]. For example in [126], a robust subspace separation method has been developed that uses Agglomerative Lossy Compression (ALC) for motion segmentation. The advantage of this method is that it can successfully deal with corrupted or incomplete trajectories where most of other methods in this category fail. In [17], long term point trajectories based on dense optical flow are clustered into temporally coherent moving segments. However, a major drawback is that these techniques only consider up to 4% of the pixels in the video which is not enough for segmenting the entire video including non-moving objects. Hence, compared to dense video segmentation, the number of labeled points obtained by these sparse methods is considerably smaller.

There are also spatio-temporal segmentation methods which represent the video in a spatio-

temporal domain and generalize from image segmentation. For example, in segmentation by hyper-graph cuts [72], first the image is over-segmented into regions, then the regions are taken as the nodes of a hyper-graph that represents the relationship among regions and finally the hyper-graph is partitioned to get the video segmentation. Also in hierarchical graph-based segmentation [59], a volumetric video graph is over-segmented into spatio-temporal regions, and then a region graph is constructed by merging the similar regions obtained from previous over-segmentation. This process is repeated iteratively to create a hierarchical segmentation of the video. Hierarchical mean shift analysis [41], is another method that maps pixels of a video stack into a 7D feature space (three color, two motion and two position component) and then uses mean shift repeatedly to achieve hierarchical clustering of the points in the video. In [90], the concept of spatio-temporal closure over a graph of temporal super-pixels has been introduced which is an extension of closure detection techniques in 2D figures. Motion aware hierarchical image segmentation (MAHIS) [52], is another technique that obtains super-pixels from different variety of pixel-based affinities such as brightness,color and texture and then in the next step obtains video segmentation from a variety of super-pixel-based affinities. Another recent work [23], finds temporally consistent super-pixels in a video using a generative probabilistic model.

Spectral clustering has been used for image and video segmentation, and our work is closely related to [137, 51]. We aim to partition a graph that encodes appearance and motion similarity in a sequence of figures and obtain robust perceptual group which are coherent in time. However, spectral clustering is computationally expensive which makes it often impractical for large scale data set such as spatio-temporal data. In [51], the Nyström method has been applied as a numerical solution for speeding up the spectral clustering. They consider the video as a space time volume and partitioned the volume into coherent segments. In [149], they gained more computational power by parallelism in GPUs and perform video segmentation more efficiently. We instead use an algorithm proposed by [93], which is a time and space efficient spectral clustering algorithm for estimating the rank $k$ approximation of normalized Laplacian. This forms the core of our method to perform spatio-temporal segmentation in memory on a single CPU.

## 2.3 Approach

### 2.3.1 Creating the Graph

Given an image $I$, we create a graph $G = (V, E, W)$, where the graph nodes $V$ are the pixels in the image and are connected by edge $E$ if they are within distance $r$ from each other. As discussed in [34], in general choosing a smaller $r$ makes the segmentation faster and choosing a larger $r$ helps with propagating local grouping cues across larger neighborhood which is useful for detecting faint contours in the image. $W$ measures the similarity of pixels connected by an edge. We define $W$ as the following:

$$W_{ij} = \exp \frac{-d^2(s_i, s_j)}{\sigma_i \sigma_j}$$

where $W_{ij} = 0$ for $i =$j, and $s_i$ denotes pixel color and $d(s_i, s_j)$ is the Euclidean distance. $\sigma$ is a local scaling parameter [172] which takes into account the local statistics of the neighborhood around pixels $i$ and $j$. Local scaling parameter is defined by:

$$\sigma_i = d(s_i, s_k)$$

where $s_k$ is the $K$'th neighbor of pixel $i$.

In order to extend this to video, we make use of optical flow and add temporal motion information to the graph. We use optical flow [96] to compute the motion vectors between frames. Then we connect pixel $(x, y)$ in frame $t$ to its 9 neighbors along the backward flow $(u, v)$ in frame $t-1$, *e.g.* $(x + u(x,y) + \delta_x, y + v(x,y) + \delta_y)$ for $\delta_x, \delta_y \in \{-1, 0, 1\}$. The similarity matrix for the video is a sparse symmetric block diagonal matrix of the size $n$ = *number of frames* $\times$ *number of pixels in one frame*.

### 2.3.2 Partitioning the Graph

### 2.3.3 Spectral Clustering

Spectral clustering has been widely used for partitioning the similarity graphs which is also the core of our algorithm. Below we provide the details of the algorithm presented in [116]:

Given a set of $n$ points $S = \{s_1, \cdots, s_n\}$, cluster them into $K$ clusters as follows:

(1) Form the affinity matrix $W \in \mathbb{R}^{n \times n}$ (described above)

(2) Define $D$ to be a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} W_{ij}$ and construct the normalized affinity matrix $L = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$.

(3) Select a desired number of groups $K$.

(4) Find $v_1, \cdots, v_K$, the $K$ largest eigenvectors of $L$, and form the matrix $X = [v_1, \cdots, v_K] \in \mathbb{R}^{n \times K}$

(5) Re-normalize the rows of $V$ to have unit length yielding $U \in \mathbb{R}^{n \times K}$, such that $U_{ij} = \frac{V_{ij}}{(\sum_j V_{ij}^2)^{\frac{1}{2}}}$

(6) Treat each row $Y$ as a point in $\mathbb{R}^K$ and cluster using k-means.

(7) Assign the original point $s_i$ to cluster $k$ if and only if the corresponding row $i$ of the matrix $U$ was assigned to cluster $k$.

It is worthy to note that the $O(n^3)$ complexity of eigen decomposition of $L$ and also storing the similarity matrix often makes it expensive and impractical when $n$ is large.

#### 2.3.3.1 Nyström Method

The Nyström method used by [51] is a popular technique to tackle the mentioned problem by approximating a low-rank matrix of $W$, since we only need a number of eigenvectors. Given a symmetric matrix $W \in R^{n \times n}$, first $m \ll n$ columns of the matrix will be sampled. For simplicity, we assume that these sampled columns are the first $m$ columns of $W$. Matrix $W$ can be represented as follows:

$$W = \begin{pmatrix} W_{11} & W_{21} \\ W_{21}^T & A_{22} \end{pmatrix}$$

and

$$W_{:1} = \begin{pmatrix} W_{11} & W_{21} \end{pmatrix}$$

where $W_{11} \in R^{m \times m}$, $W_{21} \in R^{n-m \times m}$, and $W_{22} \in R^{n-m \times n-m}$. The Nyström approximation of $W$ is defined by:

$$\hat{W_{nys}} = W_{:1} W_{11}^{-1} W_{:1}^T$$

The normalized cut requires the computation of the k smallest eigenvectors of the normalized Laplacian $L = I - D^{-1/2} W D^{-1/2}$, which are the same as the $k$ largest eigenvectors of the $n \times n$ matrix

$$M = D^{-1/2} W D^{-1/2}$$

As we mentioned earlier, the direct decomposition of $M$ takes $O(n^3)$ and can be extremely expensive when $n$ is large. In [51], this problem is tackled by using the Nyström method as follows. First to avoid using the whole $W$, the degree matrix is approximated as:

$$\hat{D} = \text{diag}(\hat{W_{nys}} 1)) = diag(W_{:1} W_{11}^{-1} W_{:1}^T)$$

by using the Nyström approximation of $W$. Let $D_{:1} = diag(A_{:1}^T 1)$. The sub-matrices in $M$ are:

$$M_{11} = D_{:1}^{-1/2} W_{11} D_{:1}^{-1/2}$$

and

$$M_{:1} = D^{-1/2} W_{:1} D^{-1/2}$$

Using $\hat{D}$, $M_{:1}$ can be reformulated as:

$$\hat{M}_{:1} \simeq \hat{D}^{-1/2} W_{:1} \hat{D}^{-1/2}$$

and $M$ as

$$\hat{M} = \hat{M}_{:1} M_{11}^{-1} \hat{M}_{:1}^T$$

where $\hat{M}_{:1}$ now has the approximate eigenvectors of $M$. In general, $\hat{M}_{:1}$ is not orthogonal and orthogonalization precedure needs to be done is $W$ is positive semi-definite(psd), we can define $S$

as:

$$S = M_{11} + M_{11}^{-\frac{1}{2}} \hat{M}_{21} \hat{M}_2 1^T M_{11}^{-\frac{1}{2}}$$

and perform singular value decomposition (SVD) on $S$ to get

$$S = U_S \Lambda_S {U_S}^T$$

The approximated $\hat{L}$ then can be defined as $\hat{L} = V \Lambda V^T$ and $V^T V = I$, where

$$V = \hat{M}_{:1} M_{11}^{-1/2} U_S {\Lambda_S}^{-1/2}$$

But if $A$ is indefinite, a more complicated process is required. We define $Z$ as:

$$Z = \bar{U}_S \Lambda^{\frac{1}{2}}$$

And ${\bar{U}_S}^T$ as:

$${\bar{U}_S}^T = \begin{bmatrix} U_S^T & \Lambda_S^{-1} U_S^T B \end{bmatrix}$$

so $\hat{M} = ZZ^T$. If singular value decomposition of $ZZ^T = F \Sigma F^T$, then $V = ZF\Sigma^{-\frac{1}{2}}$ has the orthogonalized eigenvectors of $\hat{M}$. This algorithm is expensive for large scale data sets, since the time complexity of this algorithm is $O(nm^2 + m^3)$ and it requires keeping $W_{11}$ and $W_{:1}$ in the memory. Hence, we are using a time and space efficient spectral clustering via column sampling [93], that is similar to Nyström method, but with a further rank-$k$ approximation of the normalized Laplacian using the sampled sub-matrix of the similarity matrix. This algorithm has shown promising results, since it reduces the time and space complexity of Nyström method and also it is able to recover all the degree information of the selected points. Here we briefly review the algorithm proposed in [93]. First we form the sampled sub-matrix $W_{11} \in R^{m \times m}$ and compute the degree matrix $D_*$ for $W_{11}$. Next, we define $M_* \in R^{m \times m}$ as

$$M_* = D_*^{-\frac{1}{2}} W_{11} D_*^{-\frac{1}{2}}$$

and $S_* \in R^{m \times m}$ based on the rank-$k$ approximation $M_{*,k}$ of $M_*$ as

$$S_* = D_*^{-\frac{1}{2}} M_{*,k}^{-1} D_*^{-\frac{1}{2}}$$

$M_{*,k}$ is computed by performing eigen decomposition of $M_*$ and find the $k$ largest eigenvalues $\Lambda$ and the corresponding eigenvectors $V$. Finally the approximated $\hat{W}$ is obtained by

$$\hat{W} = W_{:1} S_* W_{:1}^T$$

By having $M_{*,k} = V\Lambda V^T$ and replacing it in above formulas, we get $\hat{W} = Q\Lambda Q^T$, where

$$Q = W_{:1} D_*^{-\frac{1}{2}} V \Lambda^{-1}$$

Lastly, rank-$k$ approximation of $L$ is obtained by following

$$\hat{M}_k = \hat{D}^{-\frac{1}{2}} \hat{W} \hat{D}^{-\frac{1}{2}} = U\Lambda U^T$$

where $U = \hat{D}^{-\frac{1}{2}} Q$ and $\hat{D}$ is the degree matrix of $\hat{W}$. The time complexity of this algorithm is $O(nmk)$ and there is no need to store large similarity matrix $W$ or its sampled columns in the memory. Also we are using the proposed inexpensive algorithm by [93] to orthogonalized estimated eigenvectors $U$.

### 2.3.4 Post Processing

After performing spectral clustering on the similarity graph and obtaining the clusters, we first quantize each cluster into 256 bins (16 bins for each channel) and compute the *RGB* histogram. Then we find the Euclidean distance between adjacent clusters and merge them repeatedly if their similarity is more than a threshold $\tau$ to achieve the final segmentation. Figure 2.2 shows an illustration of this step.

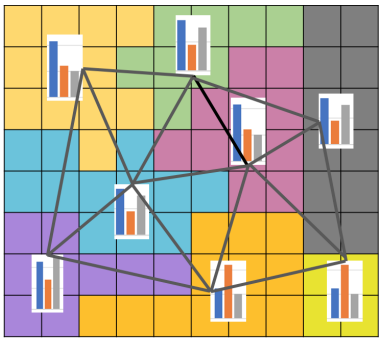

Figure 2.2: Illustration of post processing step

## 2.4 Results

To evaluate the performance of our segmentation method, we run several experiments and provide quantitative and qualitative results. We compare our method against motion segmentation techniques such as [126, 17], which cluster point trajectories in the sequence of figures and also other dense video segmentation techniques such as [149, 59]. In all of our experiments, we randomly select a subset sample of size 30%-50% for approximating the similarity matrix. The quality of our segmentation highly relies on the selected number of clusters for k-means and also distance $r$ for computing pixel similarities. Choosing larger numbers for these parameters makes the overall segmentation better. However, the disadvantage is that it creates unsmooth boundaries and captures most of the undesired details in the scene and also increases the size of matrix rapidly. We found that by choosing a distance size of 11-15 and cluster size of 300-500, we achieve reasonable performance.

We use an annotated dataset by [17] for motion segmentation. This dataset includes 26 sequences which is annotated sparse in time and dense in space. We ran our method on three scenarios and show the numerical result in Table 1. The first section shows the segmentation result for the first 10 frames of all 26 available sequences in the dataset, the second section shows the segmentation result for the first 50 frames of 15 selected sequences in the dataset and the third scenario shows the result for the first 200 frames of 7 selected sequences. The density shows the percentage of pixels for which a label is reported. The overall error is the number of mislabeled pixels over the total number of labeled pixels. the overall accuracy is the number of correctly labeled pixels over the total number of pixels. The average error is similar to the overall error but computed separately for each region first and then the average across all the regions is reported. The over-segmentation error indicates the number of clusters that needs to be merged to fit the ground truth regions. Also the number of regions with less than 10% error is reported as the extracted objects.

In this experiment, we aimed to compare our method against the method in [149]. The

Table 2.1: Comparison of dense and sparse segmentation methods

| Method | Density | Overall Error | Overall Accuracy | Average Error | Over segmentation | Extracted Objects |
|---|---|---|---|---|---|---|
| First 10 frames (26 sequences) | | | | | | |
| Ours (Sample:50%) | 100% | 5.06% | 94.94% | 22.31% | 6.12 | 22 |
| Sundaram **et al**[149] | 100% | 6.97% | 93.03% | 19.72% | 7.15 | 20 |
| Brox **et al**[17] | 3.34% | 7.75% | 3.08% | 25.01% | 0.54 | 24 |
| Rao **et al**[126] corrupted | 2.98% | 7.88% | 2.74% | 24.05% | 0.15 | 26 |
| Rao **et al**[126] incomplete | 3.34% | 11.20% | 2.97% | 26.73% | 0.54 | 19 |
| First 50 frames (15 sequences) | | | | | | |
| Ours(Sample:50%) | 100% | 7.63% | 92.37% | 32.24% | 14.9 | 5 |
| Sundaram **et al**[149] | 100% | 8.42% | 91.58% | 28.76% | 13.0 | 5 |
| Brox **et al**[17] | 3.27% | 7.13% | 3.04% | 34.76% | 0.53 | 9 |
| Rao **et al**[126] corrupted | 1.53% | 7.91% | 1.41% | 42.13% | 0.36 | 8 |
| Rao **et al**[126] incomplete | 3.27% | 16.42% | 2.73% | 49.05% | 6.07 | 2 |
| First 200 frames (7 sequences) | | | | | | |
| Ours(Sample:50%) | 100% | 13.24% | 86.76% | 47.83% | 5.3 | 6 |
| Sundaram **et al**[149] | 100% | 15.46% | 84.54% | 45.05% | 5.86 | 3 |
| Brox **et al**[17] | 3.43% | 7.64% | 3.17% | 31.14% | 3.14 | 7 |
| Rao **et al**[126] corrupted | 0.20% | 0.00% | 0.20% | 74.52% | 0.40 | 1 |
| Rao **et al**[126] incomplete | 3.43% | 19.33% | 2.77% | 50.98% | 54.57 | 0 |
| Grundmann **et al**[59] | 100% | 20.77% | 79.23% | 40.37% | 10.42 | 0 |

reason is that in contrast with their method which achieves good performance by using GPUs and heavily depending on parallelization, we exploit numerical algorithms to make our method fast and efficient. The density of both methods is 100% but we achieved comparable errors while using just a subset of pixels (30%-50%) to label all of the pixels as compared with their method that uses all the pixels for clustering. Compared to the motion segmentation methods [126, 17], our over-segmentation error is higher but that is due to the fact that these methods only use sparse number of points to label about 4% of the data while we are labeling all the pixels and achieve better accuracy.

## 2.5 Discussion

There are several avenues for future research based on the methods presented in this work. Sparse spectral clustering algorithms are designed to be agnostic to the problem domain. However, we have a substantial amount of domain knowledge we can exploit to expedite the clustering.

The matrix factorization employed uses uniform sampling to select columns. However, selective sampling strategies generally decrease the approximation error for a given number of columns. As the error rate of our method is comparable to using 100% of the columns, this indicates that we may achieve an even higher savings in memory using selective sampling. An additional likely source of improvement is through the use of similarity measures based on feature functions that encode texture or other information sources in addition to color.

Figure 2.3: Results form the Army Men sequence. Notice the fine detail around the top-right gun figure.

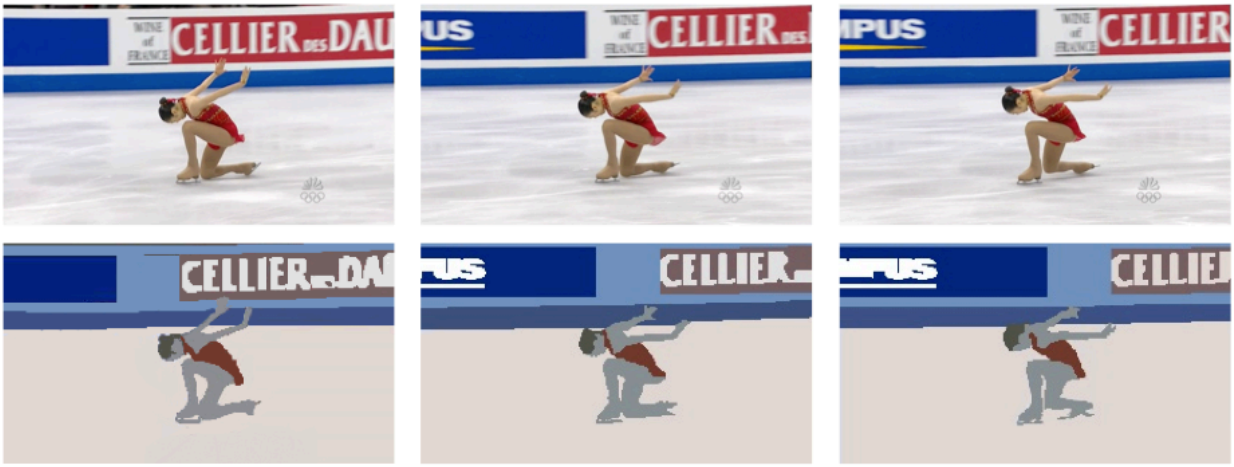


Figure 2.4: Results from Ice-skater Yu-Na Kim, 2009 World Championships, ©2009 NBC Olympics.

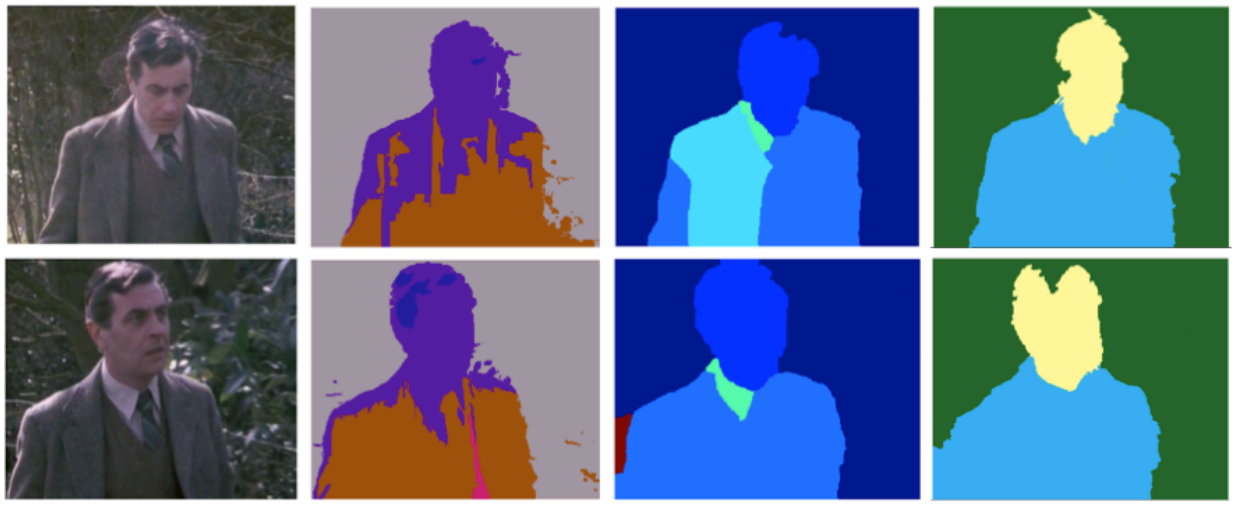

Figure 2.5: Comparison of our technique vs motion segmentation by [149] and [59]. Left-to-right shows video frames 1 and 50. Second row: results from Hierarchical graph segmentation [59]; third segmentation with the full graph using a cluster of GPUs [149]; fourth row, our result runs on a single core, does not employ contour smoothing or texture similarity.

## 2.6 Summary

In this work, we have demonstrated a novel method for spatio-temporal segmentation of dense pixel trajectories based on spectral clustering. We found that fully connecting pixels to their spatial neighbors within a given radius is an effective strategy for improving segmentation accuracy. Additionally, we use optical flow to more accurately compute temporal connectivity than a simple method based on an interpretation of the video sequence as a 3D volume. In contrast to previous work, we do not resort to super-pixel segmentation to achieve computational tractability and memory efficiency. Instead, we exploit the natural sparsity structure of the graph, and employ a low rank approximation of the Laplacian closely related to the Nyström method. We have found that sampling 30-50% of the pixels to index columns of the low rank approximation leads to comparable performance with a method that uses 100% of the columns. This strategy results in a spectral clustering method that can run on a single processor with the graph representation

fitting in core memory. For instance, a 100 frame 640x480 sequence naively requires 6.8GB of memory, while with our method we are able to reduce this to 2.1GB and still retain accuracy. We have demonstrated the effectiveness of the approach on the Hopkins 155 data set, where we have achieved the best reported results for dense segmentation using an order of magnitude less computation than [149].

# Chapter 3

## Efficient, dense, object-based segmentation from RGBD video

Spatio-temporal cues offer a rich source of information for inferring structural and semantic scene properties. A particularly useful representation in computer vision is a spatio-temporal video segmentation. Together with motion, knowledge of depth can substantially improve superpixel segmentation. In this work we present a novel framework for spatio-temporal segmentation from RGBD video. The method employs both low-level (intensity, color) and high-level (deformable parts model) appearance features. Motion is incorporated through the use of optical flow to construct the temporal connections in the graph Laplacian. Depth cues are incorporated in the similarity metric to provide an informative cue for object boundaries at depth disparities. Naïve application of spectral clustering to dense spatio-temporal graphs leads to a high computational cost that is typically addressed through the use of GPUs or computer clusters. By contrast, we build upon a recently proposed Nyström approximation strategy for spatio-temporal clustering that enables computation on a single core. We further explore structured local connectivity patterns to give high performance at low computational cost. Also we propose a novel context-aware aggregation method that uses a deformable parts model to group the detected parts of the object as a single segment with an accurate boundary. Detailed experiments on the NYU Depth Dataset and TUM RGBD Dataset is performed to compare against previous large-scale graph-based spatio-temporal segmentation techniques which shows the substantial performance advantages of our framework.

## 3.1 Introduction

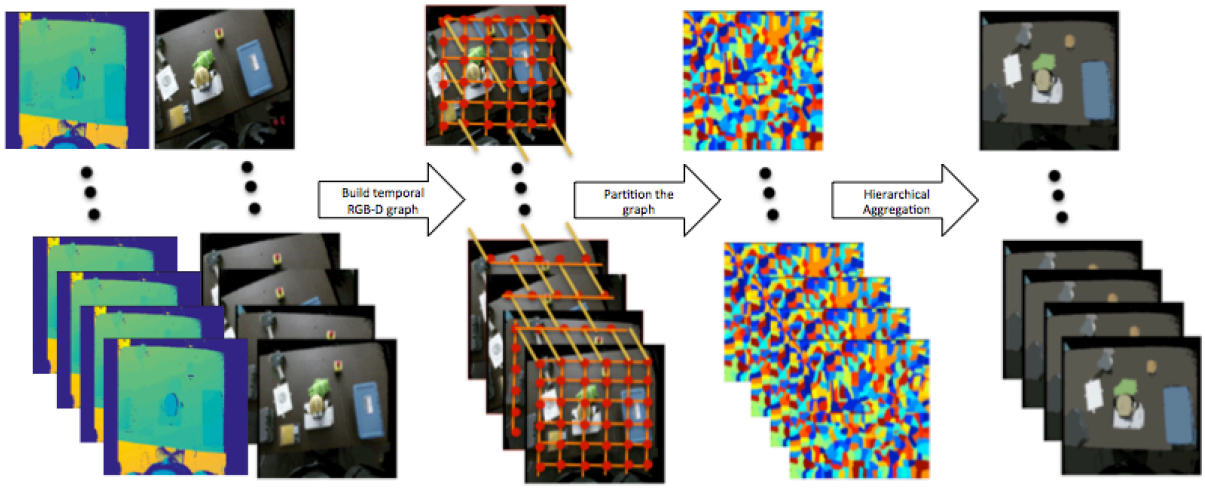


Figure 3.1: Approach Overview: (left-to-right) Given a sequence of RGBD figuress, We first construct a similarity graph using appearance and depth information, we also add the temporal links computed using optical flow, then we use Nyström method to obtain an over-segmentation of the figuress, and finally we use a hierarchical aggregation method to obtain the final spatio-temporal segments in the video

Video segmentation is defined as the problem of grouping a sequence of figuress into coherent regions with similar appearance, motion and temporal continuity. The goal of video segmentation is to represent figures sequences through homogeneous regions where the same object should carry the same unique label along the whole video. Spatio-temporal cues are essential to obtain good performance in video segmentation, as appearance-based cues may be ambiguous in regions of low texture and similarity between foreground and background. An important additional information source has come in the form of RGBD cameras. These cameras have become ubiquitous in recent years, and are being incorporated in a large variety of entertainment systems, user interfaces, robotic sensors, etc. The information provided by depth is **complementary** to that provided by motion cues: motion can provide information about points that (do not) move together, while

depth sensors provide information about depth disparities even in the absence of motion. This provides an excellent opportunity to boost spatio-temporal segmentation accuracy through the use of potentially noisy depth information. Spatio-temporal segmentation of RGBD videos has numerous applications in sub-fields of computer vision. In robotics, RGBD cameras are more-and-more frequently employed to infer environment structure and to identify individual objects that can be manipulated [2]. Self-driving cars must be able to accurately determine scene components, their spatial extents, and their trajectories [106]. Consequently analysis of RGBD figuress and videos is becoming an important problem in robotics [50].

High level vision tasks often require, or can be aided by knowledge of the spatio-temporal relationship between objects inherent in a high-quality RGBD video segmentation. In object tracking, frequent difficulties such as occlusion handling can be alleviated through segmentations that account for both appearance motion cues and depth disparities. Action recognition has been shown to be substantially aided through the incorporation of depth cues [82] and spatio-temporal segmentations with depth information is a rich representation for this task.

In this paper, we show that the inclusion of depth and deformable part-model information can greatly improve video segmentation. We are interested in unsupervised spatio-temporal segmentation of the RGBD video sequence. Our approach extends prior work on graph-based spatio-temporal segmenation [53] by adding depth information, parts-based aggregation, and also a novel technique for connecting pixels within a neighborhood. The central strategy is to build a pixel level spatio-temporal graph that encodes appearance, depth and motion information with temporal links based on optical flow and then cut the graph to obtain segments that are coherent over time. We present two different aggregation methods for merging the over-segmentation result. One is a hierarchical approach and the other is based on the deformable part models for an object of interest. We employ a recently proposed Nyström approximation strategy to ensure computational tractability, resulting in a method that is both accurate, and tractable to run on a single core. We believe this to be the first demonstration of dense spatio-temporal segmentation from RGBD videos that does not make use of extensive computational infrastructure such as GPUs or compute clusters. At the same

time, we demonstrate substantially improved performance over a variety of approaches. Advantages of the proposed method are demonstrated using the NYU Depth Dataset [112] and the TUM RGBD Dataset [147] as well as on our own dataset. This paper makes three main contributions: (i) a novel extension of a spatio-temporal graph Laplacian based spectral clustering technique is introduced that incorporates depth information, (ii) a randomized strategy is proposed for the structure of a pixel neighborhood in the construction of the spatio-temporal graph Laplacian, and (iii) a context aware aggregation strategy that employs an object detection system is employed to find object-centric segmentations.

## 3.2 Related Work

Techniques for spatio-temporal segmentation can generally be grouped into several categories based on the assumptions made by the approach. One category of techniques is based on tracking and clustering a sparse number of point trajectories (up to 4% of the pixels) into temporally coherent moving regions [126, 17]. Another category is dense video segmentation in which the video is presented as a spatio-temporal volume. One popular method is hierarchical graph-based segmentation [59], which builds on graph-based figures segmentation proposed by [49], in which a volumetric video graph is over-segmented into spatio-temporal regions. These regions form a region graph which is used to merge similar regions repeatedly into larger regions and creates a hierarchical segmentation of the video. In [41], hierarchical mean shift analysis is introduced, which maps pixels of a video stack into a 7D feature space and then uses mean shift repeatedly to achieve hierarchical clustering of the points in the video. Another recent work [23], uses a generative probabilistic model to find temporally consistent superpixels in a video stream. Normalized cuts for graph partitioning has been widely used in both figures and video segmentation [137]. Since the underlying spectral clustering is computationally expensive, it is often impractical to use for large scale data. The Nyström method [160, 51], is a numerical solution for approximating p.s.d. matrices, which can be used to speed up spectral clustering. In [53], an efficient method is applied to randomly select a subset of pixels and estimate a rank-$k$ approximation of the similar-

ity matrix which enables the performance of spatio-temporal segmentation in memory on a single CPU. In [149], GPU clusters are necessary to perform spectral clustering naïvely on the original, unapproximated Laplacian. These described methods are based on only appearance cues.

In [158] a depth-adaptive supervoxel video segmentation technique is proposed that uses color and depth information to compute the supervoxels and then applies spectral clustering to partition them into spatio-temporal segments. Another algorithm [1] uses a GPU based parallel Metropolis algorithm to combine color and depth information and segment the figuress. They use real-time optical flow to warp obtain segment labels and track them to the next frame. Also in [68], an efficient hierarchical graph based segmentation is presented for segmenting 3D RGBD point clouds. They consider a moving window over a point cloud that segments the similar points into regions which later are merged together repeatedly. A bipartite graph matching is used to obtain the final segmentation at a given level of the hierarchy. Other methods for 3D segmentation of point clouds such as [70, 132, 146] do not consider time and are focused on static scenes, or require strong supervision [62].

## 3.3 Spatio-Temporal Segmentation

Our approach has three main steps, as shown in Figure 3.1. First, we randomly sample a subset of pixels in the video sequence and form a sparse affinity matrix, which is later used to estimate a low rank approximation of the full affinity matrix. Next, we perform spectral clustering on the affinity matrix to obtain an over-segmentation of the sequence. Finally, we apply ultra-metric contour maps [3] on the resulting over-segmentation to obtain a smaller number of regions.

### 3.3.1 Spatio-temporal Graph

For each frame in an RGBD video sequence, we create a similarity graph $G = (V, E, W)$, where each pixel is a vertex $v_i \in V$ and is connected to $v_j$ by an edge $e_{ij}$ if they are within distance $r$ from each other. The edge $e_{ij}$ is weighted by $w_{ij}$ that shows the similarity between pixels $v_i$ and

$v_j$. $W$ is defined as the following:

$$w_{ij} = \exp\left(\frac{-d^2(p_i, p_j)}{\sigma_p^2}\right) \tag{3.1}$$

Where $w_{ij} = 0$ for $i = j$, and $d(p_i, p_j)$ is a linear combination of the difference in intensity, color and spatial position of two pixels:

$$d(p_i, p_j) = \alpha \times d_c(p_i, p_j) + \beta \times d_l(p_i, p_j) \tag{3.2}$$

Also $\sigma_p$ is a scaling factor, and we set $\sigma_p = 0.01$, $\alpha = 0.4$ and $\beta = 0.6$ **a priori**. $d_c$ is the color distance of pixels $p_i$ and $p_j$, measured by the euclidean distance in the CIELAB color space, and $d_l$ is the euclidean distance between 3D position of $p_i$ and $p_j$ in camera coordinates, which is computed using a pinhole camera model:

$$xyz_p = depth_p \times (\frac{x - c_x}{f}, \frac{y - c_y}{f}, 1)^T \tag{3.3}$$

where $x$ and $y$ are the position of pixel $p$ in figures plane and $f$, $c_x$ and $c_y$ are the camera parameters. As mentioned earlier, we only measure the similarity of the pixels within distance $r$ in the figures plane due to the fact that it is more likely that nearby points in 3D with similar colors belong to the same region. Choosing the graph radius $r$, is a tradeoff between segmentation quality and computation cost. As discussed in [35], larger $r$ makes the quality of segmentation better and smaller $r$ makes the segmentation faster. If we consider a neighborhood around pixel $p$, the graph weights have low variance, if there is no edge falling between any two adjacent pixels in the neighborhood it therefore is redundant information. By considering larger $r$, the chances of having an edge in the neighborhood increases which leads to detecting contours and propagating the local cues across larger figuress regions, however the graph becomes denser and more expensive to compute. We alleviate this trade-off by considering a mid-range sparse neighborhood around pixels to benefit from grouping information of a larger area while low cost computation of the sparse neighborhood (Figure 3.2 and Figure 3.4, implementation details are provided in Section 6.5). The random pixel-neighborhood pattern in the third column of Figure 3.4 gives an empirically

superior balance between computational cost and accuracy than either a structured pattern or a dense neighborhood connectivity.

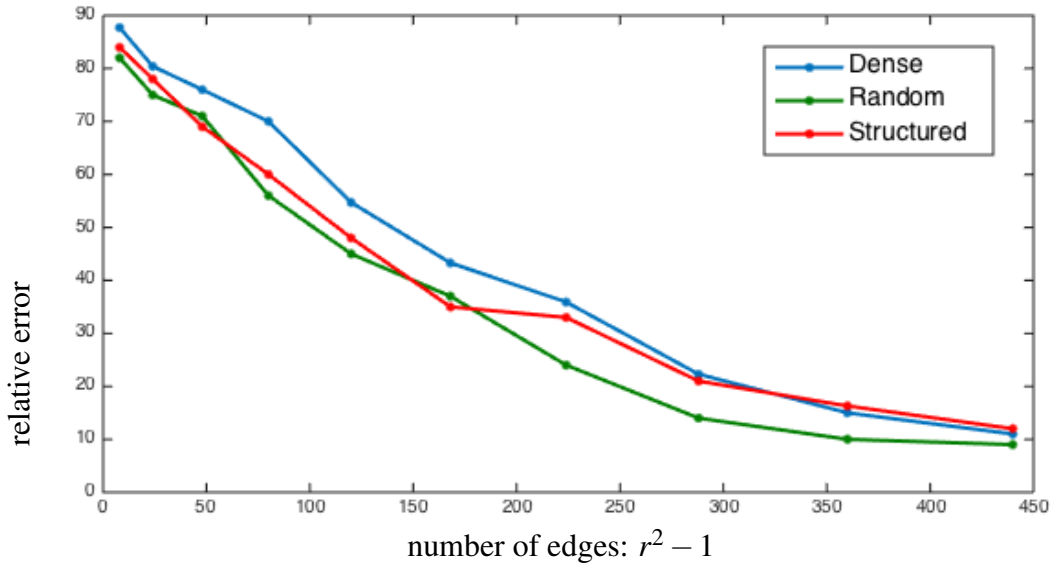


Figure 3.2: The error based on choice of neighborhood size $r$ = 3, 5, 7, 9, 11, 13, 15, 17, 19 and 21 for structured pattern, random pattern and dense pattern is shown in red, green and blue respectively. Note that the error is reported for the number of edges between the center pixel and the pixels in the neighborhood $r$. The random pixel neighborhood strategy consistently gives the best performance (cf. Figure 3.4).

Prior to computing the similarity between the pixels, we use a joint bilateral filter [152, 45] for smoothing the figures and removing the noise in color values. The joint bilateral filter controls the smoothing weight for color figures with regards to the depth figures and prevents color from flowing over the depth boundaries. The joint bilateral filter is defined as:

$$I_{p_c} = \frac{1}{k}\Sigma g_c(p_c - p\prime_c)g_d(p_d - p\prime_d)I\prime_{p_c} \tag{3.4}$$

where

$$k = \Sigma g_c(p_c - p\prime_c)g_d(p_d - p\prime_d) \tag{3.5}$$

is a normalization term and $p_c$ is the color value, $p_d$ is the depth value and $g$ denotes the Gaussian kernel. Here, kernel $g_c$ sets the weights based on the color difference of the pixels, while the edge stopping kernel $g_d$ computes proper weights for nearby pixels based on the depth differences.

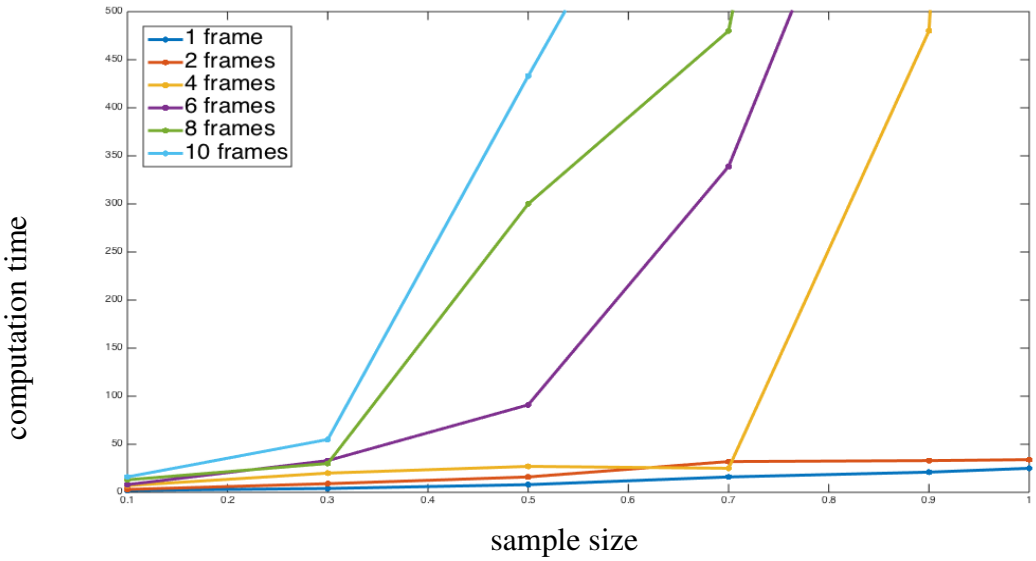


Figure 3.3: Computation time for eigen-decomposition of the normalized Laplacian is shown for different sample sizes in efficient Nyström method. Different colors represent different number of frames used to build the graph. (cf. Figure 3.5).

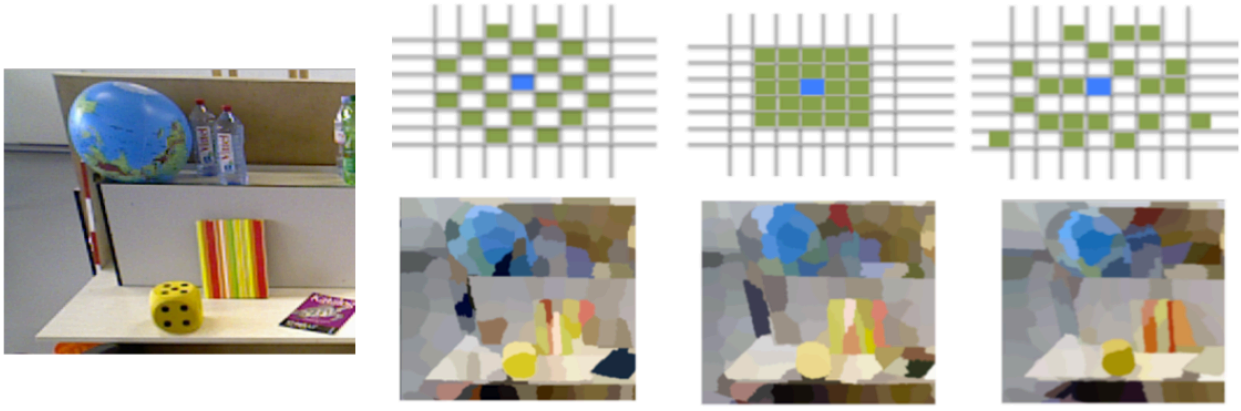

Figure 3.4: Random, dense, and structured pixel neighborhood patterns and the resulting spatio-temporal segmentations. In all cases, the number of edges, and thus the computational complexity, are the same.

After creating the graph for each frame in the RGBD sequence, we need to add the motion information and extend the graph to the spatio-temporal case. In this step, we use optical flow [96] to compute the motion vectors between frames. Since the optical flow vector implies that the source pixel at time $t-1$ belongs to the same region as the destination pixel in time $t$, we connect

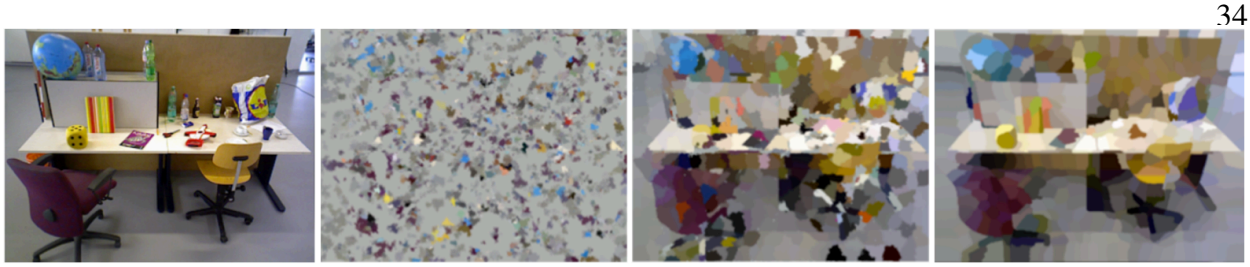

Figure 3.5: Over-segmentation quality for different sample size in Nyström method: original frame, sample size: 10%, 30% and 50%

pixel $(x,y)$ in frame $t$ to its 9 neighbors along the backward flow $(u,v)$ in frame $t-1$, *e.g.* $(x+u(x,y)+\delta_x, y+v(x,y)+\delta_y)$ for $\delta_x, \delta_y \in \{-1,0,1\}$, and then we compute the similarities between the connected pixels using Equation (3.1).

For processing long sequences, We use a window of 5 consecutive frames with one overlapping frame to build the spatio-temporal graph each time and then use the cluster centers obtained from over-segmentation of these frames to initialize the centroids of $k$-means in the next window.

### 3.3.2 Over-segmentation

Given the similarity graph $G$, we form the affinity matrix $W$ which is a sparse symmetric band matrix of size $n \times n$, $n$ is the total number of pixels in the video. $W$ has the following structure:

$$W = \begin{vmatrix} W_{11} & W_{12} & 0 & 0 & \dots & 0 \\ W_{21} & W_{22} & W_{23} & 0 & \dots & 0 \\ 0 & W_{32} & W_{33} & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \dots & W_{nn} \end{vmatrix} \tag{3.6}$$

where $W_{i,i}$ denotes the affinity matrix between pixels in frame $i$ and $W_{i,j}$ denotes the affinity matrix between the pixels in frame $i$ and frame $j$. After building the spatio-temporal graph, which encodes appearance,depth and motion information, the spectral clustering algorithm is applied to partition the graph and obtain the over-segmentation of the video. In short, for spectral clustering, first we

need to compute the normalized graph Laplacian of $W$ by $L = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ where $D$ is the diagonal degree matrix defined as $D_{ii} = \sum_{j=1}^{n} W_{ij}$. Then an eigen-solver is used to find the eigenvectors corresponding to the $k$ largest eigenvalues of the normalized Laplacian matrix. The complexity of the eigen decomposition of $L$ is $O(n^3)$ which makes it very expensive and impractical for large datasets like videos. Therefore we apply a time and space efficient Nyström method [93] similar to [53] that has a complexity of $O(nmk)$. The idea is to randomly select a subset of $m$ pixels in the video and form the affinity matrix for these selected pixels. Then the normalized Laplacian matrix $M$ is computed and used to estimate the eigenvectors of a rank-$k$ approximation of the normalized Laplacian matrix $L$ (Figure 3.5) for the entire set of pixels in the video. Finally another clustering algorithm such as $k$-means is used to cluster the estimated eigenvectors of the normalized Laplacian $L$. This will give us the cluster assignment for all of the pixels in the video. Further details are provided in [53]. This gives an over-segmentation of the video in the form of 3D superpixels that are coherent in terms of appearance and position in space and time. Note that in the context of this paper, the term superpixel has been used interchangeably with the term 3D superpixel.

### 3.3.3 Hierarchical Aggregation

The over-segmentation of the RGBD video produces 3D depth-labeled superpixels. For each of these superpixels we compute a feature vector $f$. The components of $f$ are the superpixel LAB color histogram $s_c$, superpixel mean surface normal $s_n$ and superpixel centroid $s_{xy}$. We start by defining a graph structure where each superpixel is a vertex in the graph and is connected to all neighboring superpixels with edges. The edges are weighted by the similarity between connected superpixels, based on the fact that similar superpixels have similar appearance, are spatially close and have normals that point to the same direction. We measure the similarity of neighboring superpixels $s_i$ and $s_j$ by:

$$S_{ij} = \exp\left(\frac{-d^2(s_i, s_j)}{\sigma_s^2}\right) \tag{3.7}$$

where

$$d(s_i, s_j) = \gamma \times d_c(s_i, s_j) + \delta \times d_n(s_i, sj) + \eta \times d_{xy}(s_i, s_j) \tag{3.8}$$

$d_c(s_i, s_j)$ is the $\chi^2$ distance between the color histograms, $d_n(s_i, s_j)$ measures the angle between surface normals, $d_{xy}(s_i, s_j)$ is the Euclidean distance of the superpixels centroids and $\sigma_s$ is a scaling factor. Also we set $\gamma = 0.4$, $\delta = 0.4$, $\eta = 0.2$ and $\sigma_s = 0.01$ **a priori**. Here, the intuition is that we want the superpixels that have similar color and surface normal to be grouped together but we put less emphasis on where in the figures they are located (for example superpixels of a background wall that is occluded with a foreground object should be grouped into one segment). After creating the neighborhood graph for superpixels, we use an approach similar to the ultra-metric contour map [3] to merge the superpixels repeatedly. We sort the edges in the neighborhood graph and start from the edge with highest value of similarity, and merge the two corresponding superpixels. Then we update the weights and repeat this process until there are no more edges. Also it is possible to make this process faster by using a bucket sort list in which we go thorough all the edges in one bucket and merge the corresponding superpixels before updating the neighborhood graph. This process gives us a hierarchy of segmentation in a bottom-up approach. We can obtain the final segmentation by using a threshold to stop merging the superpixels either when it reaches a certain level in the hierarchy or when a desired number of segments is obtained. Figure 3.6 presents the hierarchical aggregation results for different levels of segmentation.

Figure 3.6: Hierarchical Aggregation using an Ultra-Metric Contour Map approach is shown. Left column is the original frame taken from TUM RGBD Dataset [147], the second to fourth columns show different levels of segmentation in the hierarchy with maximum number of clusters: 400, 100, and 50 respectively

### 3.3.4 Parts-Based Aggregation

In this section, we introduce a novel method for merging the superpixels obtained from the over-segmentation by applying an object detection technique [48] based on mixtures of multiscale deformable part models. These models are trained using a discriminative process that requires bounding boxes. A DPM model for an object with $n$ parts is defined by a root filter and $n$ part filters. To detect an object in the figures, an overall score for each root location according to the best placement of the parts is computed. High-scoring root locations define detections while the locations of the parts that yield a high-scoring root location define a full object hypothesis with a bounding box. Given these bounding boxes and the filter scores, we want to segment out the object from the background. To do this, for each superpixel inside the bounding box, we first find the mean value of the scores given by the filter responses $s_{dpm}$. Then we build the superpixel

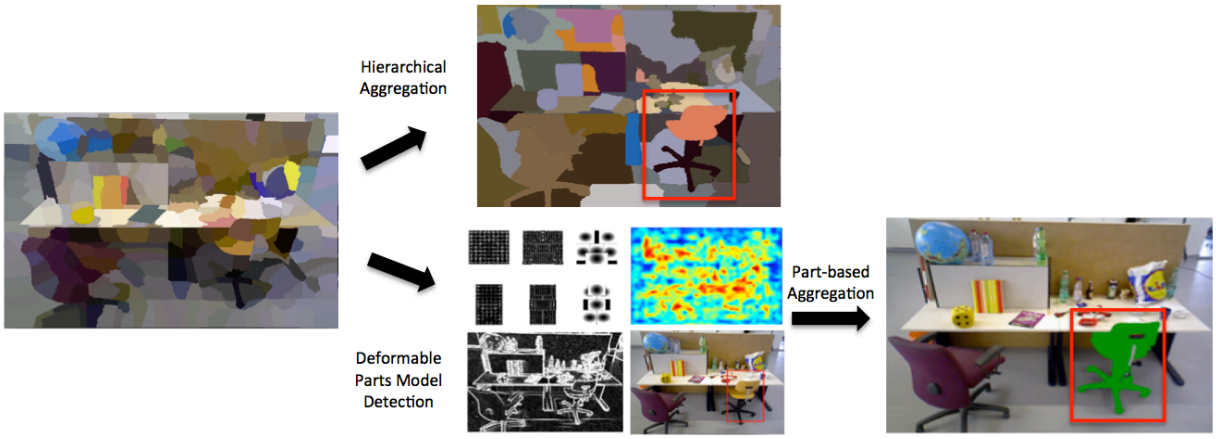


Figure 3.7: Hierarchical Aggregation vs. Parts-Based Aggregation Approach: (Top-row from left to right) shows the segmentation result for the chair using hierarchical aggregation, (Bottom-row from left to right) the root and part filters for "chair", the filter score for chair detection, detected edges using Sobel operator, prediction of boundary box and result of aggregating superpixels in the bounding box. Without the use of the context aware aggregation, the different articulated parts of the chair are misidentified as belonging to different objects. The incorporation of the parts based model enables the segmentation framework to correctly group these parts as a single object.

neighborhood graph and compute the edge weights using Equation (3.7) where

$$d(s_i, s_j) = \mu \times d_{dpm}(s_i, s_j) + \nu \times max_{x \in B}(edge(x)) \tag{3.9}$$

$d_{dpm}(s_i, s_j)$ is simply the difference between superpixel scores and $edge(x)$ is the edge strength at pixel $x$ on the boundary of two superpixels. We set $\mu = 0.6$ and $\nu = 0.4$ **a priori**. If the filter responses for two adjacent superpixels are high enough, we still want to merge them even if there is an edge between them, so we set a higher value for $\mu$. We also applied a Sobel operator to detect the edges using the gradient of the figures. Then the same approach as the previous section is used to merge the superpixels repeatedly. This method of aggregation can be used if one is interested in tracking a boundary accurate object in contrast to bounding box methods. To evaluate our results in this paper, however, we only used the hierarchical aggregation to obtain a dense segmentation of the sequence. Figure 3.7 shows the difference between hierarchical aggregation and parts-based aggregation.

## 3.4 Results

Several comparative experiments were run to assess the performance of the proposed method using publicly available RGB-D video sequences. In Table 3.1, 'RGB-D Video' indicates the proposed method. The second method called 'Depth two-stage' uses the two-stage strategy proposed in [68] to incorporate depth information in the spatio-temporal segmentation, while still using the same appearance based cues as our proposed method. The fundamental difference between this method and the proposed method is that instead of combining depth and color information linearly, first a segmentation of a sequence of frames using depth and motion information is obtained, and then an over-segmentation of the frames is done using color and motion information while respecting the depth boundaries from previous step. The third method called 'RGB Video [53]' uses similar appearance cues to those employed in our method, but does not take depth information into account. The fourth method called 'RGB Video [59]' uses a similar approach to the graph based figures segmentation 'RGB figures [49]', but extends it to the video setting by adding temporal information. These results show that combining depth information with color and temporal cues improves the segmentation result significantly and having an initial depth segmentation is not necessary. The presented method tends to maintain the depth boundaries of the objects and avoids merging them into other parts of the scene. Three further experiments have been performed: two sequences of size 100 frames and one sequence of size 200 frames have been chosen from NYU Depth Dataset [112] and one sequence of size 150 frames has been chosen from our own dataset Fig. 4.5. The qualitative results shown in Figures 3.4, 3.5 and 3.6 are sequences chosen from TUM RGBD Dataset [147]. Table 3.1 shows the detailed numerical comparison between our method and the rest of methods. Here, the density of all the methods is 100% which denotes the percentage of pixels for which a label is reported. The overall error is the number of mislabeled pixels over the total number of labeled pixels. The overall accuracy is the number of correctly labeled pixels over the total number of pixels. All pixels covering their assigned region in ground truth are counted as correctly labeled pixels, all others count as mislabeled pixels. The average error

Table 3.1: Comparison between the proposed method and existing methods in the literature.

| Method | Overall Error | Overall Accuracy | Average Error | Over segmentation | Extracted Objects |
|---|---|---|---|---|---|
| 2 sequences(100 frames) | | | | | |
| RGB-D Video | **9.8**% | **90.2**% | **15.1**% | 14 | 16 |
| Depth two-stage | 11.7% | 88.3% | 16.7% | 15.5 | 16 |
| RGB Video [53] | 14.3% | 85.7% | 17.2% | 19.2 | 14.3 |
| RGB Video [59] | 16.4% | 83.6% | 19.1% | 18.3 | 13 |
| RGB figures [49] | 21.7% | 78.3% | 23.7% | 24 | 10 |
| one sequence(150 frames) | | | | | |
| RGB-D Video | **11.4**% | **88.6**% | **17.4**% | 16.8 | 12.4 |
| Depth two-stage | 13.9% | 86.1% | 20.1% | 18 | 10.7 |
| RGB Video [53] | 16.3% | 83.7% | 21.4% | 19.2 | 9.3 |
| RGB Video [59] | 16.1% | 84.9% | 21.8% | 18.8 | 10 |
| RGB figures [49] | 23.4% | 76.6% | 25.2% | 26.3 | 7.8 |
| one sequence(200 frames) | | | | | |
| RGB-D Video | **12.7**% | **87.3**% | **18.9**% | 18.1 | 14.1 |
| Depth two-stage | 14.3% | 85.7% | 19.8% | 18.8 | 13 |
| RGB Video [53] | 15.8% | 84.2% | 22.0% | 19.8 | 11.1 |
| RGB Video [59] | 17.5% | 82.5% | 23.1% | 19.6 | 13 |
| RGB figures [49] | 24.7% | 75.3% | 28.2% | 25.8 | 10.8 |

is similar to the overall error but computed separately for each region first and then the average across all the regions is reported. The over-segmentation error indicates the number of clusters that need to be merged to fit the ground truth regions. Also the number of regions with less than 10% error is reported as the extracted objects. The same evaluation method has been previously used in [53, 17]. We can see that an integrated system incorporating RGB-D video gives the best results by a substantial margin, with an approximately 1/3 reduction in error rate vs. the best performing video segmentation competitor [53]. Furthermore, we show that the direct inclusion of depth in the graph segmentation reduces the error rate by a further 16% or more as compared with our implementation that uses the two-stage method proposed in [68]. Also Figure 3.2 and 3.4, compare the error for different graph radius sizes and patterns. Observe that the best result is achieved by choosing a larger radius with a sparse random pattern. Note, in all of our experiments, the graph radius is set to $r = 5$, which means that the center pixel is connected to 120 pixels in an 11x11 neighborhood (e.g. in the dense pattern). For both structured and random neighborhood pattern, the center pixel is connected to the same number of pixels as dense pattern but each has a different structure as shown in Fig. 3.4. In all of the experiments in this paper we use the random neighborhood pattern to build the spatio-temporal graph. Since a random subset of pixels is used in the video to approximate the normalized Laplacian matrix $L$, in Figure 3.3 and 3.5 the overall computation time and accuracy is reported for different sample sizes. It is obvious that selecting more pixels improves the segmentation result, however there is a trade-off between accuracy and computation cost. For all the experiments in this paper a random subset of 50% of pixels is chosen. On average, the runtime for the segmentation was about 20 minutes for 100 frames of size $640 \times 480$ on a single CPU. Finally, Figure 4.5 presents a qualitative comparison between the proposed method and other state-of-the-art methods based on superpixel segmentation.

## 3.5 Discussion

Experiments systematically show an improvement using the proposed framework, both from a quantitative, and from a qualitative perspective. Results show a reduction in the overall error

rate by approximately one third on both the first and second experiments. Of high importance to unsupervised object detection, the number of extracted objects is increased in both experiments as well. On the second experiment, the average error increases by one percent, but the first experiment shows an average error decrease of approximately one quarter. This disparity can be accounted for by the different distribution of segment sizes in the two sequences, and the overall error is consistently decreased by our method while using only 50% of randomly chosen pixels from the entire video. The random pixel neighborhood pattern gave the best result, and did so at a reduced computational cost compared with the dense segmentation

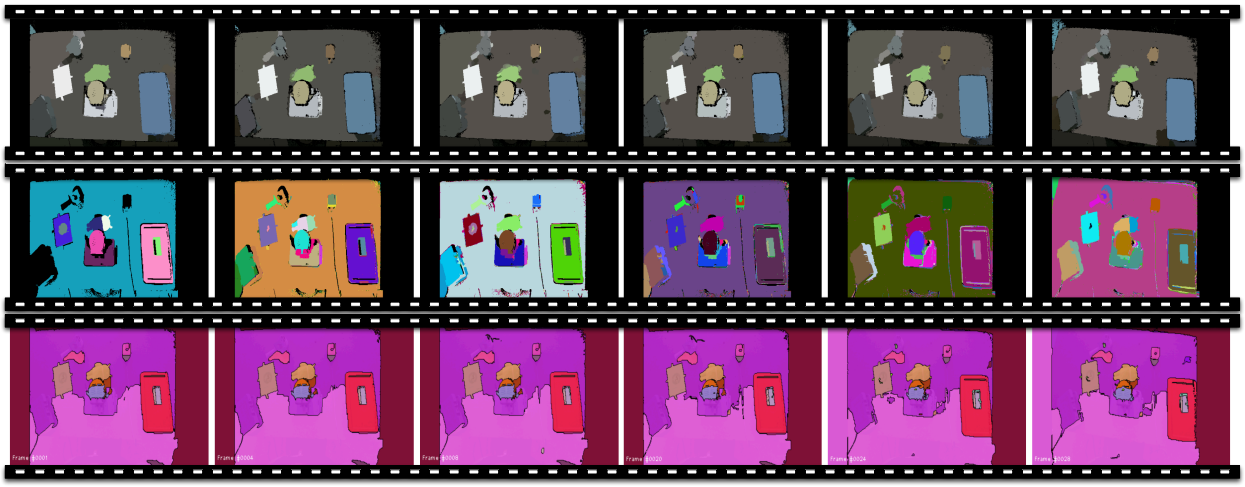

Figure 3.8: The original frames are added

Segmentation results by the proposed method for a sequence of size 6 shown in the last row. The first row: our temporal RGB-D, the second row: only RGB [49] and the third row: temporal RGB [59]. The proposed method has the best performance in terms of precision and recall of object pixels (scored using motion-verified object boundaries).

Figure 3.2 and Figure 3.4. A structured placement did not perform as well, indicating that the random structure of the pattern gives a good spatial coverage of the neighborhood. Also incorporating depth and color in spatio-temporal graph makes our method much more efficient compared to 'Depth two-stage' method inspired by [68], since we only apply spectral clustering once to

obtain the segmentation whereas the other method needs to perform spectral clustering twice. The effect of the Nyström method is demonstrated in Figure 3.5 and Figure 3.3. We see that the approximation degrades gracefully with the sparsity, and that qualitatively sensible segmentations are achieved at a sampling rate of 50% or lower. Also the computation time for spectral clustering increases drastically with respect to the number of pixels. However, we can observe that choosing a smaller size window for processing the frames in conjunction with choosing a larger sample size gives better accuracy and less computation time.

## 3.6 Summary

In this work, we have proposed a novel framework for spatio-temporal segmentation from RGBD videos that incorporates a rich set of appearance, motion, object-part and depth cues while maintaining feasible computation on a single core.

The system employs appearance cues based on low-level information such as intensity and color, while also incorporating high-level information from a deformable parts model (DPM). In this way, we are able to compute **object centric** spatio-temporal segmentations based on semantic object detection. One interesting area of future work based on the proposed idea is to use this technique to segment and track the object detected by DPM and improve the performance of object detection where DPM fails to detect the object due to different factors in the scene.

Several computational innovations are incorporated in order to make feasible a dense spatio-temporal segmentation on a single core. First, a Nyström approximation is used based that samples pixels from the dense optic-flow graph. This strategy combines the computational efficiency of the Nyström approximation with the accuracy of an intelligent sampling strategy that accounts for the temporal structure of the problem. Additionally, we have incorporated a structured connectivity pattern for construction of the graph Laplacian. This strategy balances the high accuracy achieved by a connectivity with high spatial extent with the computational imperative of a similarity matrix with low degree. The resulting method has a computational efficiency sufficiently low to enable computation on a single core. Future work will be aimed at decreasing the computation time and

use an online approach for streaming the video data.

The proposed framework has been validated using two benchmark datasets: the NYU Depth Dataset [112] and the TUM RGBD Dataset [147]. On both datasets, results show across the board improvements on a wide range of performance metrics. These include error and accuracy percentages, the degree of over-segmentation, and the number of objects extracted. In summary, we have shown that the addition of depth information, a structured connectivity pattern for the graph Laplacian, and a deformable parts model all conspire to substantially improve spatio-temporal segmentation of RGB-D video.

# Chapter 4

## Simultaneous Localization, Mapping, and Manipulation for Unsupervised Object Discovery

In this chapter, we present an example of simultaneous object DDTR by using the appearance cue for candidate objects discovery and the motion cues for objects verification (via a robot manipulator). The system performs dense 3D simultaneous localization and mapping concurrently with unsupervised object discovery. Putative objects that are spatially and visually coherent are manipulated by the robot to gain additional motion cues. The robot uses appearance alone, followed by structure and motion cues, to jointly discover, verify, learn and improve models of objects. Induced motion segmentation reinforces learned models which are represented implicitly as 2D and 3D level sets to capture both shape and appearance. We compare three different approaches for appearance-based object discovery and find that a novel form of spatio-temporal super-pixels gives the highest quality candidate object models in terms of precision and recall. Live experiments with a Baxter robot demonstrate a holistic pipeline capable of automatic discovery, verification, detection, tracking and reconstruction of unknown objects.

We present an unsupervised framework for simultaneous appearance-based object discovery, detection, tracking and reconstruction using RGBD cameras and a robot manipulator. The system performs dense 3D simultaneous localization and mapping concurrently with unsupervised object discovery. Putative objects that are spatially and visually coherent are manipulated by the robot to gain additional motion-cues. The robot uses appearance alone, followed by structure and motion cues, to jointly discover, verify, learn and improve models of objects. Induced motion segmentation reinforces learned models which are represented implicitly as 2D and 3D level sets to

capture both shape and appearance. We compare three different approaches for appearance-based object discovery and find that a novel form of spatio-temporal super-pixels gives the highest quality candidate object models in terms of precision and recall. Live experiments with a Baxter robot demonstrate a holistic pipeline capable of automatic discovery, verification, detection, tracking and reconstruction of unknown objects.

### 4.0.1 Introduction

Automatic object model acquisition is a fundamental task in robotics. Recent state-of-the-art object detection and recognition systems are based on pre-trained detectors [14, 44, 67, 107] and features [111, 87]. For object learning, most techniques reconstruct sparse 2D models [66, 32] together with dense 3D shape models. The focus of this paper is an unsupervised pipeline for discovering unknown objects in a scene without any prior information and using robots with manipulators to verify and reinforce model acquisition by robot-object interaction (e.g. poking, grasping). Hence, a pre-trained object detection technique is not suitable here – indeed, the proposed method could be used to train such systems.

2D object appearance is highly informative for image based spatio-temporal segmentation [60, 53]. Typically, per-frame image segmentation results are combined with temporal information via 2D segment tracking. Instead, we find that using appearance and dense spatio-temporal cues produces better image segmentation results, which in turn yield a high quality set of likely candidate objects. However, these cues are not sufficient for the verification of object-hood, especially for static objects [104, 105]. In the proposed approach, a mobile robot relies on dense 3D SLAM to manipulate putative objects, thereby adding possible motion cues to verify object-hood.

Once an object has been verified by poking or grasping, a dense detection, tracking and reconstruction technique is applied to learn 2D and 3D implicit models of appearance and shape. In subsequent steps, learned models are automatically detected, tracked, and refined.

Related work demonstrates dense tracking and reconstruction to generate 3D models of static or moving objects [128, 40, 10]. However, these techniques require a bounding box (or a pre-

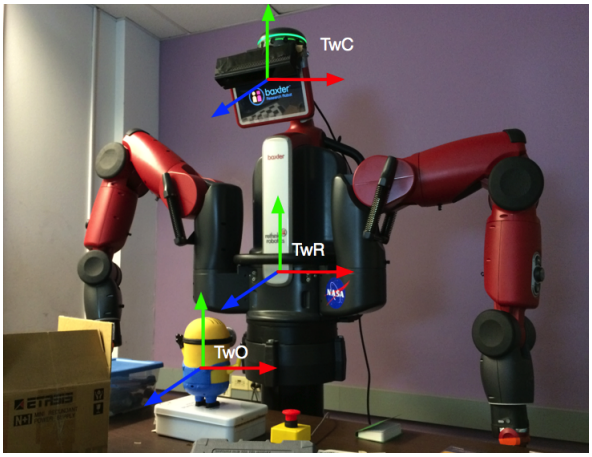


Figure 4.1: The dual-arm Baxter robot equipped with a RGB-D camera automatically discovers and learns dense 3D model of unknown objects in the scene by manipulation.

trained object detector) as system input to define the target.

This paper extends prior work on **unsupervised object discovery, detection, tracking and reconstruction** (DDTR)[104] with a novel form of appearance based object discovery and also uses a robot manipulator to verify candidate objects. A complete dense 3D model is generated once the robot has full observation of an object.

The proposed framework is different from most existing frameworks as: 1) it does not require prior information for object discovery as [107, 14]; 2) it builds models and also subsequently detects, tracks and refines **dense** 2D and 3D models which is different from [14, 154, 44, 67, 107, 110] [32]; 3) it learns high quality 3D models of unknown objects via robot-object interaction; 4) it is also a simultaneous localization and mapping system (SLAM), which enables the potential of autonomous object discovery; 5) the final learned model of an object can be used for future object detection tasks, and shared with other robots.

The remainder of this paper is structured as follows: Section 2 briefly covers preliminaries and introduces our approach. In the following sections we cover appearance-based object discovery, motion-based objects verification, tracking, reconstruction and system integration. Section 6 tests and discusses the system performance. Section 7 addresses failure cases and future work and Section 8 draws conclusions.

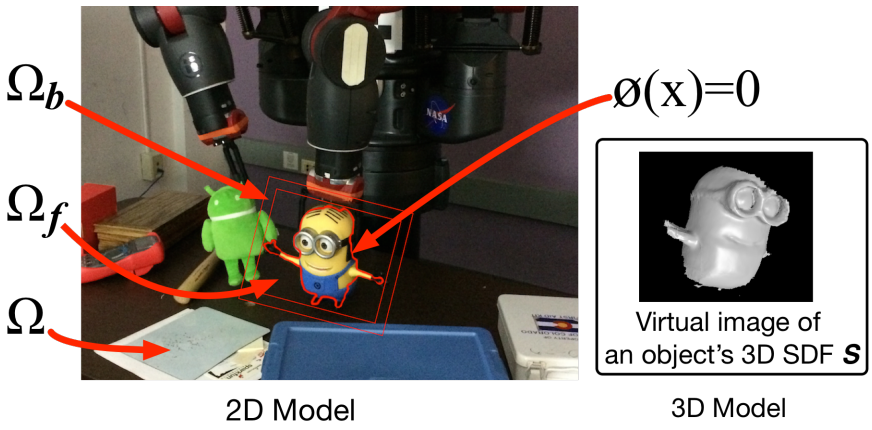


Figure 4.2: An Object is represented implicitly with a 2D and a 3D model. The 2D model is represented with a RGB histogram and a 2D SDF $\Phi(x)$. The zero level set of $\Phi(x)$ segments the image domain $\Omega$ into a foreground $\Omega_f$ and a background $\Omega_b$. The 3D model of the object is represented by the 3D SDF and can be rendered via ray casting and phong shading.

### 4.0.2 Overview

#### 4.0.2.1 Preliminaries

The proposed system (Fig.4.1) uses a dual-arm Baxter robot with a calibrated RGBD camera $\mathscr{C}$ mounted on its head, where the world pose of the center of the robot is $T_{wR} \in SE(3)$. The camera $\mathscr{C}$ is calibrated with respect to the robot center, with extrinsic matrix $T_{wC} \in SE(3)$, and intrinsic matrix $K_i$. The transformation between $T_{wR}$ and $T_{wC}$ is calibrated and considered as a known parameter. An object in the scene is noted as $O_i$, where its world pose $T_{wO}$ is noted as the center point of the object, and can be computed by $T_{wO} = T_{wC} \oplus T_{CO}$.

A 3D point in the camera frame is denoted as $x_c = (x,y,z)^\top$ and projects onto the image plane as $\pi(x_c) = (x/z, y/z)^\top$ (i.e., dehomogenization). A 2D point $\mathbf{u} = (u,v)^\top$ in the image plane can be back-projected to a 3D point by $\chi = \frac{1}{d} \cdot K_i^{-1} \cdot (\mathbf{u};1)$ with a depth value $d$. The system uses $T_{wr}^i$ and $T_{wl}^i$ to represent the camera pose in the reference frame (time $k-1$) and the live frame (time $k$).

We denote learned objects as $O = O_\emptyset, O_1, O_2, \cdots, O_n$. Each object is represented by both a 2D and a 3D model. Figure 4.2 shows the representation of an object's 2D and 3D model.

The system uses shape (2D contour) and appearance (RGB histogram) to represent an object's 2D model. A level-set embedding function $\Phi(x_i)$, namely a 2D signed distance function (SDF), is used to implicitly represent an object's 2D contour (shape). $x_i = x_{i1}, x_{i2} \cdots, x_{in}$ is the set of pixel locations in the coordinate frame of $O_i$. The zero level-set of the object's 2D SDF $\Phi(x) = 0$ is used to represent the shape (2D contour). The 2D contour $\Phi(x) = 0$ segments the image domain $\Omega$ into a foreground $\Omega_f$ and a background $\Omega_b$, with appearance models $M_f$ and $M_b$ respectively. Notice that $\Omega_f^\Phi$ and $\Omega_b^\Phi$ is used to denote the foreground (pixels inside $\Phi(x) = 0$) and the background segments by the zero level-set of $\Phi(x)$.

The system uses the truncated signed distance function (TSDF) to represent the 3D model $\mathscr{S}_i$ of an object $O_i$, Where the shape (geometry) of an object can be rendered by ray casting $\mathscr{S}_i$ in different views. The 3D SDF $\mathscr{S}$ is stored as a $n \times n \times n$ volume cube, where $n$ is the number of voxels in one dimension. The size of each voxel is $v_m = \frac{2r}{n}$, where $r$ is the radius of the volume. The 3D SDF is initialized and updated by SDF fusion.

Notice that the 2D contour ($\Phi(x) = 0$) is actually the boundary of the projection of $\mathscr{S}_i$. By using the 2D and 3D models, the system represents objects at different levels and performs 2D & 3D tracking and reconstruction Simultaneously during object learning.

#### 4.0.2.2 System Structure

The proposed pipeline goes through the following stages (Fig 4.3):

**Initialization**. The system is initialized by creating a 3D model of the first frame (noted as the dominant object $O_\emptyset$) from the input RGBD video. We use a $512 \times 512 \times 512$ volume for $\mathscr{S}_i$.

**Appearance-based Object Discovery**. For the first $200-300$ frames, the robot moves its head (the camera) to explore the scene which collects enough information for the appearance-based object discovery approach. While the robot explores the scene, the system tracks and updates $O_\emptyset$ frame by frame, and discovers (Section 4.1.0.1) a set of candidate object contours $C' = C'_1, C'_2, \cdots, C'_n$ by the appearance-based object discovery approach.

**Robotic Manipulation**. Given a candidate objects contour set $C'$, the system verifies the

candidate object $O_i^{'}$ (defined by $C_i^{'} \in C^{'}$) by applying motion (grasping, poking, etc) to $O_i^{'}$. This allow the system generate a verified objects set $O = O_1, O_2 \cdots, O_n$.

**Motion-based Object Verification**. For a verified object $O_j \in O$, the system generates its contour $C_j$ by extracting portions of the scene that fail to match with $O_\emptyset$ via the ICP+RGB algorithm. The system saves $C_j$ to the discovered objects contour set $C$.

**2D Tracking and 2D Reconstruction**. For a discovered object $O_j$ with a contour $C_j \in C$, the system matches $C_j$ with $O_i$ by the appearance, shape and motion cues. This 2D tracking result gives a rough pose estimation of $O_j$ in the image domain of $C_j$ where $C_j$ defines a foreground domain $\Omega_f^j$. 2D reconstruction is then achieved by updating $\Phi(x_i)$ to $\Phi^{'}(x_i)$ in $\Omega_f^j$ via LSE.

**3D Tracking and 3D Reconstruction**. The system uses the ICP+RGB pose estimator for 3D tracking, which estimates the relative camera pose between $\Omega_f^{\Phi}$ and $\Omega_f^{\Phi'}$. Once $O_i$ is tracked, the system updates $\mathscr{S}_i$ by fusing every pixel from $\Omega_f^{\Phi'}$ into $\mathscr{S}_i$ via SDF fusion.

## 4.1 Unsupervised Object Discovery

The system uses an appearance driven, motion verification pipeline for unsupervised object discovery.

#### 4.1.0.1 Appearance-Based Object Discovery

The system uses [53] to discover candidate objects in the RGB video sequence. This technique generalizes a graph-based image segmentation to the spatio-temporal case.

In short, given a video sequence, a spatio-temporal graph $G = (V, E, W)$ is constructed, where the graph nodes $V$ are the pixels in the video and are connected by edge $E$ if they are within distance $r$ from each other in each frame, and $W$ measures the similarity of pixels connected by an edge. Also, optical flow is used to add temporal motion information to the graph by connecting pixel $(x, y)$ in frame $t$ to its 9 neighbors along the backward flow $(u, v)$ in frame $t-1$. Affinities are calculated between pixels that are connected in the same frame and also between frames. The

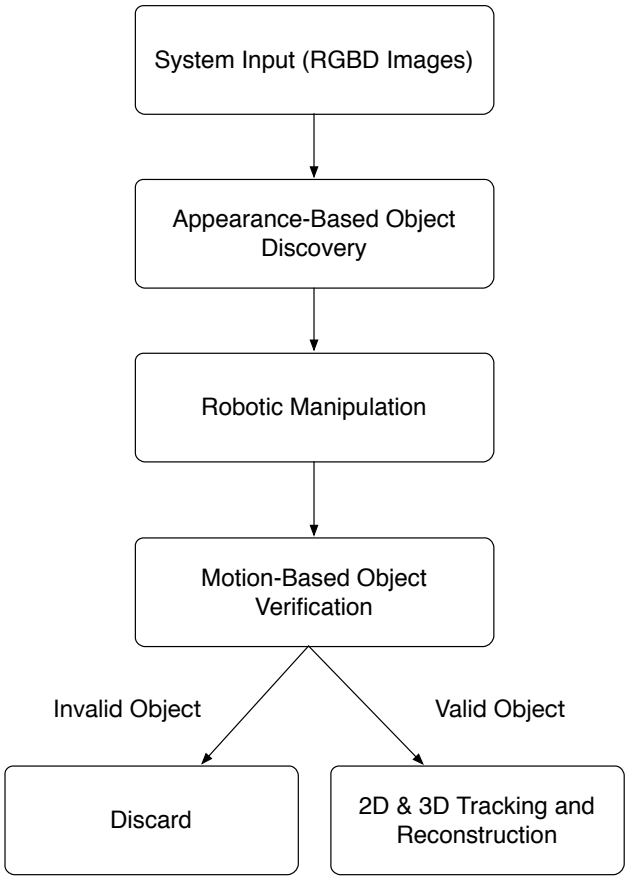


Figure 4.3: System flow chart

overall affinity matrix is a sparse symmetric block diagonal matrix that has the following structure:

$$W = \begin{bmatrix} W_{11} & W_{12} & \dots & 0 \\ W_{21} & W_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & W_{nn} \end{bmatrix} \tag{4.1}$$

Where $W_{ii}$ is the affinity matrix for frame 1, and $W_{ij}$ is the affinity matrix between frame $i$ and frame $j$. Then spectral clustering is applied to partition the affinity graph $W$ by computing its $k$ eigenvectors corresponding to the $k$ largest eigenvalues. However to make this process more efficient, only a randomly selected subset of pixels in the video is used to estimate a rank-k approximation of normalized affinity matrix $L = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, where $D$ is the diagonal degree matrix defined as $D_{ii} = \sum_{j=1}^{n} W_{ij}$. Next, k-means is used to cluster the eigenvectors which gives an over-segmentation of the video in the form of 3D superpixels. Finally, a post-processing step is applied to merge the 3D superpixels and obtain the final segmentation.

In conclusion, given a RGB image $I_k$, the appearance-based spatio-temporal object discovery approach $\mathscr{D}(\cdot)$ generates a candidate objects set $C^{'} = C_1^{'}, C_2^{'}, \cdots, C_p^{'}$:

$$C^{'} = \mathscr{D}(I), I = I_1, I_2 \cdots I_K \tag{4.2}$$

#### 4.1.0.2 Robotic Manipulation

Given a candidate objects set $C^{'}$, the system (robot and the arms) uses a motion cue to verify the validation of an object. The system randomly selects an object and then uses the robot arms to move it (grasping or poking). To interact with the object, the object pose $T_{wO}$ has to be known, which can be computed by:

$$T_{wO} = T_{wC} \oplus T_{CO} \tag{4.3}$$

where $T_{CO}$ is estimated by the relative pose between the central point of $C_i^{'}$ and the camera. The size of the object can be approximated by estimating the size of $C_i^{'}$ in $x, y, z$ direction. Given the

size, a rectangular cuboid is used to approximate the geometry of the unknown object. Given the object's pose $T_{wO}$, a grasp generator determines what grasp and pre-grasp positions are necessary to move the object. An inverse kinematics solver is used to evaluate the feasibility of the generated candidate grasps. If no valid grasp is found due to the geometry or reach-ability of the object, a poking start and end position is generated instead. A joint trajectory to the pre-grasp or poking start position is generated using the motion planning framework [31]. The trajectory is then executed by the robot. The robot then performs a straight line motion from the pre-grasp position to the grasp position, or from the poking start to poking end position.

If grasping, the end effector constrains the object and lifts it to within proper range of the camera. A pre-defined trajectory of the arm moves (rotates & translates) the object through the scene under the observation of the camera (head mounted). This trajectory is designed to optimize the views of the object that can be observed which allows the system to capture enough data for object learning (tracking and reconstruction).

#### 4.1.0.3 Motion-Based Object Verification

While the robot is manipulating the objects, the system tracks $O_\emptyset$ by estimating the relative pose $T_{rl}^{\emptyset}$ of the camera between the reference frame $I_r$ and the live frame $I_l$ via the ICP+RGB algorithm. A virtual image $I_v$ of $O_\emptyset$ can then be generated by:

$$I_v = \Upsilon(\mathscr{S}_\emptyset, T_{wl}^{\emptyset}) \tag{4.4}$$

where $\Upsilon(\cdot)$ is the ray casting operator (Section 4.2.0.4). If any objects have relative motion against $O_\emptyset$, the system can discover them by extracting portions of the scene that fail to track with $O_\emptyset$:

$$I_o = I_v - I_l \tag{4.5}$$

By searching the disjoint contours with a minimum number of valid pixels $d_1$ in $I_o$, a contour set $C$ of candidate objects can be obtained:

$$C = \Gamma(I_v - I_l) = \Gamma(\Upsilon(\mathscr{S}_\emptyset, T_{wl}^{\emptyset}) - I_l), C = C_1, \cdots, C_p \tag{4.6}$$

Here $\Gamma(\cdot)$ detects disjoint contours in an image [153].

The motion-based objects verification pipeline allows the system to verify objects from the candidate objects set and discard negative object discovery results if the expected motion between the candidate object and the dominant object does not happen. With this process, the system generates a verified objects set $C$, which will be used for object learning.

## 4.2 Tracking and Reconstruction

For a discovered (verified) object, the proposed system tracks and reconstructs it in 2D and 3D simultaneously.

#### 4.2.0.1 2D Tracking

For $C_j \in C$ that matches (tracks) with $O_j \in O$ by the appearance, shape and motion cues [104], $C_j$ defines an image domain $\Omega^j$:

$$\Omega^j = \mathscr{G}(C_j) \tag{4.7}$$

Here, $\mathscr{G}(C_j)$ inflates the image region defined by $C_j$. Notice that the 2D tracking process only gives a rough tracking result between $C_j$ and $O_j$ in the 2D image domain.

#### 4.2.0.2 2D Reconstruction.

Given a discovered object contour $C_i \in C$ (matches with an object $O_i$), the system does not update $\mathscr{S}_i$ with $C_i$ directly. Instead, it updates the 2D SDF of $O_i$ by level-set-evolution first, and then uses the updated 2D SDF to track and reconstruct $O_i$ in 3D space.

The system avoids updating $\mathscr{S}_i$ with $C_i$ directly because: 1) the precision of the $C_i$ is not guaranteed. Noisy information may be introduced from the object discovery approach to $C_i$, which would hamper the final 3D reconstruction result if the system updates $\mathscr{S}_i$ with $C_i$ directly. 2) Multiple objects may be overlapping with each other, but it is necessary to extract the precise

contours from $C_i$ for each object. 3) The system needs to update the object's 3D model by fusing pixels that have not yet been observed, where $C_i$ only matches against the known object pixels. However, LSE solves this by updating the object's 2D model based on its previous 2D model in $C_i$.

$$\Phi^{'}(x_i) = \mathscr{L}(\Phi(x_i), \Omega^i) \tag{4.8}$$

Here $\mathscr{L}(\cdot)$ is the level-set-evolution (LSE) operation [104] that updates the object's 2D SDF $\Phi(x_i)$ to $\Phi(x_i)'$ in $\Omega^i = \mathscr{G}(C_i)$. The foreground (in the RGBD space) defines by $\Phi(x_i)'$ is $\Phi^{'}_f(x_i)$

#### 4.2.0.3 3D Tracking.

Once the 2D model of an object is updated via LSE, the system tracks $\Phi^{'}_f(x_i)$ against $\Phi_f(x_i)$ for 3D tracking. This is done by estimating the relative camera pose $T^i_{rl}$ between $T^i_{wr}$ (pose of $\Phi_f(x_i)$) and $T^i_{wl}$ (pose of $\Phi^{'}_f(x_i)$) with the ICP+RGB pose estimator $\mathscr{E}(\cdot)$. This combined pose estimator $\mathscr{E}(\cdot)$ allows robust tracking for objects with either complex appearance or shape (geometry).

$$T^i_{rl} = \mathscr{E}(\Phi_f(x_i), \Phi^{'}_f(x_i)) \tag{4.9}$$

#### 4.2.0.4 3D Reconstruction.

For an object $O_i$ with the updated 2D SDF $\Phi^{'}(x_i)$, the system updates $\mathscr{S}_i$ by fusing every valid point $\chi = (x, y, z) \in \Phi^{'}_f(x_i)$ into $\mathscr{S}_i$:

$$\mathscr{S}^{'}_i = \mathscr{F}(\mathscr{S}_i, \Phi^{'}_f(x_i), T_{wl}) \tag{4.10}$$

Here, $\mathscr{F}(\cdot)$ is the SDF Fusion operation. $T_{wl}$ is the global pose of $\Phi^{'}_f(x_i)$. The system also stores intensity in $\mathscr{S}^{'}_i$, where it can render the zero level set surface of $\mathscr{S}$ and generate a virtual gray $I^g_v$ and depth image $I^d_v$ by ray casting $\Upsilon(\cdot)$ [114]:

$$I_v = \Upsilon(\mathscr{S}, T_{wc}), I_v = I^g_v \cup I^d_v \tag{4.11}$$

where $T_{wc}$ is the pose of the virtual camera.

### 4.2.1 System Integration

Given input RGBD images $I$, the system discovers a candidate objects set $C^{'}$ in $I$ via the appearance-based spatio-temporal object discovery approach $\mathscr{D}(I)$:

$$C^{'} = \mathscr{D}(I), C^{'} = C_1^{'}, C_2^{'}, \cdots, C_n^{'} \tag{4.12}$$

For $C_i^{'} \in C^{'}$, the system verifies it by the motion cue (via robot manipulation) and produces a verified objects set $C$.

$$C = \Gamma(\Upsilon(\mathscr{S}_{\emptyset}, T_{wl}^{\emptyset}) - I_l), C = C_1, \cdots, C_n \tag{4.13}$$

Here, $C_j \in C$ defines an image domain $\Omega^j$ that matches with $O_i$:

$$\Omega^j = \mathscr{G}(C_i) \tag{4.14}$$

where $O_i$ has camera pose $T_{wr}^i$, 2D SDF $\Phi(x_i)$ and 3D SDF $\mathscr{S}_i$ in the reference frame. $\Phi(x_i)$ can be updated to $\Phi^{'}(x_i)$ by LSE $\mathscr{L}(\cdot)$.

$$\Phi^{'}(x_i) = \mathscr{L}(\Phi(x_i), \Omega^j) = \mathscr{L}(\Phi(x_i), \mathscr{G}(C_i)) \tag{4.15}$$

where the foreground defines by $\Phi^{'}(x_i)$ is:

$$\Phi_f^{'}(x_i) = \mathscr{H}(\Phi^{'}(x_i)) = \mathscr{H}(\mathscr{L}(\Phi(x_i), \mathscr{G}(C_i))) \tag{4.16}$$

here, $\mathscr{H}(\cdot)$ extract pixels inside $\Phi^{'}(x_i) = 0$. Now, 3D tracking is processed by estimating the relative pose $T_{rl}^i$ between $\Phi_f(x_i)$ and $\Phi_f^{'}(x_i)$ via the ICP+RGB pose estimator $\mathscr{E}(\cdot)$:

$$T_{rl}^i = \mathscr{E}(\Phi_f(x_i), \Phi_f^{'}(x_i)) \tag{4.17}$$

3D Reconstruction is then achieved by updating $\mathscr{S}_i$ with $\Phi_f^{'}(x_i)$ by SDF Fusion $\mathscr{F}(\cdot)$:

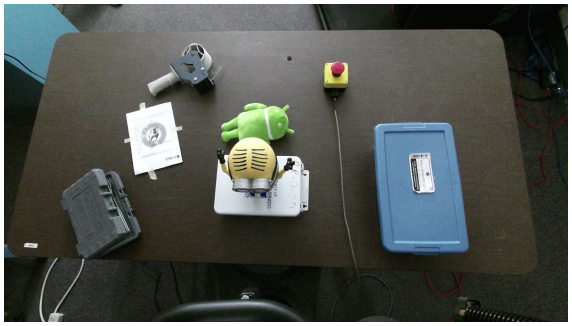

Figure 4.4: Test scenario with objects for discovery and modeling by manipulation. The white piece of paper attached to the the desk is considered as a part of the desk (not a separate object) to test the object discovery pipeline.

$$\mathscr{S}_i' = \mathscr{F}(\mathscr{S}_i, \Phi_f'(x_i), T_{wl}^i) = \mathscr{F}(\mathscr{S}_i, \mathscr{H}(\mathscr{L}(\Phi(x_i), \mathscr{G}(C_i))), \\ T_{wr}^i \oplus \mathscr{E}(\Phi_f(x_i), \mathscr{L}(\Phi(x_i), \mathscr{G}(C_i)))) \tag{4.18}$$

Equation 4.18 describes how unsupervised object discovery and learning is unified by an appearance driven ($C'$), motion verification ($C$), tracking ($T_{wr}^i$) and reconstruction ($\Phi(x_i)$ and $\mathscr{S}_i$) pipeline. Here, the optimization of the whole system is equivalent to the optimization of each subsystem.

### 4.2.2 Results

We evaluate the system in an unstructured scenario with different sizes, shapes and colors of objects. Notice that the white piece of paper attached to the middle-left side of the desk is considered as a part of the desk (not a separate object), which is used to test the robustness of the system. The system is tested with RGBD video streams captured at 20 FPS by a Kinect v2 camera. The camera is well calibrated and the RGB and the depth images are aligned during the experiment. Fig. 4.4 shows an example of the test scenario.

#### 4.2.2.1 Appearance-based object discovery

We compare our appearance-based object discovery method with two state-of-the-art object discovery approaches [49, 60] by pan/tilt the robot's head around the scene. Fig. 4.5 shows an

example of the final discovered objects. As Fig. 4.5 shows, [49] is able to generate good object discovery results (Fig. 4.5) but its performance is not stable and robust. Meanwhile, [60] generates stable object discovery results but the quality is not guaranteed (fail to discover the gray box in all sequences). The proposed method instead generates stable and precise object discovery results, which gives a good hint for the motion-based object verification pipeline.

Appearance-based object discovery methods are usually used for image segmentation, which are not designed for indoor scene object discovery. To evaluate such methods, we use a pixel level per-frame precision and recall curve approach. Here, the ground truth 3D model $\mathscr{S}_i$ of an object $O_i$ is approximated once the system learned the complete 3D model of the object, as the motion-based object discovery and learning pipeline is very robust and precise [104]. In each subsequent frames, the system uses the virtual image ($D_v$) of $\mathscr{S}_i$ to approximate the ground truth label and uses $k$ different appearance-based object discovery methods to generate different discovered objects images $D_l^k$. The performance of the system is evaluated at pixel level by comparing each corresponding object's pixel $p$ in $D_v$ and $D_l^k$. The system considers a good match (true positive) if $D_v$ and $D_l$ align (match).

#### 4.2.2.2 Motion-Based Objects Verification

Taking the appearance-based object discovery results as a hint, the robot verifies each candidate object by motion. We test the system with a poking (the gray box, the white paper, the tape dispenser, the yellow button, the green Android and the blue box) and a grasping (the yellow cartoon figure) experiment. Fig. 4.6 shows an example of the final verified objects in the poking experiment. (a grasping example is available at Fig. 4.8).

Due to the limitation of the appearance-based object discovery approach, negative results (e.g. the white paper on the desk) will be produced as it considers textures (color) to be candidate objects. However, such negative discovery results can be correctly recognized by the motion-based verification pipeline. The system will discard negative objects discovery result and only focus on the positive object discovery results. Notice that the motion-based objects verification pipeline

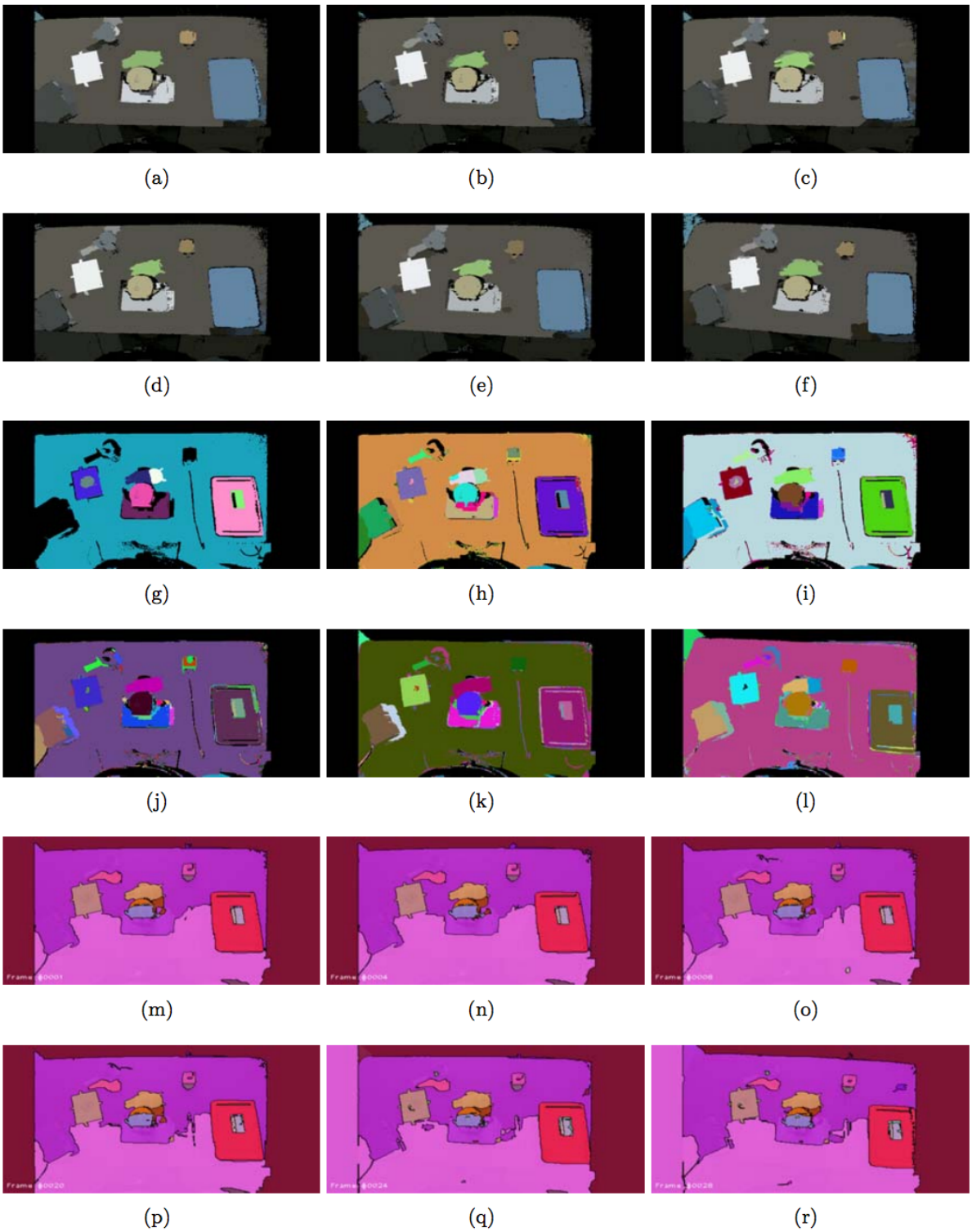


Figure 4.5: Appearance-based object discovery results by the proposed method (the first row), [49] (the second row) and [60] (the third row). The proposed method has the best performance in terms of precision and recall of object pixels (scored using motion-verified object boundaries).


Figure 4.6: Examples of the verified objects set (the five objects under the scene) via poking in the test scenario. The white paper from the appearance based objects discovery result (Fig. 4.5) is detected as invalid object and discarded. The yellow figure in the middle of the desk is verified and learned by grasping (Fig. 4.8).

does not discover the blue box completely as its initial position (Fig. 4.4) and final position overlaps (with the same geometry and appearance). However, the system is able recover it via the tracking and reconstruction approaches and learn its full 3D model (Fig. 4.7).

#### 4.2.2.3 Learning

Once an object is verified, the system learns the 2D and 3D model of the object via the tracking and reconstruction pipeline. As Fig. 4.7 shows, the system is capable to learn all candidate objects verified by the poking experiment.

To obtain a complete model of an object, the observation of multi-views of the object is required. We demonstrate this by a grasping experiment, where the robot grasps an object (the yellow cartoon figure) on the desk and moves it under the observation of the camera once the object is discovered by the proposed discovery pipeline. A complete 3D model of the object is reconstructed when the system has a complete knowledge of the object (takes approximate 500 frames). Fig. 4.8 shows an example of the final learning result of the object via grasping.

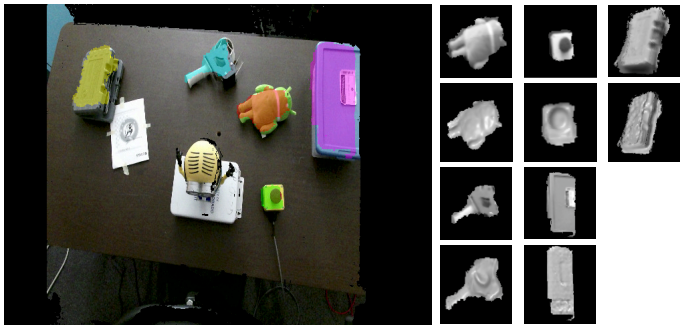

Figure 4.7: Final learned objects of the poking experiment. The left figure shows the final tracking results (shown as different color transparent masks). The right figure shows the virtual gray and the phong shading images of the final reconstruction results.

#### 4.2.2.4 Time Analysis

The system is tested using a single NVIDIA TITAN GPU, Intel i7 CPU desktop, with $640 \times 480$ resolution of input images. Table 1 shows our system run-time in different stages. Notice that the robotic manipulation, motion-based object verification, tracking and reconstruction approaches are processed simultaneously. The proposed system has a low running time complexity (22 Hz/object) during the learning (tracking and reconstruction) stages (GPU-based). The appearance-based object discovery approach is relatively slow, which takes approximately 10 min to discover objects using 10 frames (CPU-based). However, a similar CPU-based system [59] has been implemented which runs at 1 Hz. Based on the time analysis above, it is reasonable to conjecture that our system can run in real-time after optimization.

### 4.2.3 Failure Cases and Future Work

Although the system is robust to many real-world operating conditions, there are several limitations of our current work. The appearance-based object discovery method depends on the difference between the foreground and the background and cannot discover objects under extremely dark/bright environments. For robot manipulation and motion-based object verification approaches, due to the limitation of the robot platform we have, the system cannot grasp or poke

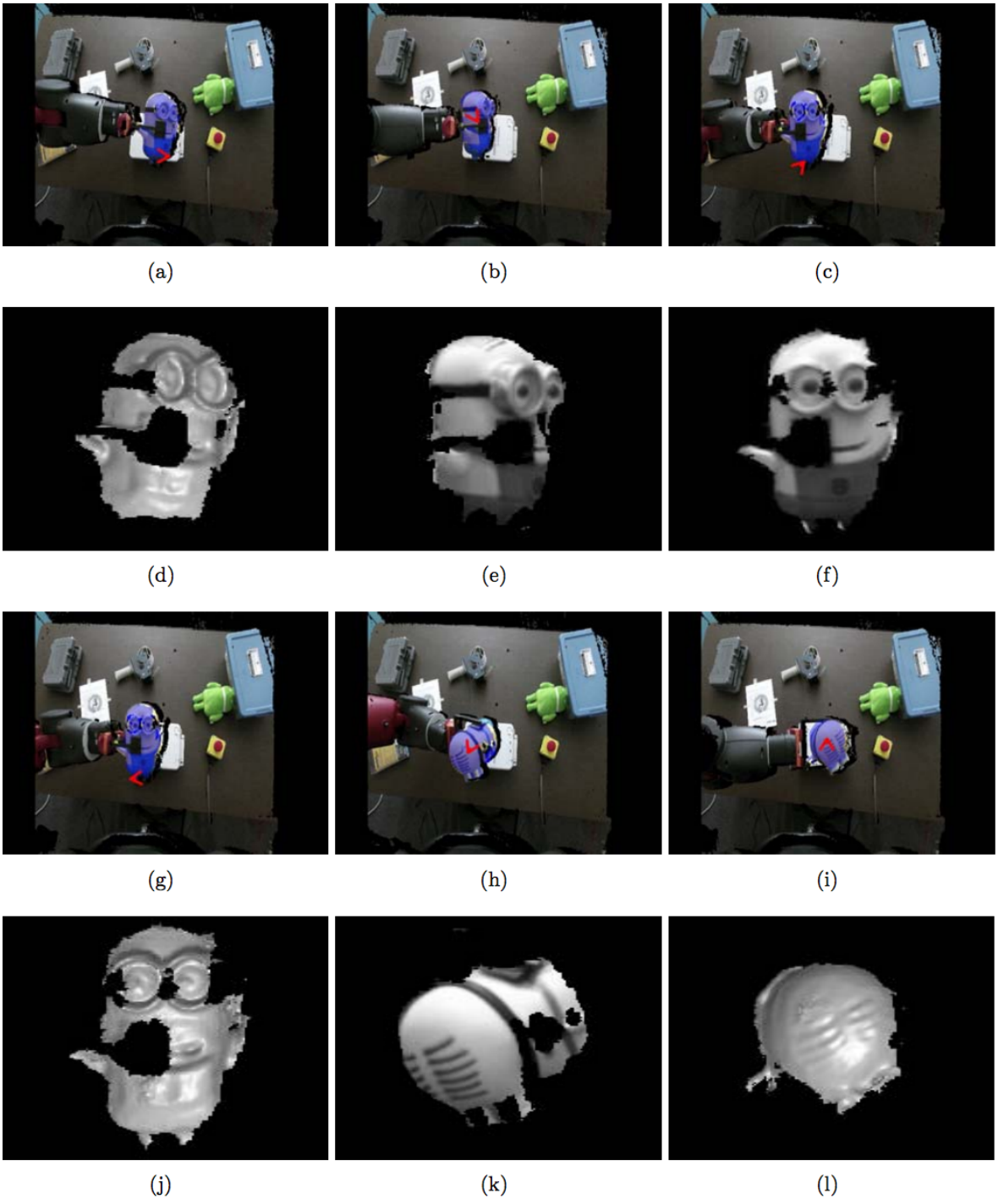


Figure 4.8: Tracking (the first row, shown by blue transparent masks) and reconstruction (the second row, shown by the virtual image of the object's 3D SDF) results of a verified object (the yellow cartoon figure). In this scenario, the robot grasps an object on the desk and moves it around the scene (for 500 frames). A complete 3D model of the object is learned after the robot has enough observation of the object. The red arrow shows the object's estimated velocities.

Table 4.1: System run-time

| | |
|---|---|
| Appearance-Based Object Discovery (10 frames) | 10 min |
| Manipulation Path Planning | 5 s |
| Robotic Manipulation | 1 min |
| Motion-Based Object Verification | 10 ms |
| Tracking & Reconstruction (one object) | 38 ms |

too small, too large, or too heavy objects. Additionally, the learning pipeline requires sufficient pixels for robust tracking, which means that sometimes the system will not learn the model for very small objects.

The success of our framework opens many areas of future work. The appearance-based object discovery pipeline can be improved by integrating object shape information. The system could use a single RGB camera by using techniques from [115, 113]. The path planning for the grasping/learning stage can be improved by a feedback system between the vision and the manipulation pipeline. The learned 3D model of objects could be shared among different robots for similar object detection by 3D tracking techniques [124].

### 4.2.4 Summary

We present a novel unsupervised framework for a robot to discover and learn unknown objects in the scene by manipulation. The system performs dense 3D simultaneous localization and mapping concurrently with unsupervised object discovery. Spatio-temporal cues and appearance cues are used to produce a set of candidate objects. A motion-based verification strategy is used to verify candidate objects, and object appearance and shape is subsequently learned by grasping and poking. We compare three different approaches for appearance-based object discovery and find that a novel form of spatio-temporal super-pixels gives the highest quality candidate object models in terms of precision and recall.

# Chapter 5

## Generating Auxiliary Groundtruth Via Label Propagation in Sequences

In this chapter, we propose an unsupervised method for transferring pixel-wise dense labels from finely annotated frames of a video sequences to the rest of unlabeled frames. We are motivated by the fact that recent progress in computer vision has been driven by high-capacity models which are trained on large datasets. However to our knowledge, there is no public dataset that has groundtruth annotation for all the frames in a video sequence. Unfortunately, creating large datasets with pixel-level labels has been extremely costly and expensive due to the amount of human effort required. In this work we show that the auxiliary groundtruth obtained from label propagation can be used to train and improve the performance of deep learning methods, more specifically for the task of temporal semantic segmentation.

Here, the idea is to leverage the temporal coherence between adjacent frames in videos for label propagation. Given a limited set of sparse hand labeled frames, our method propagates labels through the rest of unlabeled frames in the video sequence based on color and motion. The end result is a conservative labeling of the video, where large parts of the scene are labeled into the known predefined classes, and a void label is assigned to all the pixels that we are not sure about their class due to issues like occlusion or moving objects. Our algorithm provides an efficient tool to reduce the cost of mundane manual labeling, therefore producing abundant amounts of usable groundtruth data. A novel application of this algorithm is for semantic segmentation and scene parsing especially in temporal semantic segmentation where having groundtruth for each individual frame in the sequence is beneficial. Previous video label propagation algorithms largely

avoided the use of dense optical flow estimation due to their computational costs and inaccuracies, and relied heavily on slow and complex appearance models. Here, we introduce a method based on our spatio-temporal segmentation framework for dense pixel label propagation. We also introduce a simple web-based video annotation tool which is based on the techniques introduced in previous chapters. Using this tool, we are able to acquire a rapid propagation of semantic labels within and across images of a video sequence.

We bring out the advantages and drawbacks of our approach. We demonstrate the efficacy of our proposed approach using extensive qualitative and quantitative experiments. Also we compare our results with pure motion based propagation methods. It also enables us to take a step further and study the impact of incorporating temporal information for semantic segmentation. Our main contribution is to demonstrate the use of noisy generated groundtruth along with groundtruth to improve the performance of deep learning models. We provide recommendations on how to train these models properly when dealing with large corpus of noisy groundtruth. We validate the presented approach by producing dense pixel-level semantic annotations for about 18K images extracted from CamVid dataset [16] where there is only 701 groundtruth frames are available. We also demonstrate qualitative results for Cityscapes dataset [33] for which we generated about 105K auxiliary groundtruth.

## 5.1 Introduction

Recently deep learning has driven tremendous success in computer vision tasks such as image classification [140], object detection [129, 127] and semantic segmentation [100, 117, 7]. More precisely, convolutional neural network (CNN) based models are achieving outstanding performance by training deep models on datasets with millions of labeled images such as [85] for classification problems. However, creating large datasets with pixel-wise semantic labels is known to be very challenging due to the amount of human effort required to follow object boundaries accurately. High-quality semantic labeling was reported to require 60 minutes per image for the CamVid dataset [16] and 90 minutes per image for the Cityscapes dataset [33]. Since producing

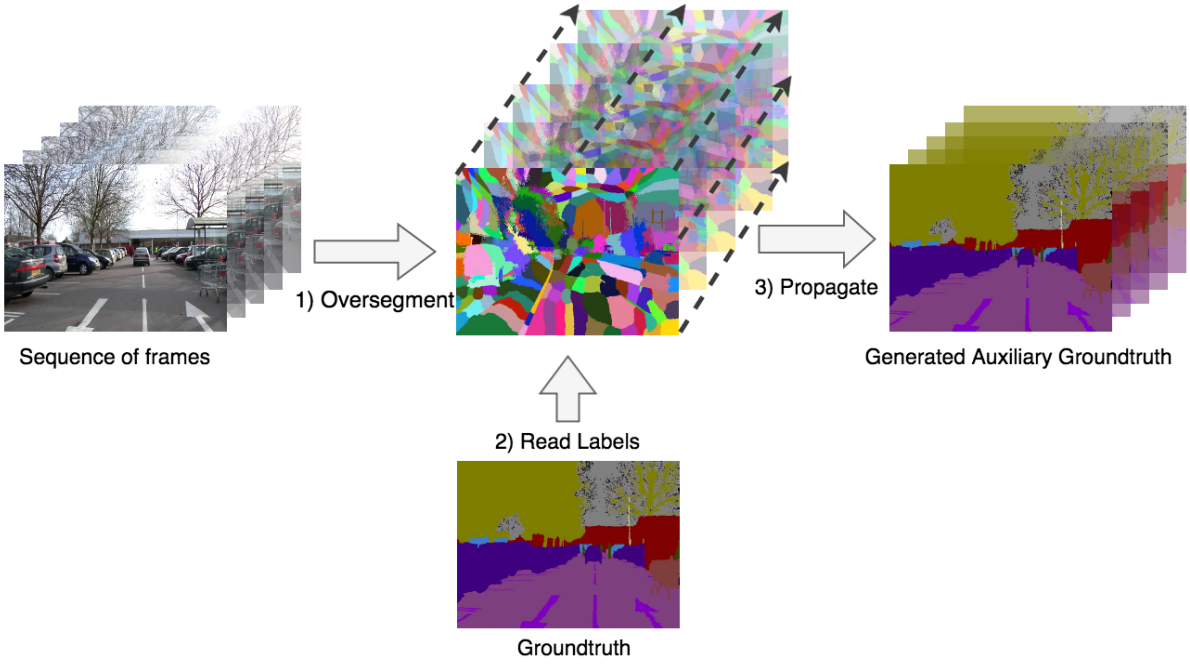


Figure 5.1: Illustration of our approach for generating auxiliary groundtruth.

precise pixel-wise annotations is labor-intensive, detailed and comprehensive semantic segmentation datasets are orders of magnitude smaller than image classification datasets. This problem is also referred to as the "curse of dataset annotation"[165]. One option to overcome this problem is to initially train semantic segmentation models for image classification, for which large annotated datasets are available, and then fine-tune them on semantic segmentation datasets, which have smaller dataset. The other option is to produce simulated dataset for which it is much faster to obtain the semantic labels, however creating photo-realistic and diverse imagery is known to be a hard problem on its own. We are therefore interested in creating large datasets with pixel-accurate semantic labels. This will eliminate the need for pre-training semantic segmentation models on image classification datasets. They may also significantly improve the accuracy of semantic segmentation models, which appear to be constrained by the amount of available data rather than their capacity to learn. Also having access to the annotation of consecutive frames might be beneficial for the tasks that study the temporality of data such as temporal semantic segmentation, action

recognition or video analysis.

In this work, we explore the use of our spatio-temporal segmentation framework for creating large-scale pixel-accurate groundtruth data that is suitable for training deep learning models. We use the CamVid dataset [16] as an example. The CamVid dataset has high quality annotation ($720 \times 960$) of street scenes. The annotated frames are picked at 1 $fps$ from a set of videos recorded at 30 $fps$. Our goal is to transfer the semantic labels from these annotated frames to all the intermediate unlabeled frames (See Figure. 5.1 for an illustration). Using our spatio-temporal based approach, we have generated pixel-level groundtruth for about $18K$ frames extracted from CamVid dataset where there is only 701 annotated frames available. The labeling process was completed in only less than 3 days which compared to the numbers reported by the dataset (60 minutes per frame) is orders of magnitude faster and more efficient. Needless to say, the generated groundtruth is inherently noisy compared to the original groundtruth which is an artifact of automatic label propagation. Typically street scenes (i.e CamVid dataset) have a lot of occlusion and motion in them which makes it very challenging to infer the pixel values of the middle frames from only the two labeled end of the sequence.

The experiments in this work perform a comparative study of naive methods for label propagation using optic flow based methods and demonstrates both quantitatively and qualitatively that the proposed label propagation method are superior to naive solutions. Also we show that training existing deep learning models with additional noisy data outperforms the baseline. We also provide recommendations on how to train such models with noisy data effectively. Our baseline is obtained by training the *Fully Convolutional Network* (FCN) only on the groundtruth which results in 87.9% pixel accuracy. We show that auxiliary groundtruth in addition to the original groundtruth helps FCN to increase the pixel accuracy by 1.5%.

The main contributions of this work are: i) We propose a simple and efficient unsupervised method to propagate semantic labels in the video sequences. ii) We show that generated auxiliary groundtruth to supplement the CamVid dataset increases accuracy on the respective dataset and can be used for pre-training semantic segmentation models. iii) We present a web-based video

annotation tool for semantic segmentation that reduces the burden of hand-labeling. iv) We analyze the usefulness of auxiliary training data for deep learning models.

The remainder of this paper is organized as follows. Section 6.2 discusses the literature related to our problem. Our proposed method is presented in Section 6.4, which is followed in Section 5.4 by a demonstrating our results. In section 5.5, we introduce our web-based video annotation tool. Section 6.6 provides some final conclusions and directions for future work.

## 5.2 Related Work

In the recent years, many research has been devoted to label propagation both in the spatial and temporal domain. One of the pioneers [97] introduces an "image matching"method to transfer label from training images to similar test images. In particular, for a given query image, their system first retrieves its nearest neighbors from a large database of annotated images. Then, dense correspondences between the input and the nearest neighbors are computed using the dense SIFT flow algorithm [98]. Finally, based on these correspondences, the system warps the existing annotation and uses Markov random field to segment and categorize the query image. They argue that their method is similar to optic flow correspondences for non-sequential data. Similar to the aforementioned work, [173] proposes a supervised label transfer method for the semantic segmentation of street scenes. They establish dense correspondence between the query image and found images with a KNN-MRF matching scheme. It is followed by a matching correspondences classification to reduce the number of semantically incorrect correspondences. Also [27] proposes a semantic label transfer method using supervised geodesic propagation (SGP). They learn a joint boost model on the similar image set of the query image.

Another well known method is that of [177]. They design a method for assigning soft labels to nodes of a fully connected graph with few labeled nodes. Their system assigns weights to the graph edges based on the proximity of the corresponding nodes in the graph and an iterative update is used to propagate labels to the unlabeled nodes. This method is shown to perform well in assigning labels to a handwritten digits database. Furthermore, [18] proposes a directed graphical model

for label propagation in video sequences. A variational EM based inference strategy propagates the labels either one of several class labels or the void label for unknown. Then they use these labels to train a multi-class classifier. The pixel labels estimated by this classifier are injected back into the Bayesian network for another iteration of label inference. In [78], label propagation has been considered as the problem of learning the correlation between multiple class labels. Although contextual information is meaningful, the main disadvantage of such techniques is their reliance on large amount of training data, which is difficult to obtain.

In [24] proposes a probabilistic framework that estimates the reliability of the sources and automatically adjusts the weights between them[6] proposes a probabilistic graphical model that jointly models appearance and semantic labels using a coupled-HMM model in video sequences. However, their model does not have a mechanism for occlusion handling. In our method, after propagating the labels from one end to the other, we perform a reverse propagation to fill the pixels that are being dis-occluded in the reverse setting. Later on, [19] extended this model to include correlations between non-successive frames by using a tree structured graphical model. In another work, [5] introduces a mixture of trees probabilistic graphical model for semi-supervised video segmentation. They provide a variational inference scheme for this model to estimate super-pixel labels, their corresponding confidences, as well as the confidences in the temporal linkages.In contrast to these supervised approaches, we propose an unsupervised method that does not need to be trained for label propagation. More recently, [131] proposes a novel approach for creating pixel-accurate annotations for images extracted from modern computer games. They show that the associations between image patches can be reconstructed from the communication between the game and the graphics hardware. Using their technique, it is possible to rapidly propagate semantic labels within and across images synthesized by a game.

Our work is also closely related to the research that has been done on learning deep learning models from noisy data. In [121], the authors study the problem of semantic image segmentation from either weakly annotated training data such as bounding boxes or image-level labels or a combination of few strongly labeled and many weakly labeled images. They develop

Expectation-Maximization (EM) methods for semantic image segmentation model training under these weakly supervised and semi-supervised settings. Their experiments on PASCAL VOC 2012 dataset have shown that relying on only weak annotation at the image-level is insufficient to train a high-quality segmentation models. However, good performance is achievable when combining a small number of pixel-level annotated images with a large number of weakly annotated images in a semi-supervised setting. [39] proposes a method that achieves competitive accuracy that requires bounding box annotations. The idea is to iterate between automatically generating region proposals and training convolutional networks. These two steps gradually recover segmentation masks for improving the networks. Moreover, [122] proposes an approach to learn a dense pixel-wise labeling from image-level tags. Each image-level tag imposes constraints on the output labeling of a Convolutional Neural Network (CNN) classifier. Also [164] addresses the problem of training a CNN classifier where millions of easily obtained noisy labels is available with only a limited number of clean label.with a massive amount of noisy labeled training data and a small amount of clean annotations for the image classification task. They model the relationships between images, class labels and label noises with a probabilistic graphical model and further integrate it into an end-to-end deep learning system.

## 5.3 Approach

Given a groundtruth labeling $L^t$ for a frame $I^t$, our goal is to propagate the semantic labels of frame $I^t$ to the rest of frames in the sequence and obtain their "noisy"groundtruth. Figure. 5.1 gives an overview of our label propagation. In this section, we first show a naive flow-based propagation model and then we discuss the details of our spatio-temporal appearance based model.

### 5.3.1 Motion Based Model

As mentioned in [24], using *only* motion information there are two possible ways to infer the label of a pixel in frame $I^{t+1}$ from frame $I^t$ : 1) forward flow from $I^t$ to $I^{t+1}$ and 2) backward flow from $I^{t+1}$ to $I^t$. Here, we use the SIFT flow [98] due to its efficiency and relative effectiveness.

The task of propagating labels with forward flows alone is to determine the proper label $L()$ for each pixel $n$ in $I^{t+1}$ by using:

$$L(I_n^{t+1}) := L(I_{n\prime}^t) \quad \text{where} \quad f_{fwd}(n\prime) = n \tag{5.1}$$

In simple words, we read the label of pixel $n$ from pixel $n\prime$ where $n\prime$ is connected to $n$ along its forward motion vector. For the task of propagating labels with the backward flows,

$$L(I_n^{t+1}) := L(I_{n\prime}^t) \quad \text{where} \quad f_{bwd}(n) = n\prime \tag{5.2}$$

Here, the read the label of pixel $n$ from pixel $n\prime$ where $n$ is connected to $n\prime$ along its backward motion vector.

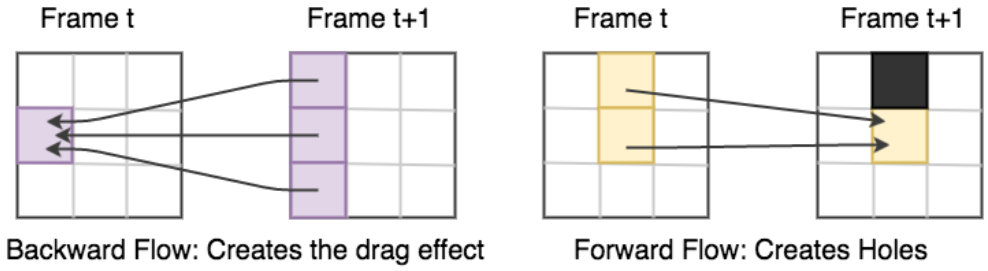


Figure 5.2: The holes in forward flows and the dragging effect in reverse flows.

At first glance, this solution seems very easy and straight forward. However, a simple experiment on the Camvid dataset shows otherwise. This is due to the fact that the forward and inverse flow functions $f_{fwd}(.)$, $f_{bwd}(.)$ are neither injective nor surjective. As demonstrated in the Fig. 5.2 inspired by [24], this method creates holes in $I^{t+1}$ when taking labels from $I^t$ along the forward motion vector. In particular, when occlusion happens it is likely that more than one pixel in $I^t$ point to the same position in $I^{t+1}$ which makes holes. Moreover, the label value of the pixels in $I^{t+1}$ that reappear again after occlusion can not be inferred easily from $I^t$. With backward-flow-based propagation, the dragging effect occur because a reappearing (previously occluded) pixel $I^{t+1}$ is forced to take some uncorrelated label from $I^t$ label by the backward flow, while in theory it has no corresponding label. It is also shown in [6] that deterministic optic flow based label propagation

is inferior to probabilistic patch based methods. In addition, naive motion based methods will be further challenged over lengthy sequences. Figure. 5.3 shows an example of label propagation with this approach for both forward motion and backward motion on Cityscapes dataset.

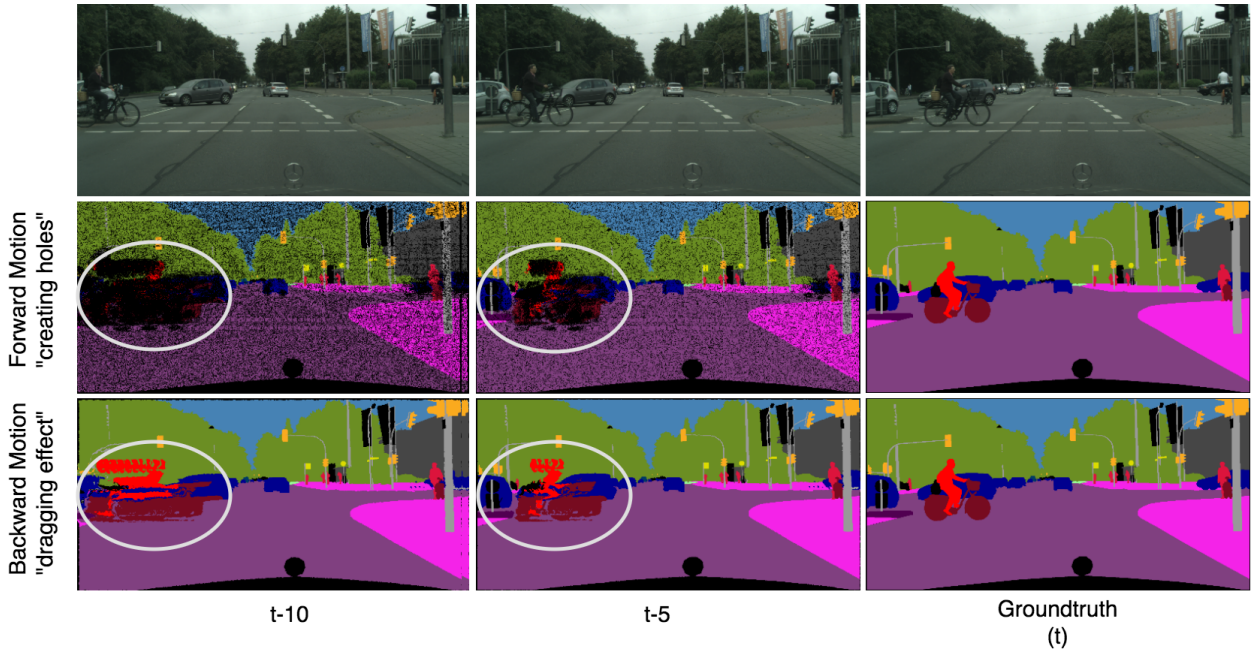


Figure 5.3: Sample of label propagation via motion only

### 5.3.2 Appearance and Motion Based Model

Our proposed approach starts with over-segmenting the video sequence to to very fine 3D superpixels. To do so, we apply the techniques introduced in previous chapters. Then for each 3D superpixel in the groundtruth frame $I^t$, we take the label that is covering the majority of the pixels and transfer the value to the rest of pixels inside of the 3D superpixel which extends to the consecutive frames. We found that best result is achieved when the video is process incrementally in a sliding window fashion instead of processing the entire video at once. More precisely, staring with the first groundtruth frame $I^0$, we consider a window of size $n$ frames ($n = 5$ in our experiments), and by this method we propagate the labels from $I^0$ to the next $n-1$ frames and obtain their noisy annotations. We process the rest of the video as if the last frame with the noisy annotation, $I^t$, is

the next reference groundtruth. In the sequences where there are rapidly moving objects with large regions of occlusion, it is recommended to process the video incrementally and even in smaller batches. However, if there is not a lot of motion in the scene, it is possible to use larger batches of frames to perform over-segmentation.

**Occlusion Handling.** Typically, the groundtruth label is given for both the start and the end of the sequence, for example in CamVid dataset[16]. This can be leveraged to tackle the occlusion/dis-occlusion problem. Our method processes each sequence twice: First, in the original order of frames where we infer the labels and propagate them with only using the information from $L^{start}$ (the start groundtruth). Second, in the reverse order where we perform the same process but only use the information from $L^{end}$. Therefore, we receive two potential labels for each pixel $I_n^t$ in the sequence suggested by $L^{start}$ and $L^{end}$. These labels are then given a score based on pixel $I_n^t$'s distance from the two ends of the sequence which is also normalized by the length of sequence. Finally, for each pixel, we choose the label with the maximum score. This way, we are capable of labeling the reappearing pixels that were previously occluded in the scene. We assign the unknown label to the pixels that wither take unknown label through propagation automatically or receive two different labels with the same probability.

One drawback of our approach is that the performance degrades over time in the datasets with only one groundtruth per sequence such as Cityscapes dataset. This is due to the fact that we are only able to transfer the labels for the objects that are present in the groundtruth frame and gradually when the objects move out of the scene, we will not be able to infer the labels anymore. The comparison between the naive label propagation method based on motion only and our approach is shown in Figure 5.5 and Figure 5.4 for CamVid and Cityscapes datasets respectively.

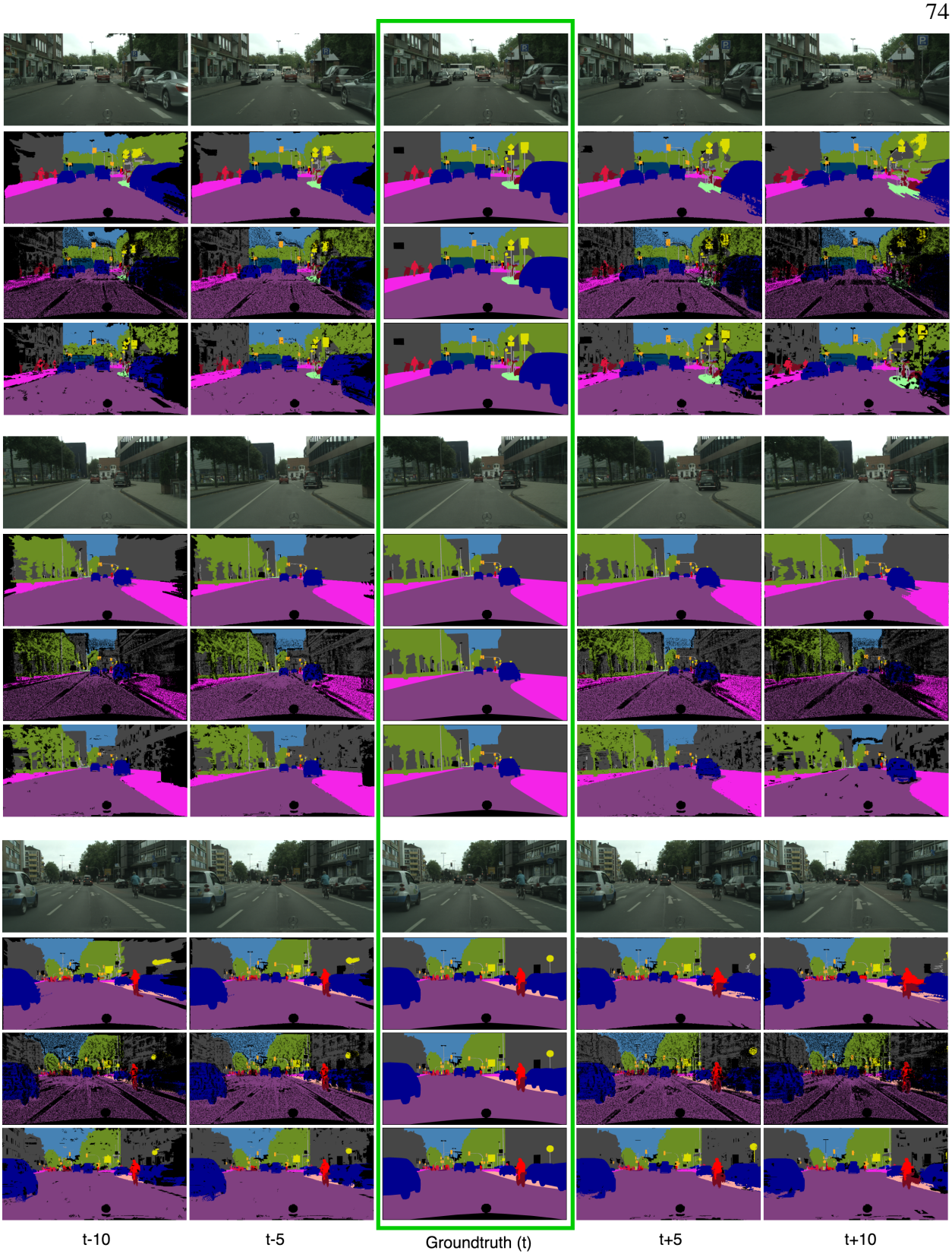


Figure 5.4: Comparison of our method and motion only label propagation on Cityscapes dataset

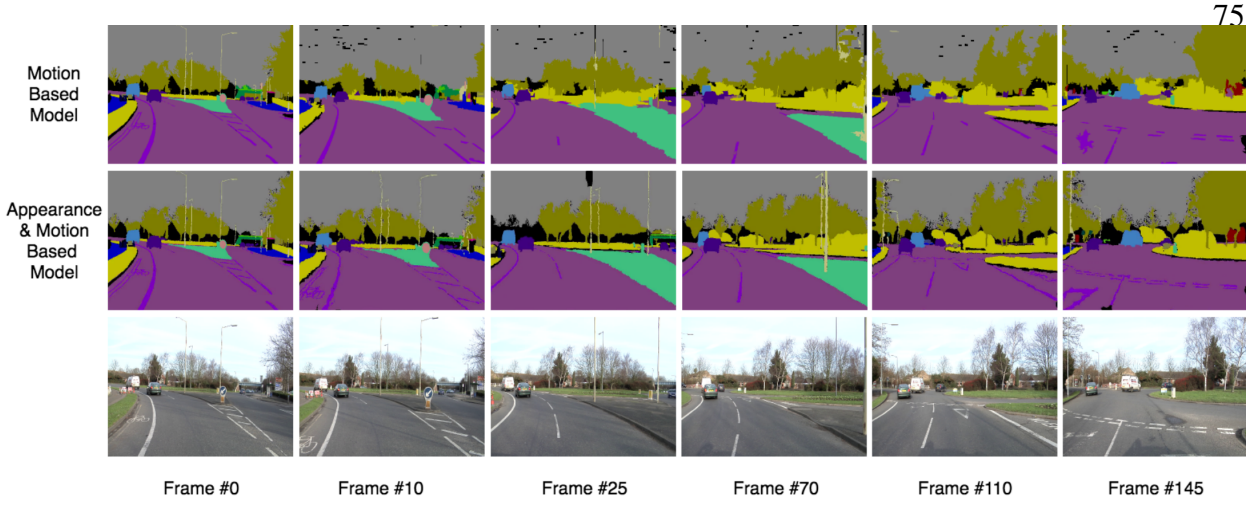


Figure 5.5: Groundtruth frames are provided at 30 hrtz frequency starting from frame 0. Propagated groundtruth for frames 10, 25, 70, 110 and 145 are shown for comparison between motion based model and our model. Note that our method captures the details much better and preserves a more accurate boundary compared to the motion only label propagation method.

### 5.3.3 The Reliability of Generated Auxiliary Groundtruth

The focus of this research is to reduce the burden of hand-labeling and create large-scale auxiliary groundtruth. However, we need to ensure that the the quality of generated groundtruth is appropriate for training deep learning models. We are therefore interested in establishing strategies to use the produced data in the most effective way for semantic segmentation. As demonstrated in Figure. 5.6, the groundtruth labeling for the intermediate frames can be erroneous around the object boundaries but for the majority of the scene, the quality is acceptable. Also adding the generated groundtruth frames that are farther away from the original groundtruth provides more diversity in the training set. This will help us to train more robust models that are better in generalizing data and enhance the overall performance. Due to the inherent noise caused by label propagation, the generated groundtruth can not be completely trusted when it is being trained on. We address this by assigning a confidence weight to each generated groundtruth sample which is an indication of how much we can trust the labels. We use these values to compute a weighted

cross-entropy loss. The prediction scores made by the semantic segmentation model are being passed through a Softmax function that maps these scores to a probability distribution over $K$ classes. Softmax classifier is defined as:

$$\hat{p}_{nk} = \frac{e^{x_{nk}}}{\sum_{k'=1}^{K} e^{x_{nk'}}} \quad \text{for k = 1, . . . , K}$$

And the cross-entropy loss has the form:

$$L = \frac{-1}{N} \sum_{n=1}^{N} \log(\hat{p}_{n,l_n}) \tag{5.3}$$

Where $n$ is the sample and $l_n$ is its corresponding label. We update the cross-entropy loss to:

$$L = \frac{-1}{N} \sum_{n=1}^{N} w_n \log(\hat{p}_{n,l_n}) \tag{5.4}$$

Here, $w_n$ shows the importance of sample $n$. Effectively, We want the model to treat each sample differently based on the quality of their annotation during training. It is important to ensure that errors in predicting the clean samples is "amplified"when computing the loss. This can be done by providing a larger weight for the clean samples to encourage the model not to miss these examples. All the groundtruth samples have a weight $w > 1$ that is chosen based on the ratio of noisy and clean groundtruth. For example, if the size of noisy groundtruth dataset is 5 times bigger than the size of clean groundtruth we set $w = 5$ for clean samples.

## 5.4 Evaluation

In this section, we evaluate the presented label propagation technique using extensive experiments, followed by a discussion of the results.

### 5.4.1 Training Datasets and Optimization

We use the standard implementation of FCN-8s [100] in all the experiments. We set the base learning rate to $10^{-5}$ and used Stochastic Gradient Descent for optimization. The momentum

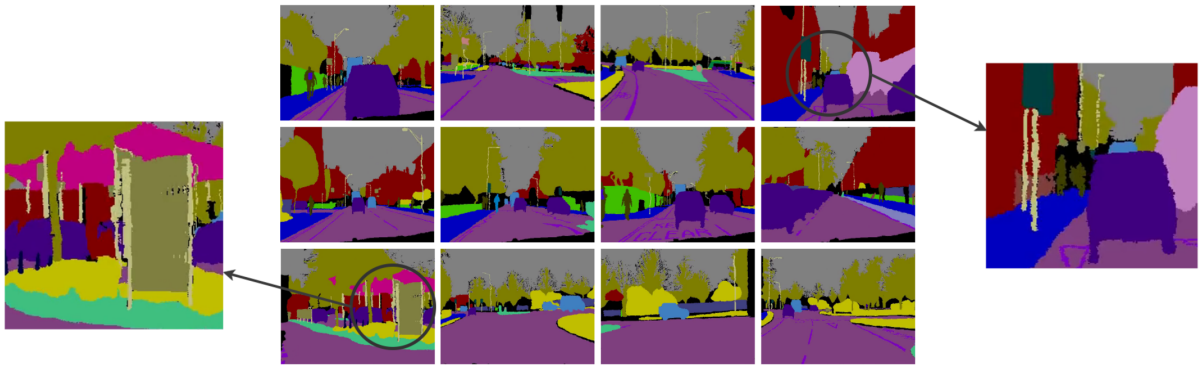

Figure 5.6: Quality of some pseudo groundtruth labellings with our method. All of these examples chosen to be the 10th frame after the available groundtruth for the CamVid dataset. Overall, the quality of label propagation is good but the zoomed-in version illustrates the noisy labellings

and weight decay are set to 0.9 and 0.0001 respectively. Additionally, we initialize the weights using pre-trained VGG-16 [140] on ILSVRC dataset. The training data is first resized to half the resolution and then augmented with random crops and mirroring to resist over-fitting. We train the network for 80K iterations on CamVid dataset.

We perform our experiments on two prominent datasets of road scene understanding: the CamVid [16] and Cityscapes [33] datasets. Both datasets provide video input along with pixel-level semantic annotations of selected frames. Regardless of the size of the groundtruth dataset, the main difference between the two is the frequency in which the frames are chosen for annotation. CamVid is a dataset for urban scene understanding. We used the split and image resolution from [7], which has been adopted in a number of prior works. The split consists of 367 frames for training, 101 frames for validation and 233 frames for test. The frame resolution is $720 \times 960$ and each pixel belongs to any of defined 11 semantic classes namely Building, Tree, Sky, Car, Sign, Road, Pedestrian, Fence, Pole, Sidewalk, Bicycle. Over 10 min of high quality 30 Hz video footage is available for this dataset, with corresponding semantically labeled images at 1 Hz and in part, 15 Hz. Similarly, Cityscapes is a benchmark suite and large-scale dataset for semantic scene understanding. It contains 5,000 high quality ($1024 \times 2048$) pixel-level finely annotated images collected from 50 cities in different seasons. The images are divided into sets with numbers 2,975,

500, and 1,525 for training, validation and testing. In addition, 20,000 additional images with coarse annotation is available to enable methods that leverage large volumes of weakly-labeled data. Pixels belong to 19 semantic classes, including road, car, pedestrian, bicycle, etc. These classes further belong to 7 categories, i.e., flat, nature, object, sky, construction, human, and vehicle. Each annotated image is the 20th frame from a 30 frame video snippets, and proceeding and trailing video frames for each annotated image are available upon request. Using our efficient and simple label propagation method, we produced about 18K auxiliary groundtruth from CamVid dataset and about 105K labeling from Cityscapes dataset. The qualitative results is shown in Figure 5.5 and 5.4 for comparison against naïve motion based approaches.

### 5.4.2 Experiments

Table 5.1: Comparison of training FCN on the auxiliary groundtruth by the naïve flow-based model and our proposed model

| Method | Mean IOU | Overall Pixel Accuracy | Dataset Size |
|---|---|---|---|
| Only GT | 54.6% | 75.8% | 367 |
| GT + Flow based AGT | 55.1% | 76.3% | 5505 |
| GT + Our AGT | 56.8% | 78.2% | 5505 |

In the first experiment, we performed a preliminary experiment to compare our approach with the most naïve flow-based label propagation model. The results have been reported on the CamVid dataset. For this experiment we created a training set with all the clean groundtruth and incrementally added the noisy groundtruth in the order of frames for the first 10 frames. The cross-entropy loss is computed with a fixed weight ($w = 1$) for all the samples. Table 5.1 gives a detail comparison of the mean IOU and pixel accuracies for both methods. We observed the the accuracy

sharply decreases as we move away from groundtruth frame in both methods, however the change is more dramatic for the flow-based approach compared to ours. These results show that simple flow-based models are more erroneous, especially in the presence of rapid motion in the scene, and therefore are not suitable for label propagation on their own.

Table 5.2: Comparison of training FCN for different weights in computing the cross-entropy loss

| Method | Mean IOU | Overall Pixel Accuracy | Dataset Size |
|---|---|---|---|
| Only GT | 54.6% | 75.8% | 367 |
| GT + weighted AGT ($w = 1$ for all) | 55.7% | 76.2% | 1835 |
| GT + weighted AGT ($w = 4$ for GT) | 57.1% | 79.1% | 1835 |

In the second experiment, we investigate the impact of weighted cross-entropy loss in training. Our baseline is the FCN network trained on *only* the clean groundtruth. We compare the baseline with FCN trained on a noisy dataset that is consisted of the clean groundtruth and the next 4 noisy groundtruth after it. We train FCN on the noisy dataset in two different configuration: 1) with $w = 1$ for all the samples (default weight), 2) with $w = 4$ for all the clean samples. The results is shown in Table 5.2. It is evident that when a larger weight is chosen for the clean samples, the performance improves over time. This can be due to the fact that the size of noisy groundtruth is multiple times larger than the clean groundtruth, so it is necessary to encourage the model to pay more attention to the prediction errors on the clean samples.

Table 5.3: Comparison of training FCN for various experiments, frames are being added incrementally (1st: all the first frames after the groundtruth, 2nd: all the second frames after the groundtruth and so on)

| Method | Mean IOU | Overall Pixel Accuracy | Dataset Size |
|---|---|---|---|
| Only GT | 54.6% | 75.8% | 367 |
| GT + 1st | 55.3% | 76.0% | 734 |
| GT + (1st,2nd) | 56.4% | 76.8% | 1101 |
| GT + (1st,2nd,3rd) | 56.8% | 76.7% | 1468 |
| GT + (1st,2nd,3rd,4th) | 57.1% | 79.1% | 1835 |
| GT + (1st,2nd,3rd,4th,5th) | 57.0% | 78.7% | 2202 |
| GT + first 15 frames | 56.8% | 78.0% | 5505 |

In the third experiment, we incrementally add the noisy samples to the groundtruth. We want to investigate whether it is better to have a smaller training set that has better quality or to have a larger, much noisier dataset that is also more diverse. Table 5.3 shows that the performance improves slightly as model sees more samples that are farther apart from the groundtruth but it is not a significant improvement dues to the increasing noise.

Table 5.4: Comparison of training FCN for various experiments; frames are chosen one set at a time (1st: all the first frames after the groundtruth, 2nd: all the second frames after the groundtruth and so on)

| Method | Mean IOU | Average Pixel Accuracy | Dataset Size |
|---|---|---|---|
| Only GT | 54.6% | 75.8% | 367 |
| GT + 1st | 55.3% | 76.0% | 734 |
| GT + 2nd | 55.8% | 76.8% | 734 |
| GT + 3rd | 55.9% | 77.2% | 734 |
| GT + 4th | 56.5% | 77.9% | 734 |
| GT + 5th | 56.1% | 77.4% | 734 |
| GT + 6th | 55.6% | 76.7% | 734 |

The results in the previous experiment motivated us to analyze the diversity alone without changing the size of groundtruth. In this experiment, we train the FCN on the groundtruth and only one set of noisy groundtruth at a time. Interestingly we observe the same trend in the performance as before. Table 5.4 shows that the accuracy increases in the beginning but degrades afterwards. We observe that diversity in the training set help with the performance but also having a descent quality is important.

Table 5.5: Comparison of training FCN with the same size auxiliary dataset; frames are chosen either in consecutive order or randomly

| Method | Mean IOU | Average Pixel Accuracy | Dataset Size |
|---|---|---|---|
| Only GT | 54.6% | 75.8% | 367 |
| GT + consecutive frames | 57.1% | 79.1% | 1835 |
| GT + random frames | 55.8% | 75.5% | 1835 |

We also experimented with choosing the noisy samples in random. In this experiment, we train the FCN with 1) a training set that consists of the clean groundtruth and the next 4 noisy groundtruth frames after it. 2) a training set that consists of the clean groundtruth and randomly chosen noisy groundtruth where the final size is the same as case 1. As it can be seen in the Table 5.5, the random selection of noisy groundtruth is inferior to the selection of consecutive frames which again reinforces the importance of having good quality data.

Table 5.6: Comparison of the best results on pixel accuracies per class

| Method | Mean IOU | building | tree | sky | car | sign symbol | pedestrian | road | fence | column-Pole | sidewalk | bicyclist |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GT + Auxiliary | 57.6 | 75.0 | 70.6 | 88.2 | 70.7 | 36.9 | 93.3 | 43.0 | 37.5 | 10.8 | 74.8 | 36.0 |
| Only GT | 54.2 | 73.4 | 69.8 | 87.9 | 66.8 | 34.0 | 92.6 | 38.2 | 33.1 | 8.0 | 74.0 | 43.0 |

**Overall performance** In this experiment, we compare the overall performance of FCN trained on the original groundtruth against the best results obtained on our auxiliary groundtruth. We provide the pixel accuracies per class for both models. The details of comparison is shown in

Table 5.6. From Table 1, it can be seen that the FCN trained with extra auxiliary outperforms the FCN trained only on GT in all the classes. In this configuration, we are using the groundtruth and the next 4 noisy groundtruth after it, with $w = 4$ for the clean samples. The mean IoU accuracy is 57.6% that is 3.4% higher than baseline FCN trained on only the groundtruth .

## 5.5 Online Video Annotation Tool


Figure 5.7: Video annotation tool snapshots

Our proposed algorithms for label propagation is very well suited for creating large amount of labeled video data. Inspired by the recent success of online image annotation applications such as LabelMe Mechanical Turk [171] by MIT, InteractLabeler by Cambridge University and Vatic [157] of University of California Irvine, we developed an online video annotation tool based on the proposed framework in this chapter. Our tool is fundamentally different from these previously developed ones. We provide a temporal dense labeling of the scene(i.e. not only bounding boxes).

and also there is no need to include advanced tracking or motion analysis during annotation which makes the process seemless, more user-friendly and real-tiime. Our goal is to speed up and automate the process of video annotation where there is only minimal human interaction is necessary.

Figure 5.8: Video annotation tool snapshots

Our tool works as follows: For a given sequence of interest, we first over-segment the entire video offline to produce the superpixels in different hierarchical levels. The results are then uploaded on our server in conjunction with the raw frames. The annotator is then asked to color the scene with predefined semantic categories. When the annotator starts coloring the scene in say first frame of the sequence, the tool automatically transfer the coloring to the rest of frames in the sequence. So when the annotator navigates through the frames, they only need to either correct the erroneous propagated labels or color the new regions that have not been previously colored. This significantly increases the annotation speed mainly for two reasons: 1) superpixels are used instead of pixels that cover larger spatio-temporally coherent regions 2) By only annotating one frame in the sequence, automatically the rest of frames are being annotated for the majority of the scene that is common between them. We can also utilize this tool by uploading pre-annotated frames (i.e obtain from our label propagation pipeline) and ask the annotator to *only* correct the bad annotations. Figure 5.5 demonstrates a snapshot of our video annotation tool, the outline of superpixels is shown in white. In Figure 5.5 a sample of user annotation is shown. On the left, user has annotated the first frame on the sequence. On the right, user has navigated to another frame

in the sequence (the 10th frame) and the labeled regions (previously annotated in the first frame) have already been propagated.

## 5.6 Summary

In this work, we have investigated the possibility of using auxiliary groundtruth to produce more training data for CNNs. Our key contribution is to introduce a novel and efficient unsupervised method to propagate labels from groundtruth to the unlabeled frames in the sequence. We also analyze in details how the auxiliary groundtruth can be used to enhance the performance of a CNN-based semantic segmentation framework. We also proposed a simple weighting system to account for noisy data while computing the cross-entropy loss. We showed that when the number of generated noisy groundtruth samples is multiple times larger that the clean groundtruth samples, it is important to add more diversity to groundtruth data. We also observed that it is necessary to have much less reliance on the noisy groundtruth samples and encourage the network to pay more attention to the clean data. Moreover, we found out that accumulating the noisy groundtruth data does not necessarily improves the performance of semantic segmentation by a considerable amount whereas including more diversity almost always helps. It is important to note that in cases such as temporal semantic segmentation, labeled data has to be presented for the sequential data to train the deep learning models. Additionally, there are many exciting avenues for future research. One direction is to come up with ways to improve the quality of groundtruth data generation. Another interesting area of research would be to design algorithms that can handle large noisy groundtruth where there is little clean groundtruth is available more effectively.

# Chapter 6

## Temporal and Predictive Semantic Scene Understanding

The ability to quickly and accurately understand pixel-level scene semantics is a key capability required for robotics applications such as autonomous driving. Until now, the temporal aspect of this problem has been largely overlooked due to the expensive computation and substantial amount of effort required for labeling temporal data. In this work, we present a deep, recurrent multi-scale network that is capable of leveraging information present in temporal data to improve the accuracy of semantic segmentation. We propose using temporal information in two ways. Firstly, we formulate an approach that integrates semantic segmentation into a recurrent architecture where the model is given access to the previous frames in a sequence. We show that our model outperforms single frame segmentation when given access to the temporal data even when groundtruth information is not provided for all the intermediate frames. Secondly, we show that our model can be easily extended to the related problem of prediction: estimating the expected semantics of the scene a small number of frames into the future. This is a crucial capability for many robotics applications, where planning needs knowledge of the probable near future as well as the present.

Our architecture for semantic temporal segmentation has three key components: 1) an encoder that extracts the spatial features of frames using CNNs; 2) a convolutional LSTM module that integrates temporal features over many frames while preserving the spatial correlation; and 3) a decoder that combines different scale feature maps and produces final pixel-wise predictions for either the input sequence or the predictive future sequence. Our spatio-temporal CNNs achieve

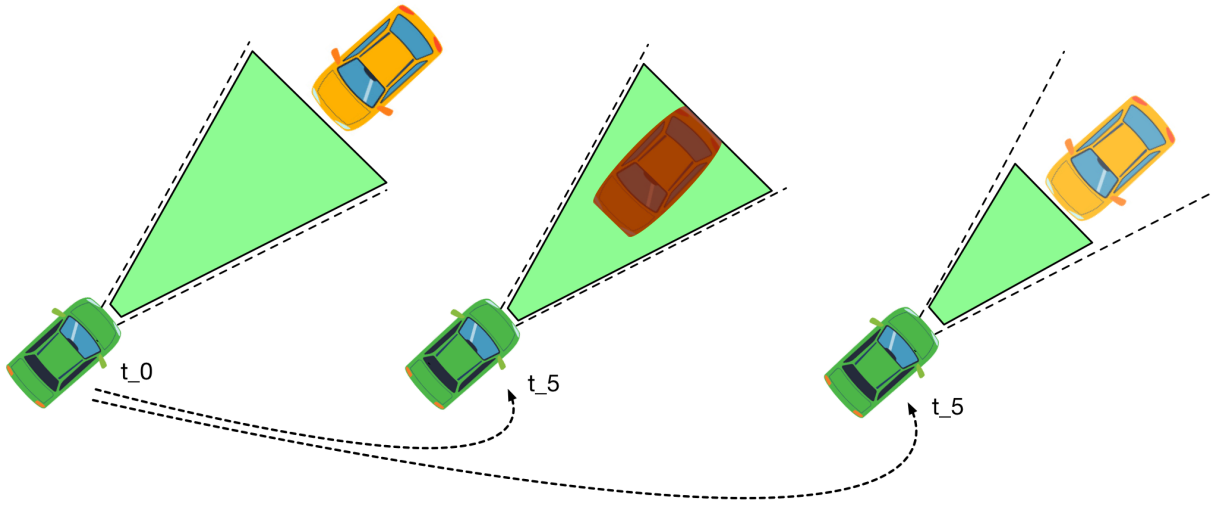


Figure 6.1: Free-space estimation for autonomous driving. The hero (in green) is directly approaching the vehicle (in yellow). (a) depicts the free-space inference at time $t = 0$. (b) if we do not consider the estimated state of free-space at $t = 5$ then our planner would consider trajectories that would be violated by the approaching vehicle. (c) only by considering *predicted free-space* we can correctly plan in this situation.

promising results as demonstrated for the CamVid and Cityscapes datasets without resorting to more complex random-field inference or instance detection driven architectures. Our proposed methodology can also be used for other visual tasks to benefit from temporal information with little modification. To our knowledge, our work is the first to exploit deep learning techniques for temporal and predictive semantic scene understanding.

## 6.1 Introduction

Pixel-level semantic labeling is becoming increasingly crucial for real-world robotics applications such as autonomous driving. The goal is to assign each pixel in the image a semantic label and provide complete understanding of the scene. These semantics can be leveraged for multiple tasks such as obstacle detection, odometry, and free-space estimation for trajectory planning. Standard approaches to semantic segmentation focus on improving the quality of single frame in-

ference, but do not consider how to leverage temporal consistency (which is a very strong signal frame-to-frame) over video sequences to improve accuracy. Another reason to use temporal information when estimating pixel-level semantics is for the important task of prediction: this is crucially important for robotics planning tasks, where one needs to know not only the state of the world at the current time-step but also the estimated state of the world in the near-future. For instance, for autonomous driving a crucial task is free-space estimation where the planner needs to ensure that any proposed trajectories (both nominal and emergency) lie within the envelope estimated from *predicted freespace*. This freespace task is depicted in figure 6.1.

One of the main obstacles that makes temporal semantic segmentation a challenging problem is the lack of finely annotated temporal data due to the substantial manual effort involved in producing pixel-accurate annotations. The existing efforts [86, 155] in video semantic segmentation mostly focus on the structure prediction from single frames and rely on random field optimization. This highlights the need for introducing methods that are capable of learning a meaningful representation of videos when groundtruth information is not available for each individual frame.

The typical architecture used in image semantic segmentation literature has two main components: (1) an encoder that downsamples the data and extract low-resolution coarse semantics, (2) a decoder that reconstructs the original input resolution from learned featured through either upsampling predictions or deconvolutions. Instead, we propose a simple and efficient model that uses RNNs for modeling the temporal correlation between learned coarse semantics. We use convolutional LSTMs to preserve the spatial dynamics while modeling the sequence temporally. Furthermore, our model utilizes the inherent hierarchical architecture of ConvNets in the encoder, forms a pyramid of feature maps that have strong semantics at different scales and strategically aggregates them to obtain the final prediction. We combine low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections.

Although conceptually simple, our method is able to combine temporal, local and global features in one unified architecture. Our objective is to investigate if it is beneficial to *see* the preceding frames to the current frame of interest while making prediction on it. Moreover, we

show how our model can fully take advantage of the predictive nature of RNNs to generate a rough future estimation of the semantics of the scene based on what has been observed so far. We use an encoder-decoder convolutional LSTM architecture to first learn a representation of the input frames and then we generate a corresponding expected future.

We evaluated our method for both temporal and predictive semantic segmentation. Our results show that it is possible to train effective models using noisy data by leveraging inherent temporal information present in videos. We report a state-of-the-art result on challenging urban scene parsing datasets. Our method improves the average pixel accuracy score of the baseline non-temporal model by 4% on CamVid dataset and by 3% on Cityscapes dataset.

In summary, the main contributions in this work are:

(1) We propose a temporal pyramid network that utilizes multi-scale resolution feature maps and is capable of modeling the temporal dependencies.

(2) We show that our framework is suitable for the task of semantic scene understanding and achieves promising result.

(3) We demonstrate that our model can be extended easily to the problem of prediction where the semantics of the scene are estimated for the future frames.

The rest of this paper proceeds as follows: After an overview of related work in Sec. 6.2, we provide an overview of our noisy generated dataset used for temporal semantic segmentation in Sec 6.4.1. Sec. 6.3 provides some background on Recurrent Neural Nets, LSTMs and convolutional LSTMs and other relevant topics. We then introduce our proposed architecture in details in Sec. 6.4.3 and Sec. 6.4.4 for both temporal and predictive semantic segmentation. We then demonstrate the efficacy of the architecture for both methods through our experiments and show the qualitative and quantitative results in Sec. 6.5, and conclude with discussion in Sec. 6.6.

## 6.2 Related Work

Our temporal semantic segmentation work relates to a variety of research areas in computer vision such as image-based semantic segmentation and more remotely, to (unsupervised) video segmentation and temporal modeling for action recognition. In this section, we will only briefly review the vast literature with some attention towards formulations based on deep architectures which represent the foundation of our approach.

**Semantic Segmentation.** In recent years, deep neural networks have been extensively used in segmentation and achieved promising results on the accuracy and execution time. The most successful methods for semantic image segmentation are based on CNNs. In [143], color and texture features from oversegmented regions are merged by Recursive Neural Networks. This work has been extended by [142] who combined it with convolutional neural networks. [47] introduced multi-scale CNNs to learn scale-invariant features, but had problems with global contextual coherence and spatial consistency. These problems were addressed by combining CNNs with several post-processing algorithms, i.e., super-pixels, CRF, and segmentation trees. Likewise, [95] proposed a method which combines CNNs and CRFs to exploit complex contextual information for semantic image segmentation. We formulate CNN based pairwise potentials for modeling semantic relations between image regions. In [84], the reconstructed depth is used as a guide to produce a scale-invariant representation in which the appearance is decoupled from the scale. Also [99] addresses semantic image segmentation by incorporating rich information into Markov Random Field (MRF), including high-order relations and mixture of label contexts. In [26] applied the "atrous convolution"with upsampled filters for dense feature extraction and extended it to atrous spatial pyramid pooling, which encodes objects as well as image context at multiple scales.

Many semantic segmentation methods start with the VGG-16 network in [140] and refine it for semantic segmentation. In [169], they presented a module that replaces the pooling layers by dilated convolutions to systematically aggregate multi-scale contextual information without losing resolution. The architecture is based on the fact that dilated convolutions support exponential

expansion of the receptive field without loss of resolution or coverage. DeconvNet [117] learns deconvolution for semantic segmentation and is composed of deconvolution and unpooling layers, which identify pixel-wise class labels and predict segmentation masks. Similarly, SegNet [7] is composed of an encoder (stack of convolutions and pooling layers) followed by a decoder (stack of corresponding mirrored-convolution and unpooling layers) which perform the opposite operations. The decoders help map low resolution feature maps at the output of the encoder stack to the full input image size feature maps. [100] transformed pre-trained classification networks [140] into fully convolutional form to perform dense predictions. They augmented the architecture with a skip architecture that takes advantage of the full feature spectrum that combines deep, coarse, semantic information and shallow, fine, appearance information. In [65] residual connections are used making it possible to increase depth substantially. Another successful approach is to apply a dense conditional random field (CRF) [81] as a post-processing step on top of individual pixel or frame predictions. [25] use a fully convolutional network to predict a segmentation and then apply the dense CRF as a post processing step.

**Video Segmentation.** Video segmentation has received significant attention starting from early methodologies based on temporal extensions to normalized cuts [137], random field models and tracking [155, 91], motion segmentation [118] or efficient hierarchical graph-based formulations [59, 167, 53]. However, much of this work has been done to address the problem of segmentation in videos and the focus has not been on learning the semantics. .[86] used the dense CRF of [81] for semantic video segmentation by using pairwise potentials based on aligning the frames using optical flow. Nonetheless, these methods are not suitable for real-time or the complex multi-class, multi-object scenes in the real world problems.

**Recurrent Convolutional Neural Networks.** In order to improve modeling of long range dependencies, [123] first revealed the use of large input patches to consider larger contexts. However, that would reduce the resolution of the final label image, and a huge redundancy of overlapping regions would make the learning inefficient. Hence, they introduced Recurrent Convolutional Networks (RCNNs) for scene labeling. RCNNs train different input patch sizes (the instances)

recurrently to learn increasingly large contexts for each pixel, whilst ensuring that the larger contexts are coherent with the smaller ones. LSTMs of recurrent neural networks [69] were originally introduced for sequence learning. These networks include recurrently connected cells to learn the dependencies between two time frames, then transfer the probabilistic inference to the next frame. The LSTM memory block stores and retrieves this information over short or long periods of time. LSTM networks have been successfully applied to many tasks such as handwriting [57] and speech recognition [55, 56], machine translation [150, 30], and caption generation for images [156]. They have also been applied on videos for recognizing actions and generating natural language descriptions [43]. A general sequence to sequence learning framework was described by [150] in which a recurrent network is used to encode a sequence into a fixed length representation, and then another recurrent network is used to decode a sequence out of that representation. In [145] used multilayer Long Short Term Memory (LSTM) networks to learn representations of video sequences. In [166] By extending the fully connected LSTM (FC-LSTM) to have convolutional structures in both the input-to-state and state-to-state transitions, they proposed the convolutional LSTM (ConvLSTM) and use it to build an end-to-end trainable model for the precipitation nowcasting problem. In [125] proposed a strong baseline model for unsupervised feature learning using video data. By learning to predict missing frames or extrapolate future frames from an input video sequence, the model discovers both spatial and temporal correlations which are useful to represent complex deformations and motion patterns.

**Action Recognition.** Temporal modeling using deep architectures is also an important topic in action recognition. [75] proposes 3D convolutional neural networks which are a natural extension of a standard 2D convolutional network for action recognition. However, it increases the learning complexity, as the filters need to model both appearance and motion variations. To avoid having to deal with this added complexity of spatio-temporal convolutional filters, [139] proposed a two streams architecture to predict actions. They process images and optical flow independently in two different streams which is merged at a later stage in order to predict the final action label. To account for the lack of training data a multi-task setting is proposed, where the same network

is optimized for two datasets simultaneously. Later, [170] implemented a network in which the output of the two streams is sent into a recurrent neural network, where, at later stages the features are pooled to produce a final labeling. [80] uses frames at different time-steps and merges them as input to a neural network in different fashions. [148] attempt to make the problem of learning spatio-temporal features easier by using separable filters. More specifically, they apply spatial filters first and then temporal filters after that to avoid learning different temporal combinations for every spatial filter. [9] spatio-temporal filters by stacking GRUs at different locations in a deep neural network and iterating through video frames.

## 6.3 Preliminaries

In this section, We first briefly review feature learning by convolutional networks. Then, we introduce the fundamentals of recurrent networks in Section 6.3.1, in particular those built on Long Short Term Memory(LSTM) units, which are well suited to model temporal models. In Section 6.3.2, we review Convolutional LSTM units which are the core of our temporal architecture. Finally, we review Batch Normalization in Section 6.3.3.

### 6.3.1 Learning via Deep Neural Networks

A multilayer perceptron, is a feedforward computational neural network that processes information through interconnected nodes and maps a set of input data onto a set of desired output. A layer is formed by grouping these computational nodes together and connecting them with some weights. These nodes are called neuron units which mainly transform the information between layers by applying non-linear operations on it to create a decision boundary for the input. Here the goal is to stretch and squish the input data by applying a series of non-linearities and finally projecting it into a space where it preserves the topological properties but becomes linearly separable. These networks have been successfully applied to many classification problems by supervised

training. Formally, units are defined in terms of the following function [120]:

$$X^{l+1} = \sigma(W^l X^l + b^l) \tag{6.1}$$

where $X^l$ denotes the activation map in layer $l$ ($X_i^l$ is the activation for the unit $i$ in layer $l$), $W$ is a weight matrix, where $W_{ij}^l$ $ij$ represents the parameter associated with the connection between unit $j$ in layer $l$, and unit $i$ in layer $l+1$, $b^l$ is the bias associated with units in layer $l$ and $\sigma$ is the activation function. $X$ is the input data of the network when $l = 1$. The activations of the units in the deepest(last) layer will give the the output of the network. It is common to fully-connect the neurons in these networks, in which each node in layer $l+1$ connected with every node in layer $l$.

On the other hand, a Convolutional Neural Network (CNNs) [89] with a single layer extracts features from the input by applying a convolution operation on the input and kernel filter. In CNNs, the activations in each layer show the result of the convolution of the kernel and the input image. similar to sliding window approaches, it is possible to detect a particular pattern captured by a specific filter, regardless of where the pattern occurs by computing the activation of a unit on different regions of the same input which results in feature maps. The advantage of CNNs is that the weight matrix and bias parameters are shared between units. Therefore, CNNs have the ability to learn complex data representation by passing the information through multiple layers of feature hierarchies which increases the levels of abstraction in deeper levels.

Even though these networks are very powerful for learning complex systems, one major limitation is that they assume that all inputs and outputs are independent of each other in time. To overcome this problem, Recurrent Neural Networks (RNNs) are proposed which use a recurrent connection in every unit. Here the idea is to learn the dynamics of temporal data by feeding back the activations of a layer to itself by some time delay. This is similar to having a memory that keeps track of past activations. A representation of a single recurrent unit is shown in Figure 6.2.

Given a temporal input sequence $X^l = (X_1^l, ..., X_T^l)$ of length $T$ (being $X_{ti}^l$ the activation of the unit $i$ in hidden layer $l$ at time $t$), an RNN maps it to a sequence of hidden values $h^l = (h_1^l, ..., h_T^l)$ and outputs a sequence of activations $X^(l+1) = X_1^(l+1), ..., X_T^(l+1)$ by iterating the following

recursive equation [120]:

$$h_t^l = \sigma(W_{xh}^l X_t^l + h_{t-1}^l W_{hh}^l + b_h^l) \tag{6.2}$$

where $\sigma$ is the non-linear activation function, $b_h^l$ is the hidden bias vector and $W$ terms denote weight matrices, $W_{xh}^l$ being the input-hidden weight matrix and $W_{hh}^l$ the hidden-hidden weight matrix. The activation for these recurrent units is defined by:

$$X_t^{(l+1)} = h_t^l W_{hx}^l + b_x^l \tag{6.3}$$

where $W_{hx}^l$ denotes the hidden-activation weight matrix and the $b_x^l$ terms denote the activation bias vector. Notice that the weight matrix $W^l$ defined for the multilayer perceptron is equivalent to the $W_{xh}^l$ matrix in Equation 6.2.

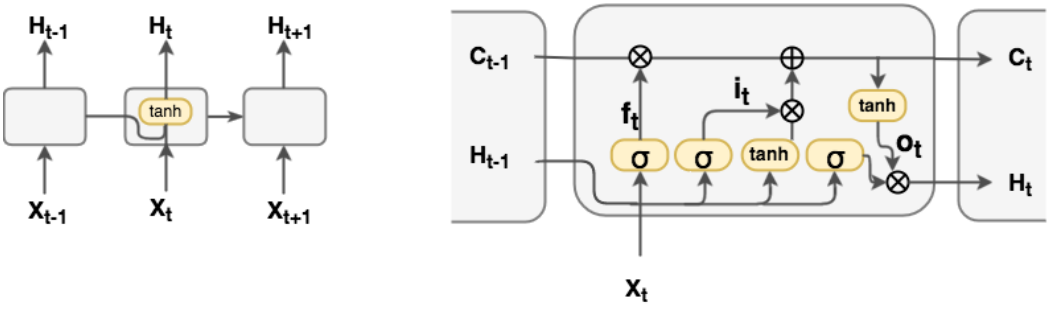


Figure 6.2: The internal structure of RNN and LSTM units

Furthermore, Long short term Memory(LSTM) units extend RNN with memory cells that enables us to learn temporal relationships on long sequences. LSTMs introduce the concept of gating and cell state. The LSTM updates the cell state, according to the activation of its internal gates. The input provided to an LSTM is fed into different gates: The forget gate $f$, decides what parts of the input data are not necessary and need to be forgotten, the input gate $i$ decides what parts need to be updated and written to the cell state based on the input, whether the latest cell output $c$ will be propagated to the final state $h$ is further controlled by the output gate $o$.. The activation of

the LSTM units is calculated as in the RNNs. See Figure 6.2 for a detailed illustration of LSTM units. The computation of the hidden values $h_t$ and $c_t$ of an LSTM cell is updated at every time step $t$. The below equations show the update of an LSTM layer [120] :

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \odot c_{t-1} + b_i) \tag{6.4}$$

$$ft = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \odot c_{t-1} + b_f) \tag{6.5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}ht - 1 + b_c) \tag{6.6}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \odot c_t + b_o) \tag{6.7}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6.8}$$

where $i$, $f$, $o$ and $c$ are respectively the input gate, forget gate, output gate and cell activations, and they all have the same size as vector $h$ defining the hidden value. Terms $\sigma$ and tanh represent non-linear functions used in LSTM units. The term $x_t$ is the input to the memory cell layer at time $t$. $W$s are weight matrices, with subscripts representing from-to relationships ($W_{xi}$ being the input-input gate matrix, $W_{hi}$ the hidden-input gate matrix, and so on). $b_i$ , $b_f$ , $b_c$ and $b_o$ are bias vectors. $\odot$ denotes element-wise product of vectors.

One important innovation of LSTMs is the memory cell $c_t$ which essentially is an accumulator of the state information through time. RNNs are known to have exploding gradient problem whereas LSTMS use memory cells to trap the gradients in the cell and prevent them from vanishing or exploding too quickly. Therefore, LSTMs are capable of modeling long-range temporal dependencies in sequences. Every time a new input comes, the internal gates control what part of information is necessary to be kept and forgotten and its information will be accumulated to the cell. In order to model more complex temporal structures, usually multiple LSTM layers are stacked on top of each other similar to multi-layer CNNs.

The combination of CNNs and LSTMs in a unified framework is able to capture time dependencies on features extracted by convolutional operations which has already showed state-of-the-art performance in the speech recognition domain [135].

### 6.3.2 Convolutional LSTMs

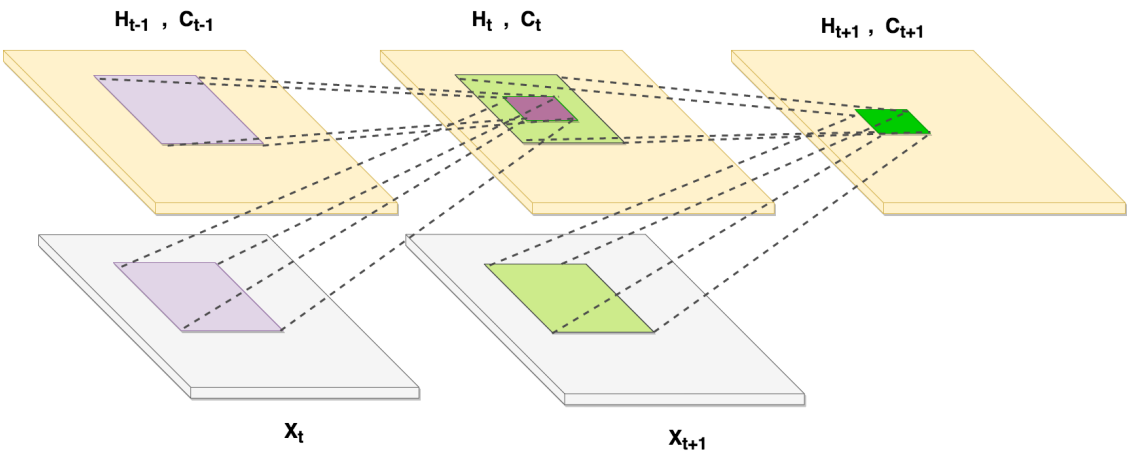


Figure 6.3: Illustration of Convolutional LSTM networks

The conventional fully connected LSTMs (FC-LSTMs) explained in the previous section have one major drawback when dealing with spatio-temporal data. Since conventional LSTMs use dense connections, the spatial information of the data will not be preserved. To overcome this problem, we are using a different architecture called Convolutional LSTMs (Conv-LSTMs) that are first introduced in [166]. Although the FC-LSTMs have proven to be powerful tools for modeling temporal correlations in temporal data, they are incapable of maintaining the locality and spatial structure of the data. This is specially very important for the task of semantic scene understanding, since we do not want to lose the spatial information existent in the scene. Moreover, FC-LSTMs generally have a large number of parameters and are prone to over-fitting. Conv-LSTMs replace the internal dense, fully connected layers of the unit with convolutional operations. The Conv-LSTM determines the future state of a certain cell by the inputs and past states of its local neighbors [166]. In other words, the Conv-LSTM learns to model the temporal information by looking at the learned spatial representation of its neighbors instead of a flat representation that does not preserve spatial information. The key equations of Conv-LSTM are shown below, where $*$ denotes the convolution operator and $\circ$, as before, denotes the Hadamard product [166]:

$$i_t = \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} \odot c_{t-1} + b_i) \tag{6.9}$$

$$ft = \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + W_{cf} \odot c_{t-1} + b_f) \tag{6.10}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{xc} * x_t + W_{hc} * ht - 1 + b_c) \tag{6.11}$$

$$o_t = \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{co} \odot c_t + b_o) \tag{6.12}$$

$$h_t = o_t \odot tanh(c_t) \tag{6.13}$$

where $\sigma$ and tanh are the non-linear functions. Respectively, $i_t$, $f_t$, $o_t$, $c_t$ and $h_t$ are vectors to represent values of the input gate, forget gate, output gate, cell activation, and cell output at time $t$. $W$s are the filter matrices connecting different gates, and $b$s are the corresponding bias vectors. The main difference is that  shows a convolution operation, where in a FC-LSTM  is a matrix multiplication. Figure 6.3 shows the internal structure of a convolutional LSTM inspired by [166]. Using Conv-LSTMs instead of FC-LSTMs, we expect to learn better temporal representations for the task of semantic segmentation while being less prone to over-fitting.

### 6.3.3 Batch Normalization

Batch normalization (BN) [73] is a widely used technique in the computer vision community which helps with to accelerate the training process and also improves generalization. Given a layer with output $x$, it is implemented by normalizing each layers inputs such that:

$$BN(x) = \gamma \frac{x - E[x]}{(Var[x] + \varepsilon)^{1/2}} + \beta \tag{6.14}$$

where $\gamma$ and $\beta$ are the parameters being learned. The idea is to compute the mean and variance statistics across all timesteps in the batch. During inference, these statistics are being used to further improve the results.

### 6.3.4 Baseline Semantic Segmentation Networks

**Fully Convolutional Networks.** In [100], previously designed classification networks such as AlexNet [85], the VGG net [140], and GoogLeNet [151] were adapted into fully convolutional networks by replacing the fully connected layers. They augment these networks for dense prediction and transfered the learned representations by fine-tuning to the segmentation task. They obtain a coarse label map from the network by classifying every local region in image, and perform a simple deconvolution, which is implemented as bilinear interpolation, for pixel-level labeling. Also by using a skip architecture, they combine semantic information from a deep, coarse layer with information from a shallow, fine layer to produce accurate and detailed segmentations. Conditional random field (CRF) can also optionally be applied to the output map for fine segmentation [81]. The main advantage of the methods based on FCN is that the network accepts a whole image as an input and performs fast and accurate inference.

**Encoder-Decoder Networks.** Networks such as SegNet[7] and DeconvNet [117] are composed of a stack of encoders followed by a corresponding decoder stack which feeds into a softmax classification layer. The encoder network corresponds to feature extractor that transforms the input image to multidimensional feature representation, whereas the decoder network is a shape generator that produces object segmentation from the feature extracted from the convolution network. The decoders help map low resolution feature maps at the output of the encoder stack to full input image size feature maps. Each encoder performs dense convolutions, ReLU non-linearity, a non-overlapping max pooling and finally down-sampling. Each decoder upsamples its input using the memorized pooled indices and convolves it with a trainable filter bank. The final output of the network is a probability map in the same size to input image, indicating probability of each pixel that belongs to one of the predefined classes. The key learning module is an encoder-decoder network where the encoder consists of a filter bank convolution, tanh squashing function, max pooling followed by sub-sampling to obtain the feature maps. The main difference between SegNet[7] and DeconvNet [117] is the implementation details in the decoder network. They both employ VGG-

16 net [140] for the encoder part where SegNet removes all the layers after pool5 and DeconvNet only removes the last classification layer. In the decoder network, They both mirror the convolution network of the encoder network. They both have unpooling and rectification layers in conjunction with these layers. Pooling in convolution network is designed to filter noisy activations in a lower layer by abstracting activations in a receptive field. Unpooling layers on the other hand, perform the reverse operation of pooling and reconstruct the original size of activations. It records the locations of maximum activations selected during pooling operation in switch variables, which are employed to place each activation back to its original pooled location. The output of an unpooling layer is an enlarged, yet sparse activation map. SegNet applies convolutional layers whereas DeconvNet uses deconvolution layers for this purpose. Contrary to convolution network that reduces the size of activations through feed-forwarding, The deconvolution layers densify the sparse activations obtained by unpooling through convolution-like operations with multiple learned filters. However, contrary to convolutional layers, which connect multiple input activations within a filter window to a single activation, deconvolutional layers associate a single input activation with multiple outputs.

**Deep Residual Networks.** Residual connection were introduced in [65] in which they describe the advantages of applying additive merging of feature maps both for image recognition, and especially for object detection. They proposed a residual learning framework to facilitate the training of networks that are much deeper compared to the previous networks. They formulate the layers as learning residual functions instead of learning unreferenced functions. The motivation behind this work is to deal with the common degradation problem in deep networks. The authors argue that the degradation is not caused by over-fitting surprisingly and address this problem by introducing a residual learning framework. Instead of trying to fit a desired underlying mapping for each few stacked layers directly, they explicitly let these layers fit a residual mapping. Formally, lets denote the desired underlying mapping as $H(x)$. Then the stacked nonlinear layers fit another mapping of $F(x) := H(x)x$ and the original mapping is converted to $F(x) + x$. They show that it is easier to optimize the residual mapping compared to the original, unreferenced mapping.

The formulation is in nature very similar to shortcut connections. Here, the shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers. See Figure 6.4 for the details of residual learning blocks.

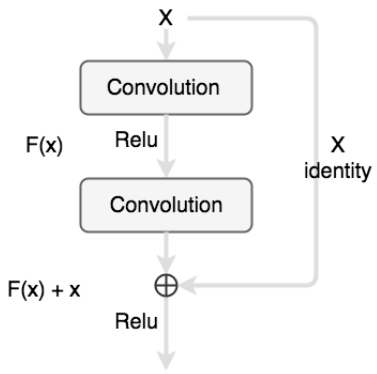


Figure 6.4: Building block of residual learning

## 6.4 Approach

### 6.4.1 Data Augmentation

The enormous success of deep architectures has been gained by training on very large datasets. Due to the substantial human effort involved in producing pixel-accurate annotations, semantic segmentation datasets with precise and comprehensive label maps are orders of magnitude smaller than image classification datasets. High-quality semantic labeling was reported to require 60 minutes per image for the CamVid dataset [15] and 90 minutes per image for the Cityscapes dataset [33]. Note that the labeled images in these datasets are either chosen randomly (Cityscapes) or at a fixed Hz (CamVid) from a video sequence which makes it not suitable for tasks that require all the frames in the video sequence. The inherent lack of temporal labeled data motivated us to design a fast and simple method for propagating the semantic labels from the labeled frames to unlabeled ones in a sequence. This gives us the opportunity to incorporate the temporal information of all the frames in the sequence and also generate orders of magnitude more

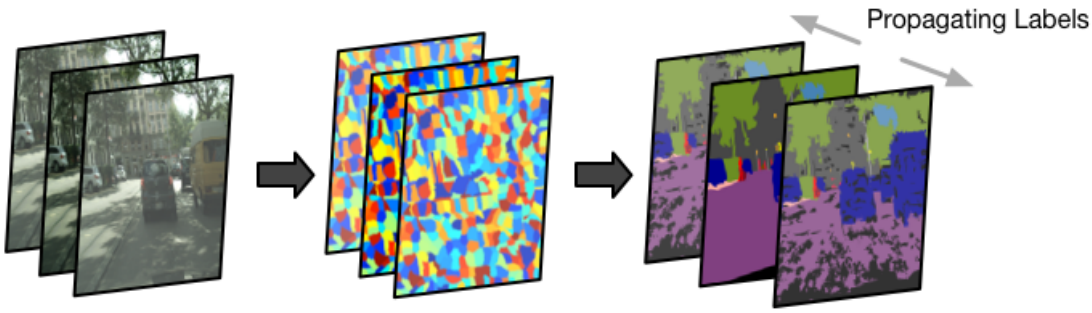


Figure 6.5: Our label propagation scheme

data to overcome problems such as overfitting. The main difference between the two dataset is that the annotation is available for only one frame in a sequence of size 30 (20th frame) in Cityscapes dataset whereas the annotation is available for both the beginning and the end of a sequence in Camvid dataset.

Our label propagation pipeline starts with oversegmenting the sequence of frames to obtain superpixels that are coherent in time and space (see [53] for more details). After that we transfer the semantic labels from the groundtruth annotation to all the pixels that belong to the same spatio-temporal superpixel. Using this method, we produced approximately 105K annotated frames from available annotated frames in the CityScapes dataset and 18K annotated frames from CamVid dataset in significantly smaller amount of time compared to the reported numbers for these datasets. In the scenes that there was no fast motion, we were able to recover the semantic labels for most of the pixels. However for the pixels that are not present in the groundtruth annotation and emerge or disappear later in the future, we use the void label.

### 6.4.2 Network Architecture

While state-of-the-art networks, FCN [100], SegNet[7] and DeconvNet [117] use the identical convolutional(encoder) network based on VGG-16 layers [140], they approach deconvolution(decoder) differently. FCN performs a simple deconvolution using a single bilinear interpolation layer, and produces coarse, but clean overall shape segmentations, because the output layer is closely connected to the convolution layers preventing the loss of spatial information. The absence of a deep deconvolution network trained on a large dataset makes it difficult to reconstruct highly non-linear structures of object boundaries accurately. DeconvNet and SegNet on the other hand, mirror the convolution process with multiple series of unpooling, deconvolution, and rectification layers, and generates detailed segmentations at the cost of increased noise. Instead, We combine the ideas from both of these approaches and develop a deep recurrent multi-scale convolutional network for semantic segmentation. We exploit the inherent multi-scale, pyramidal hierarchy of convolutional networks and combine them with recurrent layers to model the input both in space and time. The convolutional layers act as spatial feature extractors and provide abstract representations of the input data whereas the convLSTMs model the temporal dynamics of the scene. Our network is able to capture both spatial and temporal dynamics of the scene, and uses a pyramidal structure to benefit from the multi-scale contextual information. The output probability map resulting from a bilinear interpolation of different levels of the pyramid are then fused together before a final convolutional layer merges them into a single high-fidelity output map. In Sections 6.4.3 and 6.4.4 we provide the details of our proposed network for temporal and predictive semantic segmentation.

### 6.4.3 Proposed Temporal Network

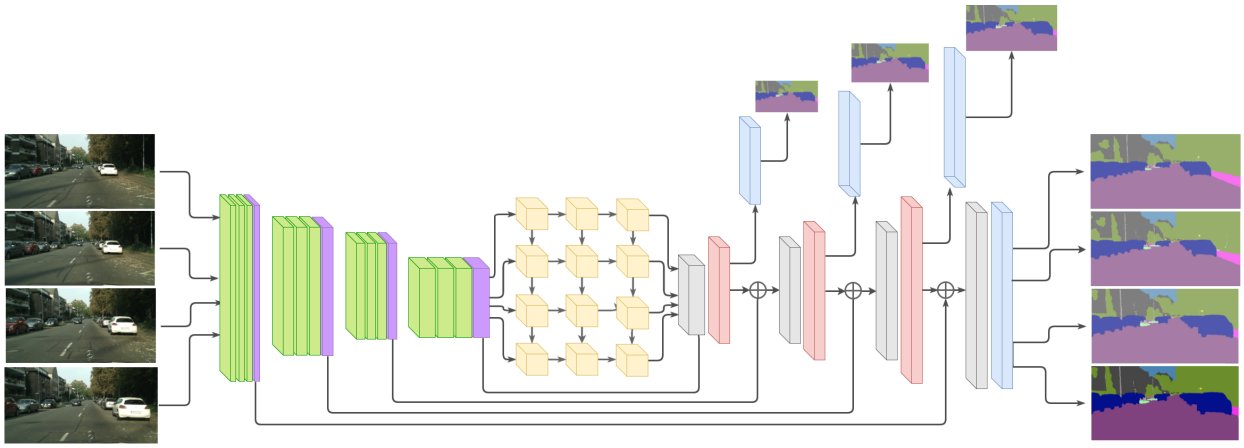

Figure 6.6: Details of our temporal semantic segmentation model; Green, purple, yellow gray, pink and blue are $3 \times 3$ convolution, pooling, convolutional LSTM, $1 \times 1$ convolution, upsampling and Softmax layer respectively.

Our network is designed based on a VGG-16 architecture. However, further modifications to the VGG network is needed in order to combine the temporal information and prediction of different levels in our network. The detailed configuration of our proposed network is shown in 6.6. For the encoder network, we keep all the convolution layers of VGG-16, followed by batch normalization and ReLu layers. We remove the fully connected layers and add multiple layers of convLSTMs in the bottleneck. We use a pyramidal multi-scale architecture to combine the feature maps obtained in the encoder part to obtain the final prediction. First we apply a $1 \times 1$ convolution to the feature map of each pooling layer in the network to change their dimensions into the same size ($c = 256$) and form a pyramid of feature maps in different scales. The convLSTM layers are added to middle of the network with a skip connection to preserve the the spatial information learned from previous layers. While the original FCN uses the last fully connected layer as the coarsest prediction, we instead use the output of the convLSTM layer, as the coarsest level. We start by the coarsest level that has the lowest resolution feature map , and sequentially upsample

the activations to be aligned with the previous feature map and fuse them together. We add another $1 \times 1$ convolution to the resulting feature map before upsampling it since it enable us to learn blending weights between the two activations as convolution parameters. We also add a loss layer to each individual layer and obtain a prediction to ensure that the network is learning meaningful representation of data at each level.

### 6.4.4 Proposed Predictive Network

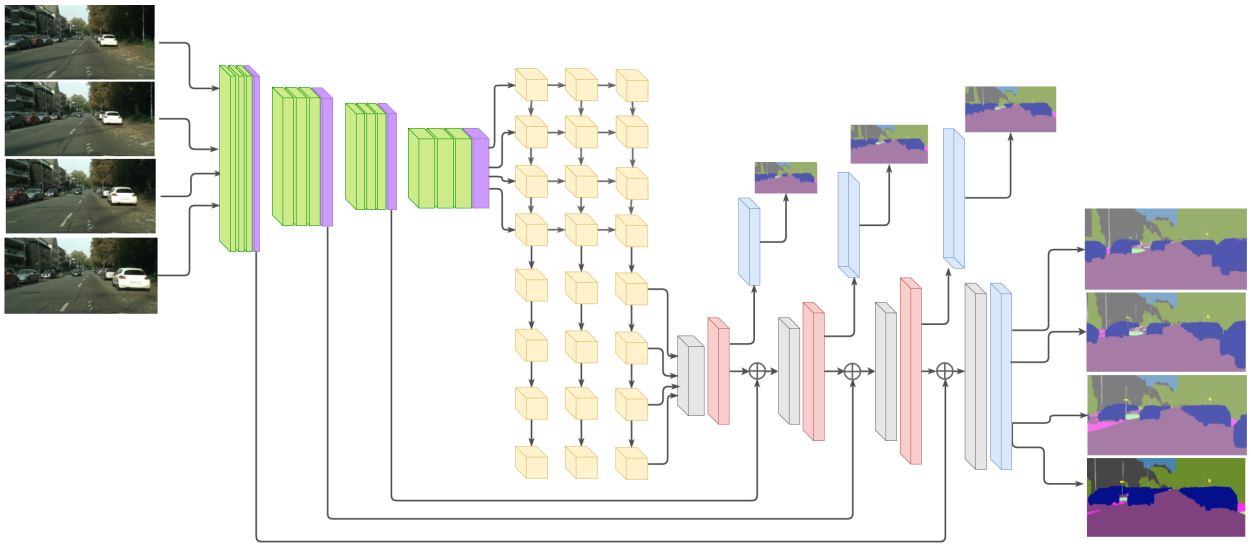

Figure 6.7: Details of our predictive semantic segmentation model; Green, purple, yellow gray, pink and blue are $3 \times 3$ convolution, pooling, convolutional LSTM, $1 \times 1$ convolution, upsampling and Softmax layer respectively.

Our predictive semantic segmentation network has been shown in Figure 6.7. Here, the motivation is to learn a representation that extracts all that is needed to extrapolate the motion and appearance beyond what has been observed. The model works as follows, the encoder CNNs learn the spatial features of the input sequence and then it runs through an encoder convLSTMs to add the temporal features and compresses it into hidden state representation. This representation is

then being unfolded by the decoder convLSTMs and generates an output sequence which again runs through the decoder CNNs to map the low resolution spatio-temporal feature to the full input resolution. Both encoder and decoder networks are formed by stacking several convLSTM layers. We fully adapt VGG-16 network and remove the fully connected layers after the last pooling layer, add the encoder/decoder convLSTM layers in the bottleneck. We simply mirror the convolutional layers of the encoder to recover the resolution of input in the decoder network. We trained the model to predict the output sequence that has the same dimensionality as the input. Since the network has multiple stacked ConvLSTM layers, it has strong representational power which makes it suitable for giving strong predictions. This structure is also similar to the LSTM future predictor model in [145] and [166]. The frames in the input sequence of our model may correspond to every *n*th frame of the video, as long as there is enough resemblance in the structure of the scene shared by the frames. The details of our network is shown in 6.7

## 6.5 Experiments

**Datasets.** The Cityscapes dataset [33] contains high quality pixel-level annotations of images collected in street scenes from 50 different cities. The training, validation, and test sets contain 2975, 500, and 1525 images respectively. This dataset contains labels for 19 semantic classes belonging to 7 categories of ground, construction, object, nature, sky, human, and vehicle. The images of Cityscapes are high resolution ($1024 \times 2048$) which makes training challenging due to limited GPU memory. We trained our models on random crops of size $256 \times 512$ of the resized images. The CamVid dataset [15] has over ten minutes of 30Hz high quality $720 \times 960$ footage, with 701 pixel-level annotated images chosen at 1Hz and in part, 15Hz. We trained our models on random crops of size $360 \times 480$ of the resized images. For both of these datasets, we increased the amount of training data by transferring the labels with the method explained in Sec. 6.4.1.

**Training and Optimization.** Figure 6.6 and 6.7 show the detailed configuration of the proposed temporal and predictive networks in details. All models were trained and tested on a single NVIDIA TITAN X GPU. The network parameters are optimized by minimizing the cross-entropy

loss function (Softmax classifier) using mini-batch stochastic gradient descent. As per suggested by the experiments of Chapter 5, we are using weighted cross-entropy loss when using the noisy groundtruth. We set base learning rate to $10^{-3}$ and decreased the learning rate over time where current learning rate equals to the base one multiplying by $(1 - \frac{iter}{maxiter})^{power}$. We set the power to 0.9 and use a weight decay of 0.0001 with a momentum of 0.9 for optimization. Batch normalization is used in all of our experiments and the batch size is set to be 4. We initialize the weights using pre-trained models for classification in all the networks. We also use a dropout layer on the output representation of multi-layer convLSTMs, as a form of regularization. Our implementation is based on the public platform Caffe [76] . The number of parameters to optimize varies according to the type of layers it comprises and has great impact in the time and computer power required to train the networks. To prevent overfitting, we augment our dataset with additional images using random perturbations of translation, scale, mirroring and cropping. We also randomly resize the input images from the original image resolution prior to randomly crop them to the input size for the network. This comprehensive data augmentation scheme makes the network resist overfitting. For the auxiliary loss, we set the weight to 0.4 in all experiments.

**Temporal Segmentation.** In order to show the benefits brought by our network, we compare it to "baseline"non-recurrent deep CNN that shares the same architecture but without the convLSTM layers as well as contemporary CNNs that are widely used for semantic segmentation such as FCN, SegNet and ResNet. The input to our network is a sequence of $n$ consecutive frames, and the output is the semantic segmentation of the last frame. Due to the memory limits, we could not experimeent with large sequences. We set $n$ to be a random length between 1 and 4. Here, the goal is to investigate whether incorporating the temporal information of the prior frames is in fact improving the accuracy of prediction or not. The main difference between our architecture and the baseline CNNs is the topology of layers in the bottleneck. In the case of ours, the units of these layers are convLSTM recurrent cells, and in the case of the baseline model, the units are non-recurrent and either convolutional or fully connected. Therefore, performance differences between the models are a product of the architectural differences and not due to better optimization,

preprocessing or ad-hoc customization. In Table 6.2 and 6.3 the accuracy of segmentation for each semantic class in Cityscapes dataset is shown. Also the mean IOU and overall pixel accuracy scores have been detailed in 6.3. We achieve 94.7% pixel accuracy on Cityscapes dataset and 91.3% pixel accuracy on CamVid dataset that is 3-5% higher than the baseline. We showed that our temporal method is superior to both our non-temporal baseline and also other state-of-the-art methods for both datasets. Examples of semantic segmentation results on the validation images are shown in Figure. 6.9 on Cityscapes dataset for a sequence of frames. Also Figure. 6.11 and Figure. 6.10 show the comparison of our approach with Segnet and FCN respectively. Figure 6.8 shows the qualitative results of our approach and other methods on CamVid dataset.

Table 6.1: Comparison of the mean IoU and pixel accuracy on CityScapes dataset.

| Method | IOU | Pixel Accuracy | Precision | Recall |
|---|---|---|---|---|
| FCN-8 | 44.0 | 88.3 | 63.1 | 60.4 |
| Segnet | 45.4 | 90.1 | 64.1 | 57.4 |
| ResNet-50 | 55.1 | 92.1 | 71.1 | 67.5 |
| Our baseline model | 55.8 | 92.5 | 73.9 | 66.8 |
| Our temporal model | 57.4 | 94.7 | 78.4 | 70.3 |

**Predictive Segmentation.** For the task of predicting future frames, we consider a sequence of *n* consecutive frames preceding to the groundtruth. We feed the first half of the frames to our network and ask the network to predict the semantic segmentation for the second half. during training, network *only* sees the first half of frames and the groundtruth labels for the second half (i.e it does not have access to the ground truth of the first half or the raw frames of the second half). This forces the network to learn a meaningful representation of the video and also gives us the opportunity to evaluate our predictive result by comparing against the available groundtruth

for the last frame in the sequence. In our experiments $n$ is set to be 8 for both Cityscapes and CamVid dataset. We achieve 87.4% pixel accuracy on Cityscapes and 85.1% pixel accuracy for CamVid dataset. Figure 6.12 shows qualitative result for out predictive semantic segmentation in Cityscapes dataset. Note that the network has nicely captured the dominant motion in the scene and predicted a reasonable future scene in most cases (i.e the car that is moving to the left in the first row of Figure 6.12). We believe that it is possible to achieve even better results by having larger and more diverse datasets.

Table 6.2: Comparison of the results on CamVid dataset

| Method | building | tree | sky | car | sign symbol | pedestrian | road | fence | column-Pole | sidewalk | bicyclist |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Segnet | 75.0 | 84.0 | 91.2 | 82.7 | 36.9 | 93.3 | 55.3 | 37.5 | 44.8 | 74.1 | 16.0 |
| FCN | 70.3 | 79.0 | 88.9 | 78.8 | 31.8 | 90.1 | 56.0 | 0.38 | 45.3 | 70.5 | 14.5 |
| ResNet | 78.2 | 85.8 | 92.5 | 84.2 | 36.4 | 93.0 | 56.6 | 40.1 | 44.9 | 78.0 | 18.3 |
| Our baseline model | 79.1 | 85.2 | 93.0 | 85.1 | 35.8 | 94.7 | 57.3 | 40.1 | 43.8 | 81.2 | 16.9 |
| Our temporal model | 81.7 | 87.1 | 93.7 | 86.3 | 37.6 | 94.3 | 60.2 | 39.2 | 45.1 | 81.9 | 18.7 |

Table 6.3: Comparison of class IoU on CityScapes dataset.

| Method | road | sidewalk | building | wall | fence | pole | traffic light | traffic sign | vegetation | terrain | sky | person | rider | car | truck | bus | train | motorcycle | bicycle |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FCN-8 | 86.1 | 53.2 | 74.3 | 12.4 | 15.2 | 18.5 | 10.8 | 41.0 | 77.1 | 40.7 | 89.0 | 48.9 | 29.8 | 83.9 | 20.3 | 31.0 | 11.1 | 13.1 | 48.3 |
| Segnet | 87.9 | 57.9 | 77.3 | 20.8 | 23.4 | 30.9 | 17.8 | 39.2 | 83.4 | 40.9 | 87.6 | 54.5 | 31.1 | 80.3 | 22.2 | 35.2 | 9.7 | 12.7 | 50.9 |
| ResNet-50 | 89.9 | 62.5 | 80.4 | 39.3 | 31.0 | 35.1 | 33.5 | 42.3 | 86.0 | 49.0 | 87.8 | 57.6 | 38.5 | 86.7 | 47.3 | 61.1 | 45.7 | 25.8 | 54.1 |
| Our baseline model | 89.6 | 65.0 | 81.9 | 30.5 | 29.7 | 42.4 | 38.7 | 53.8 | 86.9 | 43.3 | 88.6 | 65.8 | 41.6 | 88.7 | 39.7 | 50.1 | 25.9 | 33.1 | 58.4 |
| Our temporal model | 90.1 | 70.5 | 86.8 | 33.5 | 31.4 | 43.8 | 41.2 | 55.9 | 88.1 | 47.0 | 91.8 | 63.1 | 46.4 | 91.7 | 44.2 | 50.3 | 28.3 | 35.6 | 59.3 |

## 6.6 Summary

We have proposed an effective network for understanding and labeling urban scenes. We combined convolutional layers with LSTMs in a pyramidal architecture. The CNNs are used to reduce the spatial variation of the input features and the LSTM layers are used for modeling the temporal dynamics of the scene. Moreover, we use a pyramidal structure to combine the prediction of different scale feature maps to capture contextual information at different in the network to obtain the final prediction. Our method shows significant improvements over several strong baselines. We show that our model can be easily extended to the related problem of prediction to estimate the expected semantics of the scene a small number of frames into the future, this is helpful for robotic applications such as free-space estimation in autonomous driving. We achieve promising state-of-the-art results on various datasets and prove that our temporal approach is superior to the non-temporal baseline. The strategies introduced in this work, can be easily applied to other contemporary networks designed for other visual tasks such as detection. We believe that using the temporal information along with spatial features in different scales helps to improve the performance of deep learning models.

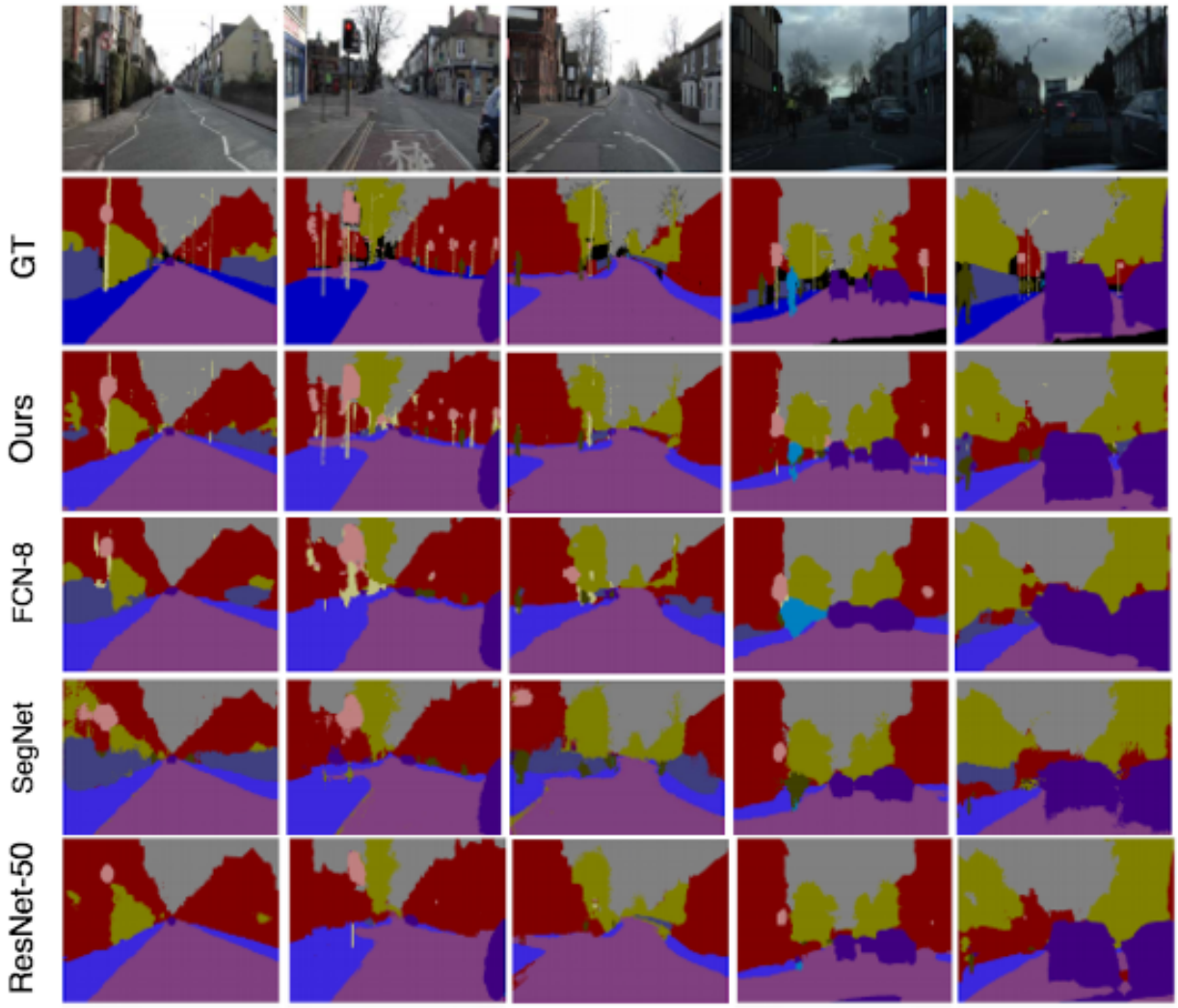


Figure 6.8: Comparison of qualitative results on CamVid dataset

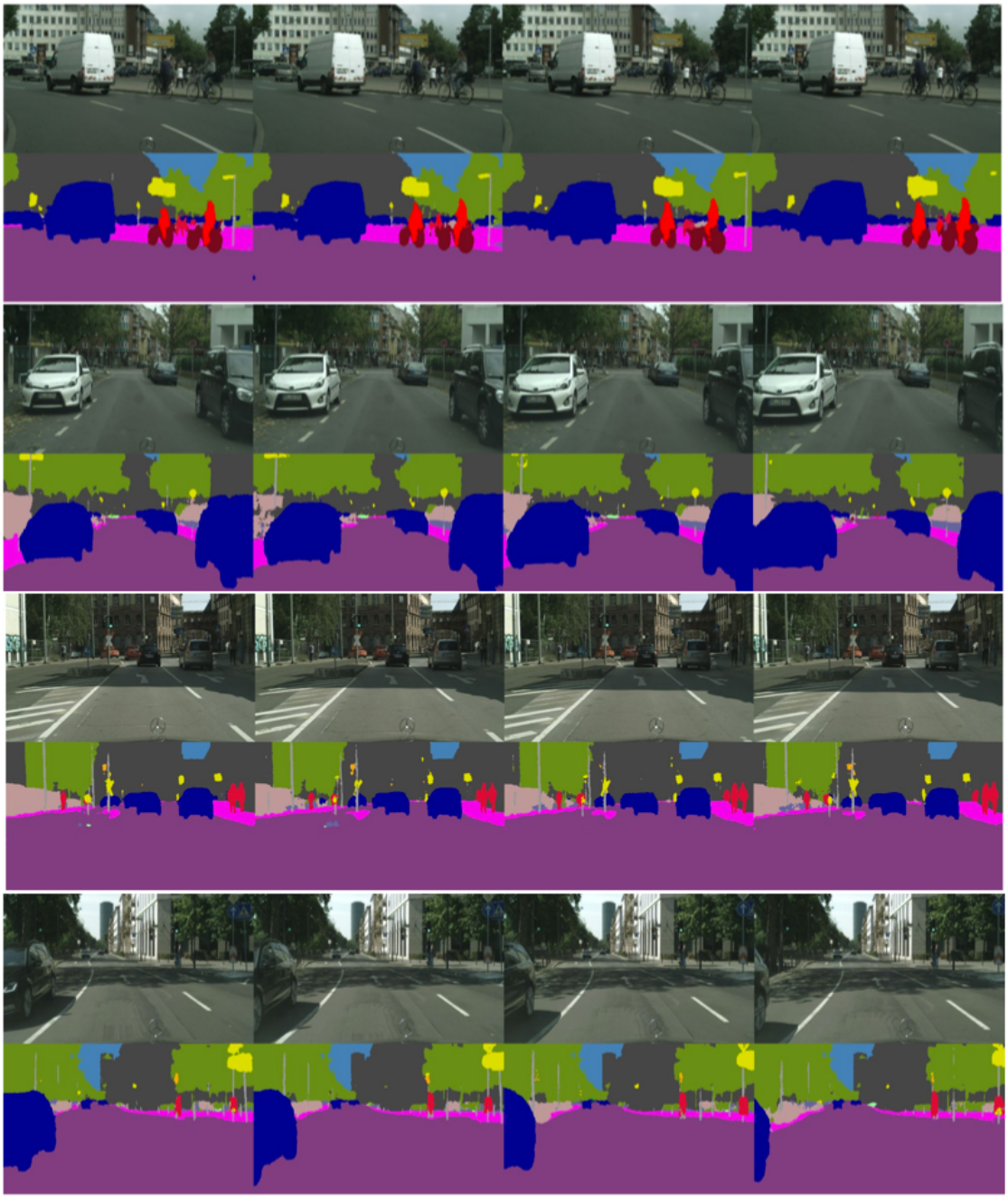

Figure 6.9: Qualitative results of our model on CityScapes

Figure 6.10: Qualitative results on Cityscapes for FCN(on the right) and our model(on the left)

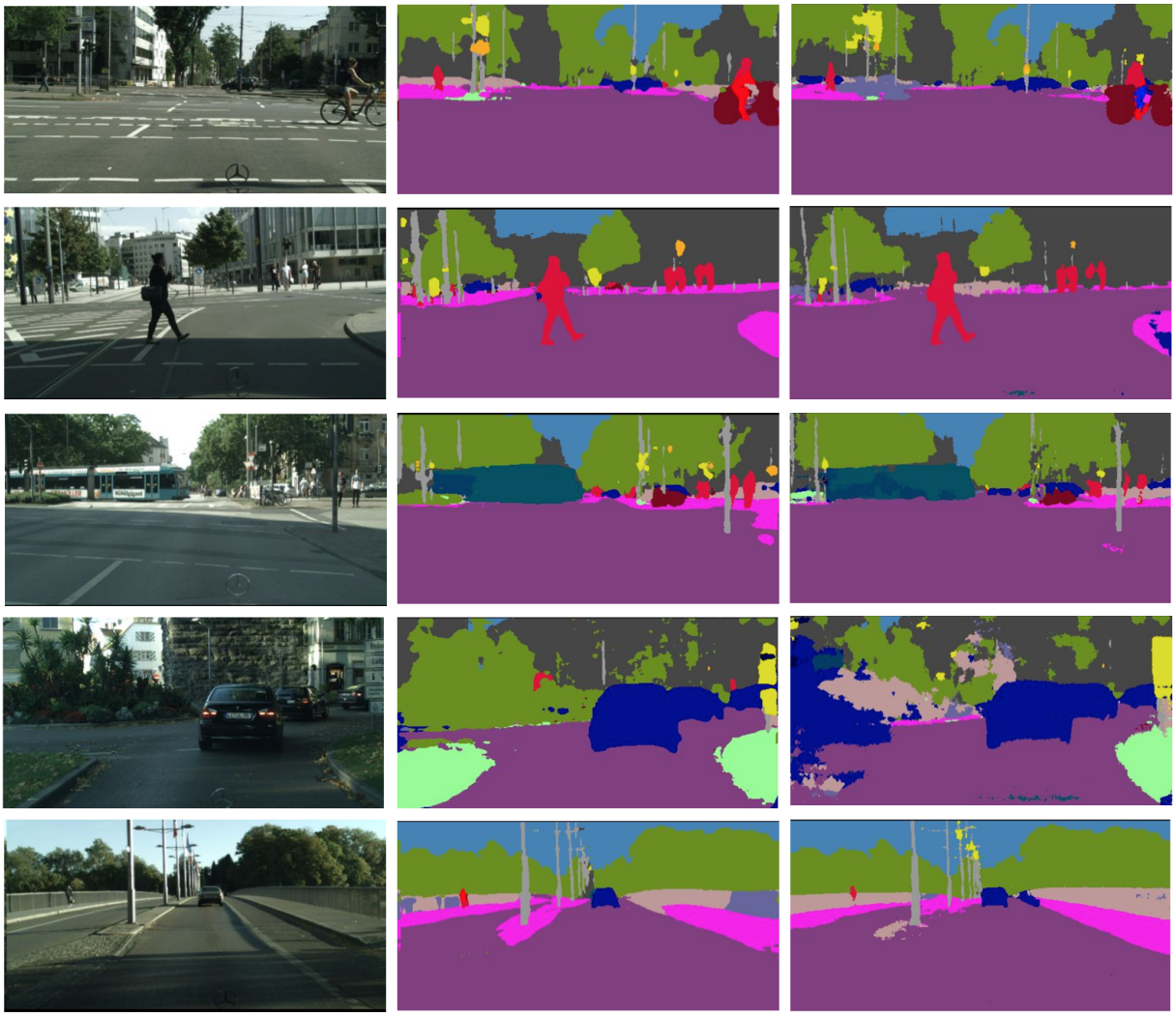

Figure 6.11: Qualitative results on Cityscapes for SegNet(on the right) and our model(on the left)

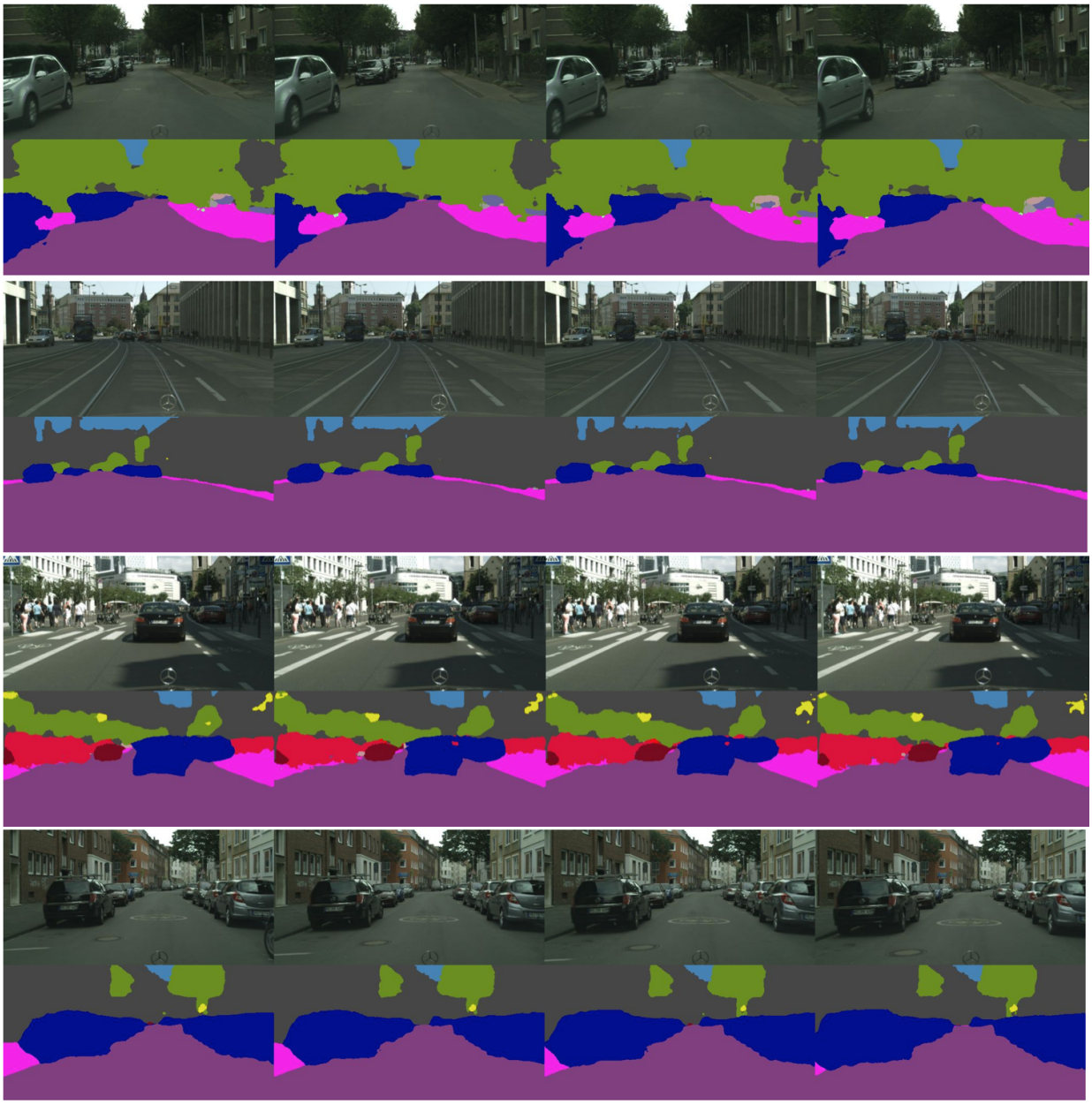

Figure 6.12: Qualitative results on Cityscapes for Predictive Semantic Segmentation, the rows with RGB images show the input to the system and the rows below each represent the predictive output for each input sequence

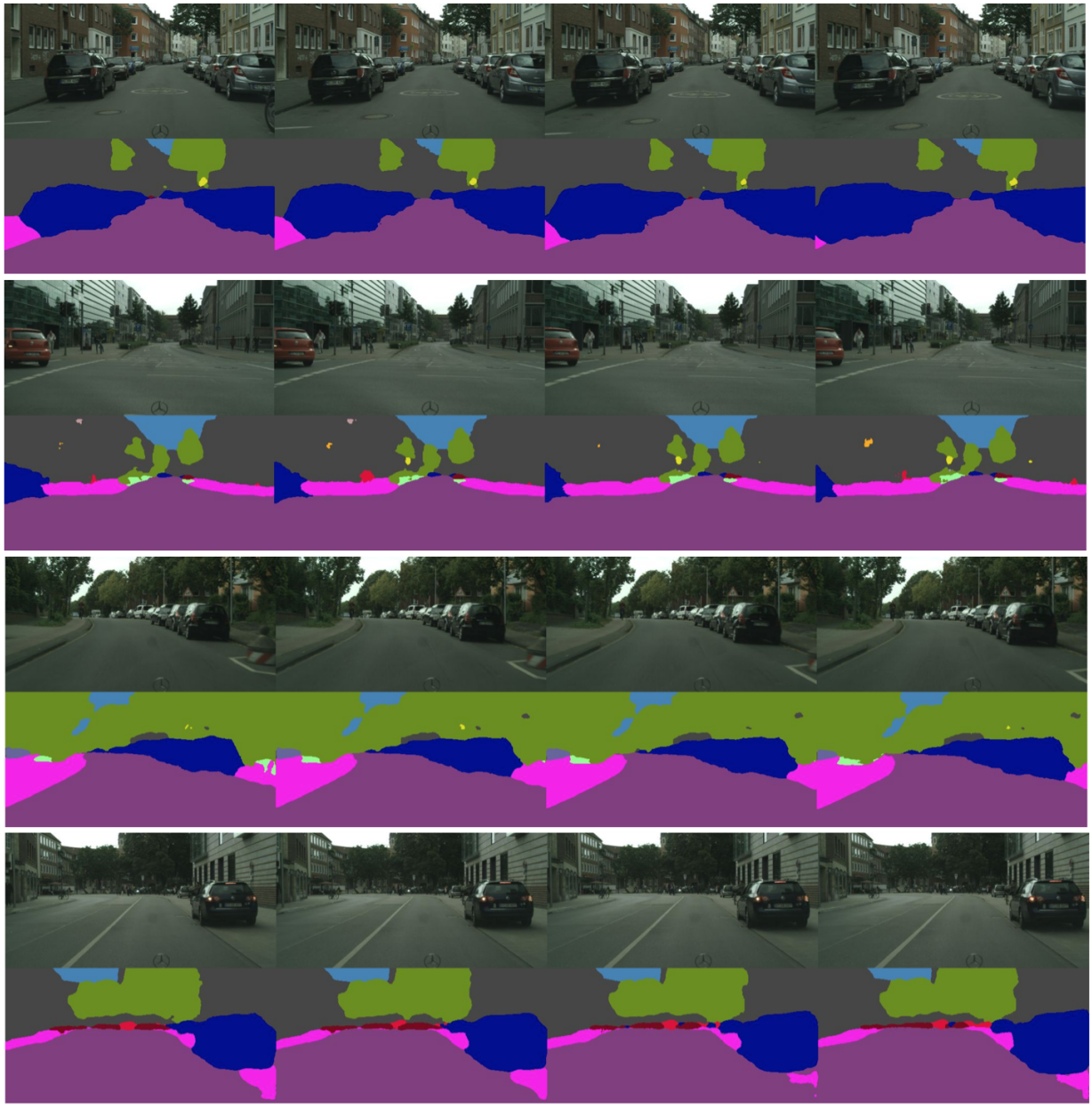

Figure 6.13: Qualitative results on Cityscapes for Predictive Semantic Segmentation, the rows with RGB images show the input to the system and the rows below each represent the predictive output for each input sequence

# Chapter 7

# Conclusion

In this chapter, we summarize the contributions of the work in this dissertation and discuss promising directions for future research.

## 7.1 Contributions

The focus of research in this thesis is on the problem of holistic scene understanding in videos. In particular, we proposed methods for accurate, efficient and temporally coherent semantic segmentation in videos. While typical approaches to these problems usually involve utilizing laborious annotations and significant amount of computing power, we are able to take advantage of the causality and data redundancy in videos to develop unsupervised approaches that effectively improve the segmentation performance over the baseline systems and give state-of-the-art results. We have also addressed the challenge of obtaining groundtruth for temporal data and presented methods to handle automatic groundtruth generation from existing annotated data. Furthermore, we presented algorithms that give temporally consistent and detailed semantic segmentation of video scenes and are able to predict the scene structure in near future. We see semantic segmentation not only as a goal in itself, but also as a tool for use in gaining a higher level of scene understanding which can be used in a variety of applications such as autonomous driving, robot navigation, 3D reconstruction and recognition, motion estimation, free-space estimation, etc. More concretely, the key contributions of this work are as follows:

**Unsupervised Spatio-Temporal Segmentation**. We proposed a novel method to incorpo-

rate time for segmentation. We improved on a family of methods that generalize graph based image segmentation to the spatio-temporal case by proposing a set of strategies that enable us to compute a dense graph based segmentation in a very computationally efficient way. We exploited the natural sparsity structure of the spatio-temporal graph, and employed a low rank approximation of its Laplacian closely related to the Nyström method. We only sampled 30-50% of the pixels in the video sequence to obtain dense segmentation for all the pixels and showed that the performance is comparable with a method that uses 100% of the pixels. Our method is simple, efficient and effective and does not require supervised learning or any complex post-processing. The input to our system is a sequence of frames, and the output is a dense segmentation of the frames into 3D super-pixels, coherent in time and space, with different levels of granularity.

**Context-aware Segmentation** We presented an algorithm to include high-level information from depth and deformable parts model for video segmentation and object discovery. We used a context aware aggregation strategy that employs an object detection system to find object-centric segmentations. This way, we are able to process the objects within the detected bounding-box, segment out the background and obtain accurate boundaries for them. We also experimented with various connectivity patterns in the construction of spatio-temporal graph and showed that a randomized strategy helps with the accuracy.

**Unsupervised Object Discovery, Detection, Tracking and Reconstruction** We proposed a unified and scalable framework for unsupervised dense object discovery, detection, tracking and reconstruction using RGBD cameras and a robot manipulator. The system simultaneously localizes a moving sensor and discovers a set of candidate shape and appearance models for multiple objects, including the background. This single framework performs dense simultaneous localization and mapping as well as unsupervised object discovery and detection for both the whole scene and each object within it. Spatio-temporal cues are used to produce a set of candidate objects. A motion based verification strategy is used to verify the candidate objects, and object appearance and shape is subsequently learned by grasping and poking.

**Generating Large Auxiliary Groundtruth Datasets** We proposed a simple and efficient

method to automatically propagate semantic labels from densely labeled frames to the rest of unlabeled video frames. Our simple method helps with reducing the cost of annotating videos and creates large-scale datasets very efficiently. We show that the auxiliary groundtruth in addition to the groundtruth can increase the accuracy of semantic segmentation techniques where there is enough diversity in the data and also the quality of generated groundtruth is generally good. We also developed a web-based video annotation tool for semantic segmentation that reduces the burden of hand-labeling.

**Temporal and Predictive Semantic Scene Understanding** We proposed a novel temporal pyramid network that utilizes multi-scale resolution feature maps of CNNs and is also capable of modeling temporal dependencies. We showed that our framework is suitable for the task of dense prediction and semantic segmentation. We also extended our network to the problem of prediction where the semantics of the scene are estimated for the future frames. We showed that using temporal variations along with spatial features in different scales helps to improve the performance of deep learning models. Our results show a significant improvement in performance over several strong baselines.

## 7.2 Future Work

We have presented video scene understanding algorithms, and have developed comprehensive datasets to evaluate future studies in this area. There were not many temporal scene understanding studies in literature before but the algorithms, and datasets developed in this thesis should help to accelerate research not only for video scene understanding but also to combine the above algorithms to solve other problems in compute vision. There are several possible directions for future work that can be made based on the material presented in this work:

**Selective Sampling:** We randomly sample the columns of graph Laplacian to perform the low-rank approximation in our spatio-temporal segmentation framework. However, selective sampling strategies generally decrease the approximation error for a given number of columns which means it is possible to increase the accuracy of our approach even more.

**Fast Eigen Decomopsition:** A major bottleneck in our super-pixel segmentation algorithm is the expensive eigen decomposition phase of spectral clustering. Even though we make the computation of eigenvectors orders of magnitude faster by using a further low-rank approximation of the Laplacian, it is still slow for the domain of video data. Therefore, improving this stage by designing algorithms that are able to find the eigenvectors efficiently would speed up our method considerably.

**Online Spectral Clustering:** Another possible way to reduce the complexity of our segmentation algorithm is to process the video incrementally and only consider a small number of frames at a time. This will drastically decrease the computation time. Hence, comping up with algorithms that can update the previously computed eigenvectors as we receive new frames would be extremely beneficial.

**Label Propagation:** The quality of our label propagation can be further improved by applying a post-processing step such as CRF. Also designing algorithms that can effectively learn from large noisy groundtruth where there is only small number of clean groundtruth is available would be very helpful.

**Datasets:** The number of annotated datasets for semantic segmentation is still relatively small compared to classification or categorization datasets. Creating larger and more diverse datasets should give results which are more consistent and accurate.

**Improve Training** One problem encountered while training semantic segmentation models is that over-fitting can quickly become an issue. It would be interesting to investigate what is causing this issue even when we have a large-scale noisy dataset with relatively good quality. One possible reason is that much of training data is not independent when dealing with sequential data, so lots of examples give no extra information. Developing techniques to find good training samples for such techniques could make learning much more efficient.

**Improved Detection algorithms:** Understanding the scene should help with improving the detection and recognition techniques. For example when a sliding window car detector scans the entire scene to detect cars, it might return falsely detected cars on the sky, or trees. We can reduce

these false positives effectively by introducing constraints on the structure of the semantic regions. Also semantic segmentation can help bounding box detection systems to segment out the detected object from the background with a fine and accurate boundary.

**Monocular Depth Estimation:** Existing monocular depth estimation methods either work on pixels or on super-pixels obtained from images. Since our spatio-temporal super-pixels are shown to preserve the object boundaries much better, we believe that our proposed method can be effectively used to estimate depth and surface normals in complex dynamics scene.

**Semantic Optical Flow:** Current optical flow estimation methods make generic, spatially homogeneous, assumptions about the structure of the flow in the scene. However, optical flow varies across an image depending on the semantic category of the objects. For example, the motion on roads is different than the motion of trees, cars or pedestrians. One interesting area of future research is to model the category based motions independently for the problem of flow estimation. Therefore, our temporal semantic segmentation method can be used to obtain an accurate and detailed segmentation of the scene that can be exploited in these semantically aware flow estimation techniques.

**Semantic 3D Mapping and Reconstruction** The temporal semantic scene segmentation can be used to guide the 3D reconstruction in videos by learning semantic class based priors. Having a semantically annotated dense 3D model gives a much richer representation of the scene than just the geometry and is useful for variety of robotic applications.

## Bibliography


[1] Alexey Abramov, Karl Pauwels, Jeremie Papon, Florentin Worgotter, and Babette Dellen. Depth-supported real-time video segmentation with the kinect. In Applications of Computer Vision (WACV), 2012 IEEE Workshop on, pages 457–464. IEEE, 2012.

[2] G. Alenyà, S. Foix, and C. Torras. Using tof and rgbd cameras for 3d robot perception and manipulation in human environments. Intelligent Service Robotics, 7(4):211–220, 2014.

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 33(5):898–916, 2011.

[4] Anurag Arnab, Sadeep Jayasumana, Shuai Zheng, and Philip HS Torr. Higher order conditional random fields in deep neural networks. In European Conference on Computer Vision, pages 524–540. Springer, 2016.

[5] Vijay Badrinarayanan, Ignas Budvytis, and Roberto Cipolla. Mixture of trees probabilistic graphical model for video segmentation. International journal of computer vision, 110(1):14–29, 2014.

[6] Vijay Badrinarayanan, Fabio Galasso, and Roberto Cipolla. Label propagation in video sequences. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 3265–3272. IEEE, 2010.

[7] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. arXiv preprint arXiv:1505.07293, 2015.

[8] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. International Journal of Computer Vision, 56(3):221–255, 2004.

[9] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. arXiv preprint arXiv:1511.06432, 2015.

[10] Sid Yingze Bao, Manmohan Chandraker, Yuanqing Lin, and Silvio Savarese. Dense object reconstruction with semantic priors. In Computer Vision and Pattern Recognition (CVPR), IEEE Conference on, pages 1264–1271, 2013.

[11] Charles Bibby and Ian Reid. Robust real-time visual tracking using pixel-wise posteriors. In Computer Vision–ECCV 2008, pages 831–844. 2008.

[12] Charles Bibby and Ian Reid. Real-time tracking of multiple occluding objects using level sets. In Computer Vision and Pattern Recognition (CVPR), IEEE Conference on, pages 1307–1314, 2010.

[13] Gérard Blais and Martin D. Levine. Registering multiview range data to create 3d computer objects. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 17(8):820–824, 1995.

[14] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Unsupervised feature learning for rgb-d based object recognition. In Experimental Robotics, pages 387–402. Springer, 2013.

[15] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. Pattern Recognition Letters, xx(x):xx–xx, 2008.

[16] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. Pattern Recognition Letters, 30(2):88–97, 2009.

[17] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, ECCV, pages 282–295. 2010.

[18] Ignas Budvytis, Vijay Badrinarayanan, and Roberto Cipolla. Label propagation in complex video sequences using semi-supervised learning. In BMVC, volume 2257, pages 2258–2259, 2010.

[19] Ignas Budvytis, Vijay Badrinarayanan, and Roberto Cipolla. Semi-supervised video segmentation using tree structured graphical models. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 2257–2264. IEEE, 2011.

[20] Wonmin Byeon and Thomas M Breuel. Supervised texture segmentation using 2d lstm networks. In 2014 IEEE International Conference on Image Processing (ICIP), pages 4373–4377. IEEE, 2014.

[21] Wonmin Byeon, Thomas M Breuel, Federico Raue, and Marcus Liwicki. Scene labeling with lstm recurrent neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3547–3555, 2015.

[22] Wonmin Byeon, Marcus Liwicki, and Thomas M Breuel. Texture classification using 2d lstm networks. In Pattern Recognition (ICPR), 2014 22nd International Conference on, pages 1144–1149. IEEE, 2014.

[23] J. Chang, D. Wei, and J. W. Fisher III. A video representation using temporal superpixels. In CVPR, 2013.

[24] Albert YC Chen and Jason J Corso. Propagating multi-class pixel labels throughout video frames. In Image Processing Workshop (WNYIPW), 2010 Western New York, pages 14–17. IEEE, 2010.

[25] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062, 2014.

[26] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv preprint arXiv:1606.00915, 2016.

[27] Xiaowu Chen, Qing Li, Yafei Song, Xin Jin, and Qinping Zhao. Supervised geodesic propagation for semantic label transfer. In European Conference on Computer Vision, pages 553–565. Springer, 2012.

[28] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. Image and vision computing, 10(3):145–155, 1992.

[29] Vincent Cheung, Brendan J Frey, and Nebojsa Jojic. Video epitomes. International Journal of Computer Vision, 76(2):141–152, 2008.

[30] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

[31] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. Journal of Software Engineering for Robotics, 2014.

[32] Alvaro Collet, Bo Xiong, Corina Gurau, Martial Hebert, and Siddhartha S Srinivasa. Exploiting domain knowledge for object discovery. In Robotics and Automation (ICRA), 2013 IEEE Int. Conf. on, pages 2118–2125, 2013.

[33] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3213–3223, 2016.

[34] Timothée Cour, Florence Bénézit, and Jianbo Shi. Spectral segmentation with multiscale graph decomposition. In In CVPR, pages 1124–1131, 2005.

[35] Timothee Cour, Florence Benezit, and Jianbo Shi. Spectral segmentation with multiscale graph decomposition. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 2, pages 1124–1131. IEEE, 2005.

[36] Daniel Cremers. Dynamical statistical shape priors for level set-based tracking. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 28(8):1262–1273, 2006.

[37] Daniel Cremers, Mikael Rousson, and Rachid Deriche. A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. International journal of computer vision, 72(2):195–215, 2007.

[38] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pages 303–312. ACM, 1996.

[39] Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In The IEEE International Conference on Computer Vision (ICCV), December 2015.

[40] Amaury Dame, Victor A Prisacariu, Carl Y Ren, and Ian Reid. Dense reconstruction using 3d object shape priors. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 1288–1295. IEEE, 2013.

[41] Daniel DeMenthon and Remi Megret. Spatio-temporal segmentation of video by hierarchical mean shift analysis. Technical Report UMIACS-TR-2002-68, University of Maryland, College Park, 2002.

[42] Charanpal Dhanjal, Romaric Gaudel, and Stéphan Clémençon. Efficient eigen-updating for spectral graph clustering. Neurocomputing, 131:440–452, 2014.

[43] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2625–2634, 2015.

[44] Felix Endres, Christian Plagemann, Cyrill Stachniss, and Wolfram Burgard. Unsupervised discovery of object classes from range data using latent dirichlet allocation. In Robotics: Science and Systems, volume 2. Seattle, Washington, 2009.

[45] Can Erdogan, Manohar Paluri, and Frank Dellaert. Planar segmentation of RGBD images using fast linear fitting and Markov chain Monte Carlo. In Computer and Robot Vision (CRV), pages 32–39, 2012.

[46] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2):303–338, 2010.

[47] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. IEEE transactions on pattern analysis and machine intelligence, 35(8):1915–1929, 2013.

[48] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 32(9):1627–1645, 2010.

[49] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. Int. J. Comput. Vision, 59(2):167–181, September 2004.

[50] Andrea Fossati, Juergen Gall, Helmut Grabner, Xiaofeng Ren, and Kurt Konolige. Consumer Depth Cameras for Computer Vision: Research Topics and Applications. Springer, 2012.

[51] C. Fowlkes, S. Belongie, Fan Chung, and J. Malik. Spectral grouping using the Nyström method. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26(2):214–225, Feb 2004.

[52] Fabio Galasso, Roberto Cipolla, and Bernt Schiele. Video segmentation with superpixels. In Computer Vision–ACCV 2012, pages 760–774. Springer, 2013.

[53] Mahsa Ghafarianzadeh, Matthew B. Blaschko, and Gabe Sibley. Unsupervised spatio-temporal segmentation with sparse spectral clustering. In BMVC, 2014.

[54] Mahsa Ghafarianzadeh, Matthew B Blaschko, and Gabe Sibley. Efficient, dense, object-based segmentation from rgbd video. In Robotics and Automation (ICRA), 2016 IEEE International Conference on, pages 2310–2317. IEEE, 2016.

[55] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[56] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In ICML, volume 14, pages 1764–1772, 2014.

[57] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. IEEE transactions on pattern analysis and machine intelligence, 31(5):855–868, 2009.

[58] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In Advances in neural information processing systems, pages 545–552, 2009.

[59] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph based video segmentation. IEEE CVPR, 2010.

[60] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph based video segmentation. IEEE CVPR, 2010.

[61] Erico Guizzo and Evan Ackerman. How rethink robotics built its new baxter robot worker. IEEE Spectrum, 2012.

[62] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In Computer Vision–ECCV 2014, pages 345–360. Springer, 2014.

[63] Swastik Gupta, Pablo Arbelaez, and Jagannath Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 564–571. IEEE, 2013.

[64] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 447–456, 2015.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[66] Evan Herbst, Peter Henry, Xiaofeng Ren, and Dieter Fox. Toward object discovery and modeling via 3-d scene comparison. In Robotics and Automation (ICRA), 2011 IEEE Int. Conf. on, pages 2623–2629, 2011.

[67] Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d object discovery via multi-scene analysis. In Intelligent Robots and Systems (IROS), IEEE/RSJ Int. Conf. on, pages 4850–4856, 2011.

[68] Steven Hickson, Stan Birchfield, Irfan Essa, and Henrik Christensen. Efficient hierarchical graph-based segmentation of rgbd videos. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 344–351. IEEE, 2014.

[69] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[70] Dirk Holz and Sven Behnke. Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. In Proceedings of the 12th International Conference on Intelligent Autonomous Systems (IAS), Jeju Island, Korea, June 2012.

[71] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-Time Plane Segmentation using RGB-D Cameras. In Proceedings of the 15th RoboCup International Symposium, volume 7416 of Lecture Notes in Computer Science, pages 307–317, Istanbul, Turkey, July 2011. Springer.

[72] Yuchi Huang, Qingshan Liu, and Dimitris Metaxas. ] video object segmentation by hypergraph cut. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 1738–1745. IEEE, 2009.

[73] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of The 32nd International Conference on Machine Learning, pages 448–456, 2015.

[74] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In Proceedings of the 24th annual ACM symposium on User interface software and technology, pages 559–568. ACM, 2011.

[75] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. IEEE transactions on pattern analysis and machine intelligence, 35(1):221–231, 2013.

[76] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia, pages 675–678. ACM, 2014.

[77] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22Nd ACM International Conference on Multimedia, MM '14, pages 675–678, New York, NY, USA, 2014. ACM.

[78] Feng Kang, Rong Jin, and Rahul Sukthankar. Correlated label propagation with application to multi-label learning. In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, volume 2, pages 1719–1726. IEEE, 2006.

[79] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3128–3137, 2015.

[80] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 1725–1732, 2014.

[81] Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. Adv. Neural Inf. Process. Syst, 2(3):4, 2011.

[82] Hema Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from rgb-d videos. IJRR, 32(8):951–970, 2013.

[83] Hema S Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Semantic labeling of 3d point clouds for indoor scenes. In Advances in Neural Information Processing Systems, pages 244–252, 2011.

[84] Ivan Krešo, Denis Čaušević, Josip Krapac, and Siniša Šegvić. Convolutional scale invariance for semantic segmentation. In German Conference on Pattern Recognition, pages 64–75. Springer, 2016.

[85] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[86] Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature space optimization for semantic video segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3168–3175, 2016.

[87] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In Robotics and Automation (ICRA), IEEE Int. Conf. on, pages 1817–1824, 2011.

[88] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Computer vision and pattern recognition, 2006 IEEE computer society conference on, volume 2, pages 2169–2178. IEEE, 2006.

[89] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10):1995, 1995.

[90] Alex Levinshtein, Cristian Sminchisescu, and Sven Dickinson. Spatiotemporal closure. In Computer Vision–ACCV 2010, pages 369–382. Springer, 2011.

[91] José Lezama, Karteek Alahari, Josef Sivic, and Ivan Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 3369–3376. IEEE, 2011.

[92] Chunming Li, Chenyang Xu, Changfeng Gui, and Martin D Fox. Distance regularized level set evolution and its application to image segmentation. Image Processing, IEEE Transactions on, 19(12):3243–3254, 2010.

[93] Mu Li, Xiao-Chen Lian, James T Kwok, and Bao-Liang Lu. Time and space efficient spectral clustering via column sampling. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 2297–2304. IEEE, 2011.

[94] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3367–3375, 2015.

[95] Guosheng Lin, Chunhua Shen, Ian Reid, et al. Efficient piecewise training of deep structured models for semantic segmentation. arXiv preprint arXiv:1504.01013, 2015.

[96] Ce Liu. Beyond pixels: exploring new representations and applications for motion analysis. PhD thesis, MIT, 2009.

[97] Ce Liu, Jenny Yuen, and Antonio Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 1972–1979. IEEE, 2009.

[98] Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T Freeman. Sift flow: Dense correspondence across different scenes. In European conference on computer vision, pages 28–42. Springer, 2008.

[99] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Semantic image segmentation via deep parsing network. In Proceedings of the IEEE International Conference on Computer Vision, pages 1377–1385, 2015.

[100] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3431–3440, 2015.

[101] David G Lowe. Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2):91–110, 2004.

[102] Aurelien Lucchi, Yunpeng Li, Xavier Boix, Kevin Smith, and Pascal Fua. Are spatial and global constraints really necessary for segmentation? In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 9–16. IEEE, 2011.

[103] Lu Ma, Mahsa Ghafarianzadeh, David Coleman, Nikolaus Correll, and Gabe Sibley. Simultaneous localization, mapping, and manipulation for unsupervised object discovery. In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pages 1344–1351. IEEE, 2015.

[104] Lu Ma and Gabe Sibley. Unsupervised dense object discovery, detection, tracking and reconstruction. In Computer Vision–ECCV 2014, pages 80–95. 2014.

[105] Julian Mason, Bhaskara Marthi, and Ronald Parr. Object disappearance for object discovery. In Intelligent Robots and Systems (IROS), IEEE/RSJ Int. Conf. on, pages 2836–2843, 2012.

[106] Colin McManus, Winston Churchill, Will Maddern, Alex Stewart, and Paul Newman. Shady dealings: Robust, long- term visual localisation using illumination invariance. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, May 2014.

[107] David Meger, Per-Erik Forssén, Kevin Lai, Scott Helmer, Sancho McCann, Tristram Southey, Matthew Baumann, James J Little, and David G Lowe. Curious george: An attentive semantic robot. Robotics and Autonomous Systems, 56(6):503–511, 2008.

[108] Maxime Meilland and Andrew I Comport. Super-resolution 3d tracking and mapping. In Robotics and Automation (ICRA), IEEE Int. Conf. on, pages 5717–5723, 2013.

[109] Hossein Mobahi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In Proceedings of the 26th Annual International Conference on Machine Learning, pages 737–744. ACM, 2009.

[110] Joseph Modayil and Benjamin Kuipers. Bootstrap learning for object discovery. In Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ Int. Conf. on, volume 1, pages 742–747, 2004.

[111] Benoit Morisset, Radu Bogdan Rusu, Aravind Sundaresan, Kris Hauser, Motilal Agrawal, Jean-Claude Latombe, and Michael Beetz. Leaving flatland: toward real-time 3d navigation. In Robotics and Automation (ICRA), IEEE Int. Conf. on, pages 3786–3793, 2009.

[112] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In ECCV, 2012.

[113] Richard A Newcombe and Andrew J Davison. Live dense reconstruction with a single moving camera. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 1498–1505, 2010.

[114] Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In Mixed and augmented reality (ISMAR), 10th IEEE Int. Symp. on, pages 127–136, 2011.

[115] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In Computer Vision (ICCV), IEEE Int. Conf. on, pages 2320–2327, 2011.

[116] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. In NIPS, volume 14, pages 849–856, 2001.

[117] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision, pages 1520–1528, 2015.

[118] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. IEEE transactions on pattern analysis and machine intelligence, 36(6):1187–1200, 2014.

[119] Abhijit S Ogale, Cornelia Fermuller, and Yiannis Aloimonos. Motion segmentation using occlusions. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 27(6):988–992, 2005.

[120] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors, 16(1):115, 2016.

[121] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In Proceedings of the IEEE International Conference on Computer Vision, pages 1742–1750, 2015.

[122] Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In Proceedings of the IEEE International Conference on Computer Vision, pages 1796–1804, 2015.

[123] Pedro HO Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In ICML, pages 82–90, 2014.

[124] Victor A Prisacariu and Ian D Reid. Pwp3d: Real-time segmentation and tracking of 3d objects. International journal of computer vision, 98(3):335–354, 2012.

[125] MarcAurelio Ranzato, Arthur Szlam, Joan Bruna, Michael Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: a baseline for generative models of natural videos. arXiv preprint arXiv:1412.6604, 2014.

[126] Shankar R. Rao, Roberto Tron, Rene Vidal, and Yi Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[127] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 779–788, 2016.

[128] Carl Yuheng Ren, Victor Prisacariu, David Murray, and Ian Reid. STAR3D: Simultaneous tracking and reconstruction of 3D objects using RGB-D data.

[129] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99, 2015.

[130] Xiaofeng Ren, Liefeng Bo, and Dieter Fox. Rgb-(d) scene labeling: Features and algorithms. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 2759–2766. IEEE, 2012.

[131] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In European Conference on Computer Vision, pages 102–118. Springer, 2016.

[132] Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Jonathan Balzer, Michael Zillich, and Markus Vincze. Towards scene understanding - object segmentation using rgbd-images. In Proc. of the 17th Computer Vision Winter Workshop (CVWW 2012), 2012.

[133] Bernardino Romera-Paredes and Philip HS Torr. Recurrent instance segmentation. arXiv preprint arXiv:1511.08250, 2015.

[134] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In Intelligent Robots and Systems (IROS), IEEE/RSJ Int. Conf.on, pages 1–6, 2009.

[135] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, pages 4580–4584. IEEE, 2015.

[136] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In Computer Vision and Pattern Recognition (CVPR), IEEE Conference on, pages 1352–1359, 2013.

[137] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell., 22(8):888–905, August 2000.

[138] Gabe Sibley, Nima Keivan, Alonso Patron-Perez, Liz Murphy, Steven Lovegrove, and Vincent Mamo. Scalable perception and planning based control. International Symposium on Robotics Research, 2013.

[139] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Advances in neural information processing systems, pages 568–576, 2014.

[140] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[141] Paul Smith, Tom Drummond, and Roberto Cipolla. Layered motion segmentation and depth ordering by tracking edges. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26(4):479–494, 2004.

[142] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. Convolutional-recursive deep learning for 3d object classification. In Advances in Neural Information Processing Systems, pages 665–673, 2012.

[143] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In Proceedings of the 28th international conference on machine learning (ICML-11), pages 129–136, 2011.

[144] Tristram Southey and James J Little. Object discovery through motion, appearance and shape. In AAAI Workshop on Cognitive Robotics, page 9, 2006.

[145] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. CoRR, abs/1502.04681, 2, 2015.

[146] Johannes Strom, Andrew Richardson, and Edwin Olson. Graph-based segmentation for colored 3D laser point clouds. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 2010.

[147] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In Proc. of the International Conference on Intelligent Robot Systems (IROS), Oct. 2012.

[148] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E Shi. Human action recognition using factorized spatio-temporal convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, pages 4597–4605, 2015.

[149] N. Sundaram and K. Keutzer. Long term video segmentation through pixel level spectral clustering on GPUs. In Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, Nov 2011.

[150] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pages 3104–3112, 2014.

[151] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.

[152] Richard Szeliski. Computer vision: algorithms and applications. Springer, 2010.

[153] C-H Teh and Roland T. Chin. On the detection of dominant points on digital curves. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 11(8):859–872, 1989.

[154] Rudolph Triebel, Jiwon Shin, and Roland Siegwart. Segmentation and unsupervised part-based discovery of repetitive objects. In Robotics: Science and Systems, volume 2, 2010.

[155] David Tsai, Matthew Flagg, Atsushi Nakazawa, and James M Rehg. Motion coherent tracking using multi-label mrf optimization. International journal of computer vision, 100(2):190–202, 2012.

[156] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3156–3164, 2015.

[157] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. International Journal of Computer Vision, 101(1):184–204, 2013.

[158] David Weikersdorfer, Alexander Schick, and Daniel Cremers. Depth-adaptive supervoxels for rgb-d video segmentation.

[159] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J Leonard, and John McDonald. Robust tracking for real-time dense rgb-d mapping with kintinuous. 2012.

[160] Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, Advances in Neural Information Processing Systems 13, pages 682–688. MIT Press, 2001.

[161] Fangting Xia, Peng Wang, Liang-Chieh Chen, and Alan L Yuille. Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net. In European Conference on Computer Vision, pages 648–663. Springer, 2016.

[162] Jiangjian Xiao and Mubarak Shah. Accurate motion layer segmentation and matting. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 2, pages 698–703. IEEE, 2005.

[163] Jiangjian Xiao and Mubarak Shah. Motion layer extraction in the presence of occlusion using graph cuts. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 27(10):1644–1659, 2005.

[164] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2691–2699, 2015.

[165] Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger. Semantic instance annotation of street scenes by 3d to 2d label transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3688–3697, 2016.

[166] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In Advances in Neural Information Processing Systems, pages 802–810, 2015.

[167] Chenliang Xu, Caiming Xiong, and Jason J Corso. Streaming hierarchical video segmentation. In European Conference on Computer Vision, pages 626–639. Springer, 2012.

[168] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In Proceedings of the IEEE international conference on computer vision, pages 4507–4515, 2015.

[169] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122, 2015.

[170] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4694–4702, 2015.

[171] Jenny Yuen, Bryan Russell, Ce Liu, and Antonio Torralba. Labelme video: Building a video database with human annotations. In Computer Vision, 2009 IEEE 12th International Conference on, pages 1451–1458. IEEE, 2009.

[172] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In NIPS, volume 17, page 16, 2004.

[173] Honghui Zhang, Jianxiong Xiao, and Long Quan. Supervised label transfer for semantic segmentation of street scenes. In European Conference on Computer Vision, pages 561–574. Springer, 2010.

[174] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In Proceedings of the IEEE International Conference on Computer Vision, pages 1529–1537, 2015.

[175] Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. Elastic fragments for dense scene reconstruction. environments, 27(16):7–35.

[176] Tianyi Zhou and Dacheng Tao. Shifted subspaces tracking on sparse outlier for motion segmentation. In Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, pages 1946–1952. AAAI Press, 2013.

[177] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.