Multi-dimensional feature and data mining

Georgiou, T.

2

Deep learning and traditional approaches for high dimensional data

Higher dimensional data such as video and 3D are the leading edge of multimedia retrieval and computer vision research. In this chapter, a comprehensive overview of the state of the art of higher dimensional features from deep learning and also traditional approaches is given. Moreover, key insights into current research areas and challenges that arise are discussed. Current approaches are frequently using 3D information from the sensor or are using 3D in modeling and understanding the 3D world. With the growth of prevalent application areas such as 3D games, self-driving automobiles, health monitoring and sports activity training, a wide variety of new sensors have allowed researchers to develop feature description models beyond 2D. Although higher dimensional data enhance the performance of methods on numerous tasks, they can also introduce new challenges and problems. The higher dimensionality of the data often leads to more complicated structures which present additional problems in both extracting meaningful content and in adapting it for current machine learning algorithms. Due to the major importance of the evaluation process, we also present an overview of the current datasets and benchmarks. Finally, we make remarks on potential future directions, some of which are tackled in this thesis.

2.1 Introduction

With the current growth of computing systems and technologies, three and four dimensional data, such as 3D images and videos, are becoming a commodity in multimedia systems. Understanding and utilizing this data is the leading edge of modern computer vision. In this chapter we present a comprehensive study (including a categorization) of these high dimensional data types, as well as the methods developed to process them, accompanied with their strengths and weaknesses. Finally, we collect and give an overview of the main areas that utilize such representations.

One of the first steps towards developing, testing and applying methods on high dimensional data is the acquisition of complicated datasets, for instance datasets consisting of 3D models [419, 45], three dimensional medical images and videos (MRI, Ultrasound etc.) [54, 143], large 2D and 3D video datasets for action recognition [235, 324] and more. Different datasets are used for different data mining tasks. For example, object retrieval, movie retrieval and action classification tasks are performed on video data such as movies, YouTube clips et cetera. Clustering and classification tasks are performed on medical images for computer aided diagnostics and surgery. Object classification and detection, as well as scene semantic segmentation are usually applied on RGB-D images and videos retrieved by sensors such as the Microsoft Kinect [444].

We perform two types of categorization. The first is dataset and application driven and the second is method driven. Although these datasets find applications in different fields there are some similarities between the methods used. For example, deep learning techniques are used for 2.5D and 3D object classification (either retrieved from depth maps or designed models), action classification, video retrieval as well as medical applications, for instance landmark detection and tracing in ultrasound video. Histograms of different metrics (e.g. gradients, optical flow or surface normals) are used as features that describe the content of the data.

One of the recent breakthroughs has been the development of new deep learning architectures which could overcome (to some extent) the well known vanishing gradient problem in training. In the case of neural networks, they changed the landscape from typically using a few layers to using hundreds of layers. These methods typically learn the features based on large datasets directly from the raw data and have the least supervision. The other main approach from the literature is the continuation of advances in traditional or ”hand crafted” and ”shallow learning” based features. Features extracted from 2D optical information, e.g. natural images, in computer vision have had a major impact in computer vision and human-computer interaction across many applications [247, 225, 19, 373, 322, 299, 7, 447] and many of the higher dimensional methods were inspired or adapted from the 2D versions. These

approaches usually require significantly more supervision but also can be effective when large training datasets are not accessible.

High dimensional computer vision, with the definition given in this thesis (i.e. higher than 2D), is a very broad field that contains many different research areas, data types and methods. There have been surveys on specific areas within high dimensional computer vision. For example, when it comes to the static world, some of surveys focus on specific research areas such as 3D object detection [112, 306], semantic segmentation [111, 432, 96], object retrieval [79, 362] or Human Action Recognition [274, 168, 132]. Others focus on methodologies such as interest point detectors and descriptors [36, 378, 195], spatio-temporal salient point detectors and descriptors [210] or deep learning [152]. Finally some surveys focus on datasets and benchmarks of a specific research area, such as human action recognition [124]. This chapter differs from these since the focus is on the generalization of methodologies with the increase of dimensionality, regardless of the research area or the type of data. The most relevant work to this was done by Ioannidou et al. [152] where they focus on computer vision on static 3D data. There are two main differences with this work, (i) they focus only on deep learning methods and (ii) they focus only on 3D representation of the static world which means that they neglect the temporal dimension, which is a significant focus of this chapter.

2.2 Deep learning

Deep learning techniques refer to a cluster of machine learning methods that construct a multi-layered representation of the input data. The transformation of the data in each layer is typically trained through algorithms similar to back-propagation. There are several deep learning methods. In this section we will give a summary of the methods that have been used with high dimensional data. The main examples are the Convolutional Neural Networks (CNNs), the Recurrent Neural Networks (RNNs), Auto-Encoders (AE) and Restricted Boltzmann Machines (RBMs). For a detailed overview of deep learning in computer vision the reader is referred to [110] and for a general deep learning overview to [102].

Deep learning approaches can be split into two main categories, supervised and unsupervised methods. Supervised methods define an error function which depends on the task the method needs to solve and change the model parameters according to that error function. These kind of methods provide an end-to-end learning scheme, meaning that the model is learning to perform the task from the raw data. Unsupervised methods usually define an error function to be minimized which depends on the reconstruction ability of the model. Together with the reconstruction error, depend-

ing on the method, an auxiliary error function might be defined which forces some characteristics to the learned representation. For example sparse auto-encoders try to force the learned representation to be sparse, which helps the overall learning procedure and provides a more discriminative representation. The most commonly used deep learning method is Convolutional Neural Networks (CNNs). In the rest of this section we give a brief introduction to the basic deep learning methods and provide an in depth analysis on their generalization from the image domain to the higher dimensional domains.

2.2.1 Basic deep learning methods

2.2.1.1 Convolutional neural networks (CNN)

Convolutional Neural Networks (CNNs) consist of multiple layers of convolutions, pooling layers and activation functions. Usually each layer will have a number of different convolutional kernels, a non-linear activation function and, maybe, a pooling mechanism to lower the dimensionality of the output data. An example of such a layer is shown in Figure 2.1. These networks were initially applied on handwritten digit recognition [199] but got the attention they have today after the introduction of LeNet [200] and more so after Krizhevsky et al.'s [184] work in 2012, where they won the ImageNet 2012 image classification competition with a deep-CNN. This recent success of CNNs highly depends on the increased processing power of modern GPUs as well as the availability of large scale and diverse datasets which made training models with millions of trainable parameters possible.

One of the main drawbacks of deep convolutional neural networks is that they tend to overfit the data. Moreover, they suffer from vanishing and exploding gradients. Resolving these issues have motivated a lot of research in various directions. More specifically, different elements of CNNs are studied and proposed, e.g. activation functions or normalization layers, training strategies and the generic network architecture, for example the inception networks [358]. Most of this research is based on image recognition as the established benchmark due to the availability of large scale annotated datasets such as the ImageNet [302] and the Microsoft COCO [217]. Nonetheless, many of these methods have been generalized and adapted to be applicable to 2.5D and 3D data, such as videos, and RGB-D images.

2.2.1.1.1 Activation functions. One of the main components of the successful AlexNet [184] on the ImageNet 2012 challenge is the Rectified Linear Unit [159, 253] activation function. The output of the function is $\max(0, y)$, where y is the output of a node in the network. The main advantages of this layer is the sparsity it provides to

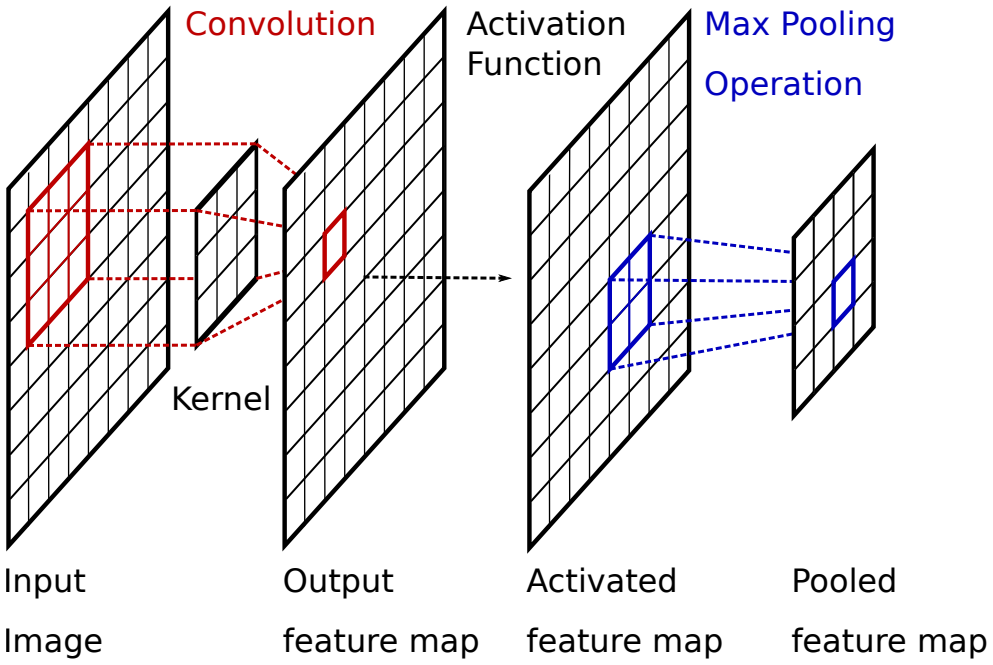

Figure 2.1: Basic CNN block. A single layer is shown which applies a kernel on an input filter followed by an activation function and a max-pooling operation.

the output as well as reduction of the vanishing gradients problem, compared to the more traditional hyperbolic tan and the sigmoid functions [101].

In the past years, many researchers have proposed new activation functions in order to improve the quality of neural networks. Some examples are the leaky ReLU (LReLU) [230], which instead of having always zero as output of negative inputs has a small response proportional to the input, i.e. $\alpha * y$, the Parametric Rectified Linear Unit (PReLU) [126], which learns the parameter α of LReLU, the Exponential Linear Unit (ELU) [53] and its trainable counterpart Parametric ELU (PELU) [381], and many more [5, 103, 163, 176]. For a more detailed overview of activation functions the reader is referred to [381].

2.2.1.1.2 Normalization. Experimental results suggest that when networks have normalized inputs, with zero mean and standard deviation of one, they tend to converge much faster [184]. In order to take advantage of this finding it is a common practice to rescale and normalize the input images [184, 338, 146]. Besides the input normalization, many researchers try to also normalize the input of individual layers,

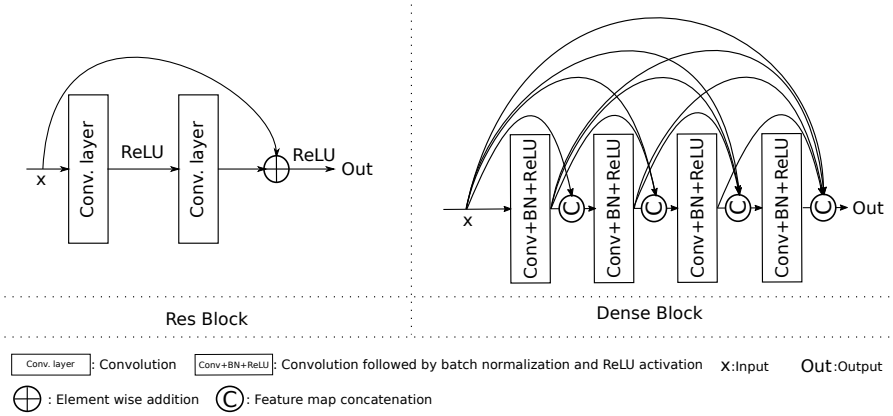

Figure 2.2: On the left is the ResBlock, the building block of ResNet [127]. After two convolution operations the input is added to the output in order to produce the residual learning function $H(x) = F(x) + x$. On the right is the building block of Dense Net [146]. The layer l gets as an input the output of all layers $[l-4, l-1]$.

in order to alleviate the covariate shift affect [331]. The traditional method of activation normalization is the Local Response Normalization [159, 184]. The most established work though is the later batch normalization technique [153]. In this work the output of each layer is rescaled and centered according to the batch-statistics of activations. The success of this method gave rise to more research in this direction like [12, 382, 311, 418, 394, 150]. For a detailed overview and comparison of these methods the reader is referred to [287, 418, 394].

2.2.1.1.3 Network structure. In an attempt to increase their performance, a large group of works have also explored different architectures of the internal structure of CNNs. After the work of Krizhevsky et al. [184], researchers tried to understand how different parameters effected the quality of the networks. Here we will give a small overview of the main milestone works since then.

One of the first important works was the one of Simonyan and Zisserman [338] who proposed the VGG nets. In their work they showed that with small convolutional kernels (3x3), deeper networks could be trained. They introduced 11, 13, 16 and 19 weighted layered networks. One main constraint on the possible depth of neural networks is the vanishing gradients problem. In an attempt to alleviate this issue, HighWay networks [349] and residual networks (ResNet) [127] make use of "skip" or "shortcut" connections in order to pass information from one layer to one or several layers ahead (Figure 2.2). Huang et al. [146] generalized this idea even further, with

their DenseNet, by giving as input to the l-th layer all previous $l-1$ layers. The building blocks of ResNet, Res Block, and DenseNet, Dense Block, are shown in Figure 2.2.

Besides skip connections, which helped deeper networks to be trained, different methods to increase the quality of networks have also been studied. Lin et al. [216] proposed the Network in Network (NiN) architecture. In their work they substituted the linear convolutional nodes with small Multi-Layer Perceptrons (MLP), giving to the network the ability to learn non linear mappings in a layer. Lee et al. [203] proposed the Deeply Supervised Nets (DSN) which use secondary supervision signals directly to hidden layers of the network. Liu et al. [222] explore a different approach, where the final decision, either classification or any other task, is made not only by the information in the last layer but also from deeper layers. They do so with their Convolutional Fusion Network (CFN), in which Locally Connected (LC) layers are used to fuse lower level information from deeper layers with the high level information of the top layer and make a more informative decision.

2.2.1.2 Recurrent neural networks (RNN)

Recurrent neural networks are a special class of artificial neural networks. A basic RNN module is composed by a feed forward node computing a "hidden state", a recurrent connection, which connects the hidden unit to the next time step input, and an output unit, as seen in Figure 2.3 . This recurrent connection gives the network the ability to make predictions not only according to the current input but also historic inputs that comprise a sequence of data.

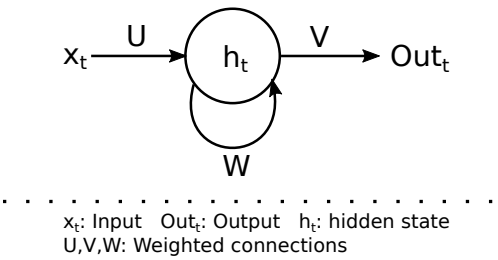

Figure 2.3: Basic module of an RNN processing time step t.

Although this architecture was successful, in problems with a large number of time steps it could no longer maintain high performance. That happens due to the vanishing gradient problem in back propagation through time (BPTT), a most widely used training procedure of RNN. In order to counter these limitations a new architecture, the long short-term memory node (LSTM), was proposed by Hochreiter and Schmidhuber [141]. It contains several gates that control the flow of information and allow the network to store long term information, if needed. Such an architecture has been used for many tasks that deal with sequential data, such as language modeling [439]

and translation [228], action classification in videos [70], speech synthesis [84] and more.

Inspired from the success of the LSTM method, researchers proposed many variations. Some are generic, and can be applied to any problem that simple LSTM is applied while others are application specific.

To the best of our knowledge, the first generic extension of LSTM was proposed in the work of Gers et al. [99]. They noticed that none of the gates have direct connections to the memory cell they are supposed to control. In order to alleviate that limitation they proposed "peephole" connections from the memory cell to the input of each gate. Cho et al. [51] proposed an extension, the Gated Recurrent Unit (GRU), that simplified the architecture and reduced the number of trainable parameters by combining the forget and input gates. Laurent et al. [198] and Cooijmans et al. [56] proposed batch normalized LSTM. Although [198] batch normalized only the input of the node, Cooijmans et al. [56] did so also in the hidden unit. Zhao et al. [446] proposed a combination of several of the above extensions. Specifically, they proposed a bidirectional [319] GRU unit, combined with batch normalization. They fed the network with human joints and action labels for 3D video action recognition. For a more thorough review regarding LSTM and its variants, the reader is referred to [106].

As mentioned above, some extensions of the LSTM are application specific. For example, Shahroudy et al. [324] proposed the Part-Aware LSTM (PA-LSTM), an architecture tailored for skeleton based data. Instead of having one memory cell for the whole skeleton, as is a common approach, they introduced one memory cell per joint of the skeleton, each with its own input, forget and output gates. Liu et al. [219] proposed the spatio-temporal LSTM unit with trust gates (ST-LSTM) for 3D human action recognition. This unit extends the recurrent learning with memory to the spatial domain as well.

2.2.1.3 Restricted Boltzmann machine (RBM)

The Restricted Boltzmann Machine (RBM) was first introduced by Hinton [140] in 1986. It is a two-layer, undirected, bipartite and undirected model (Figure 2.4). It comprises of a set of visible units, which are either binary or real valued, and a set of binary hidden nodes. A configuration with visible vector **v** and hidden vector **h** is assigned an energy given by:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i \in visible} \alpha_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{ij} v_i h_i w_{ij}, \tag{2.1}$$

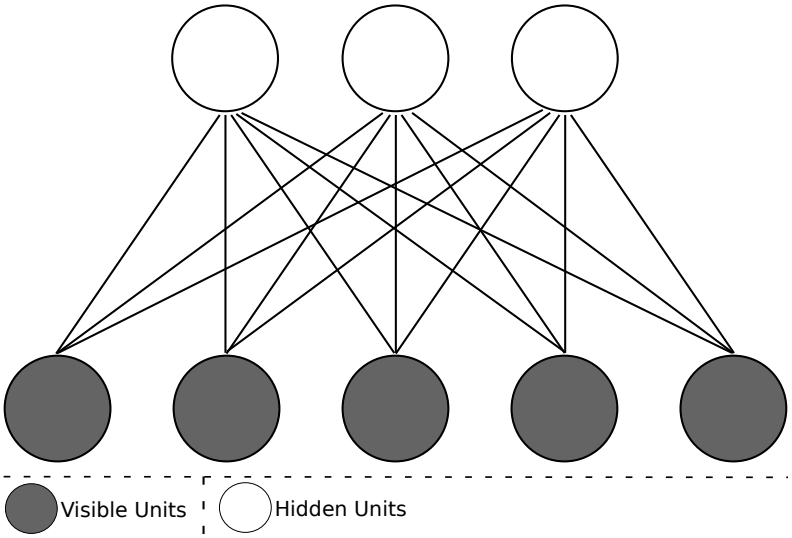

Figure 2.4: RBM architecture. Notice that the connections are undirected.

where α_i, b_j, w_{ij} are the network parameters. Given this energy the network assigns to every pair ($\mathbf{v}$, $\mathbf{h}$) a probability:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} \tag{2.2}$$

where Z is the partition function and is given by summing over all possible pairs of visible and hidden vectors. Since there are no direct connections between the hidden or visible units, we can easily obtain an unbiased pair ($\mathbf{v}$, $\mathbf{h}$). Given the visible vector $\mathbf{v}$ the hidden unit h_j is assigned to one with probability:

$$P(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_i v_i w_{ij}), \tag{2.3}$$

where $\sigma(\cdot)$ is the logistic sigmoid function. Similarly given a hidden vector $\mathbf{h}$ the probability of a visible unit v_i to be assigned to one is given by:

$$P(v_i = 1|\mathbf{h}) = \sigma(\alpha_i + \sum_j h_j w_{ij}), \tag{2.4}$$

Starting from the training data, the network parameters are tuned in order to maximize the likelihood of the visible and hidden vectors pair $(\mathbf{v}, \mathbf{h})$.

RBMs are only two layer deep models and thus are restricted in the complexity of the data they can represent. In order to alleviate this issue a number of deeper models

built on RBMs are designed. The most well known models derived from RBMs are the Deep Belief Networks (DBN) [138], Deep Boltzmann Machines (DBM) [310] and the Deep Energy Models (DEM) [255]. They are all multi-layer probabilistic models that perform non-linear transformation to the data.

DBNs are trained in a greedy layer wise manner, where each layer is trained as an RBM. After the training is done the weights are fixed and the next layer is trained. The final model keeps only the top-down connections of the layers except the top two that remain undirected. Finally, in order to avoid the poor local minimum of the greedy learning, the weights are fine tuned with the up-down algorithm. In case of classification the labels are given to the last layer as binary input. Unlike DBNs, DBMs have undirected weights in all layers. Initially the weights are also trained in a greedy fashion, like a DBN. Since it is very computationally expensive to estimate and maximize the likelihood directly, Salakhutdinov and Larochelle [310] proposed an approximative algorithm which maximizes the lower bound of the log-likelihood [308, 309]. Finally, DEM, the most recent deep model based on RBMs, is a fully connected feed forward network with an RBM on top [255]. The non-stochastic nature of the hidden layers renders it possible to have an efficient training of the whole model simultaneously. For a more comprehensive review of these models the reader is referred to [110].

2.2.1.4 Auto-encoders (AE)

Auto-encoders are a collection of neural network methods based on unsupervised learning. They were first introduced by Bourlard and Kamp [31] in 1988, as auto-association networks. The main idea is to reduce the dimensionality of the data with a fully connected layer and then try to recover the input from the reduced representation. In the case where the network is able to reconstruct the input, the intermediate low-dimensional representation should contain most of the information of the original data (Figure 2.5). Since a single layer network is able to perform only linear transformations, it is not sufficient for performing high dimensionality reduction of complicated data. Thus Hinton and Salakhutdinov [139] proposed a multiple layer version, called auto-encoder (AE). It utilizes several layers to transform or "encode" the data. In some cases, if there is large error in the first layers, these models only learn the average of the training data. In order to alleviate this issue, [139] proposed to pre-train the network so the initial parameters are already close to a good solution. Since then many variants of AEs have been proposed.

One of the first variations of AEs is the Sparse auto-encoder. The basic idea behind it is to transform the data on an over-complete representation of higher dimensionality than the original. The benefits of such a transformation is that (i) there is a high

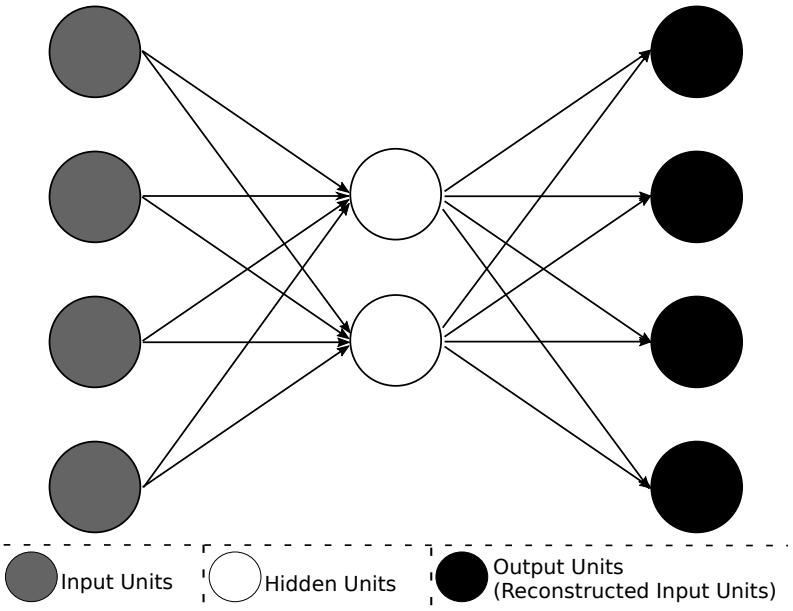

Figure 2.5: Auto-association network. Notice that the output units are reconstructed input units.

probability that in the new representation the data will be linearly separable and (ii) it can provide a simple interpretation of the input data in terms of a small number of parts by extracting the structure hidden in the data [276].

Vincent et al. [386, 387] suggested that a good transformation should provide similar representation for two similar data points. In an effort to force the model to be more robust in small variations of the data they proposed the Denoising AE (DAE), which tried to reconstruct the original data given slightly modified data as input. Rifai et al. [291] proposed a different method to achieve robustness to small input variations, the Contractive AE. They do so by penalizing the sensitivity of encoded representation with respect to the input data point.

Masci et al. [236], inspired by the success of CNNs, proposed a combination of AE with CNNs the Convolutional AE (CAE), and applied it on image datasets, MNITST and CIFAR10. The architecture comprises of several stacked convolutional layers. The model is used as a pre-train mechanism for a CNN which is then trained in a supervised manner for object classification.

2.2.2 Deep learning for high dimensional data

In this section we describe the main deep learning approaches applied on high dimensional data and provide a categorization of them. Specifically, we cluster the methods according to the type of generalization performed.

Most of the deep learning methods applied on higher than two dimensional data are generalized from lower dimensional counterparts, e.g. CNNs, CAEs, etc.. The methods can be divided into two categories, namely increase of physical dimensions and increase of modalities. There are also several models that are developed for high dimensional data and were not generalized from lower dimensions, such as the PointNet [279]. It is important to note that all of the deep learning methods developed for 2D (images) and the generalization to 3D as well are either CNNs or a variation of them, like CAE.

2.2.2.1 Increase of physical dimensions

In this section we describe the methods that were based on generalizing an existing approach to higher dimensions. Although this seems straightforward, due to the curse of dimensionality, as well as the large demand of memory and computational power of deep learning approaches, the extension from two to three dimensional data is not trivial. When considering the static world, i.e., time is not involved in some way, two main concepts exist: the straightforward extension to three dimensional kernels and the projection of data to fewer dimensions coupled with the use of an assembly of lower dimensional models, usually pre-trained on a large dataset, like the ImageNet 2012 [302].

The first approach to extend the 2D convolutional deep learning techniques to the 3D case is the work of Chang et al. [45] on ShapeNets. They implemented a convolutional DBN with three dimensional kernels with which they learned a 3D shape representation from CAD models. The three dimensional convolutional kernels (and pooling) have also been combined with other models, such as the feed forward CNNs [323], CAEs [35] and GANs [417]. Moreover, they have been utilized in many fields such as 3D medical Images [69], Computational Fluid Dynamics (CFD) Simulations [371], 3D objects [240] and Videos [160]. The main drawback of these approaches is the high computational and memory demand of the resulting models, which limit both their size and the input resolution they can support. Although this is the case they are able to exploit relationships in all three dimensions, unlike the 2D methods.

The second cluster is the reduction of the data dimensionality to two, in order to be able to construct complicated models as well as take advantage of pre-trained ones. The reduction from three to two dimensions depends on the type of data in question. For example, when CAD models or 3D objects are concerned, the projection

to two dimensions is done from an outside perspective, i.e. “taking photos” of the object from different angles [353]. Shi et al. [327] proposed an alternative representation of the 3D models. Specifically they proposed a projection of the 3D shape on a cylinder around the object. The height of the cylinder is equal to the height of the object, making their representation invariant to scaling. Three dimensional medical images contain information in three dimensional space, in which case the outside perspective misses all information relevant to most applications. Thus, the data are not projected but rather processed in a slice-by-slice manner [69]. In the case of videos three strategies for lowering the dimensionality have been proposed. In the first one each frame can be considered separately [70, 380]. The second considers frames as extra channels [87, 337, 169, 403]. This is usually done when passing to the networks the optical flow for several frames. Another approach is to try and compress the information of several frames into one. The work of Bilen et al. [23] is in that direction. They propose the Dynamic Image. More specifically, they adapt the method of Fernando et al. [88] that combines features from multiple frames to the pixel level. The result is an image which contains movement information, similar to a blurred one.

Due to the lower dimensionality of the transformed input data, it is possible to construct very complicated and large models. Moreover, a common approach is to use and fine-tune pre-trained models on very large and diverse datasets such as the ImageNet 2012 [302]. Although this is the case, as mentioned in the previous section, these methods lose the ability to explore the correlations in the data in all available dimensions.

2.2.2.2 Increase of modalities

The second type of generalization refers to the increase of the available modalities of the data. To be more precise, although the physical dimensions of the data remain the same, for example from 2D image to 2D image or 2D+time to 2D+time, the information given per point increases. Some examples are the RGB-D data, optical flow added to the videos and more. Depending on the nature of the extra information, the resulting representation might result in a partial space-dimensionality increase. For example, the RGB-D data do not increase the dimensions to three. Nonetheless, the extra information is the distance to the sensor, which provides some information about the extra third physical dimension.

When dealing with this type of dimensionality increase, researchers proposed various strategies to incorporate the extra information. In this work we identified four different strategies.

The most simple and naive approach is to consider the extra information as an

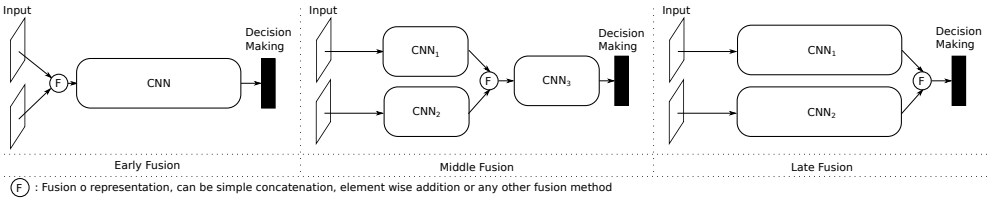

Figure 2.6: Three naive approaches of fusing information from different modalities. Left shows the early fusion, which fuses before any processing. Middle shows mid-fusion and on the right is the late fusion approach.

extra channel and process with the same data dimensionality as before. This is very common when dealing with RGB-D data [58, 391].

In the second category belong approaches that process the different types of information separately and fuse the extracted features by concatenating the feature maps [224, 117]. The extreme case that the fusion happens before any processing layers is the aforementioned first category. Some methods fuse the representations in a mid-stage [371, 43, 169] and some in a late stage [337, 87, 403], as shown in Figure 2.6.

In the third category belong methods that do not apply a naive fusion of the different representations, such as concatenation. Many works propose more sophisticated strategies for fusing the different modalities. For example, Wang et al. [401] try to specifically learn modality specific and common features during training. As a result the total complexity of the model reduces. Moreover, one modality might be missing some of the common features due to noise, such as occlusion, clutter or illumination. In such a case the quality of the representation will not drop since the other modality will provide the necessary information. The sharing of the features happen between the convolutional and deconvolutional layers with fully connected layers. In order to force the networks to learn common and modality unique features, the similarity and dissimilarity of the features, respectively, is added to the global loss function. The similarity and dissimilarity of the features is measured using multiple kernel maximum mean discrepancy (MK-MMD) [401]. Another example is the work of Hazirbas et al. [125], where they make the assumption that one of the modalities is the main source of information and the rest are complementary. They assign one CNN to each modality and then, at several levels of the CNN's hierarchy they insert information from the complementary branches to the main one. Deng et al. [64] followed a different approach. Instead of having two streams, they introduced a third stream, the interaction stream, which is comprised be their newly defined GFU unit. By using this interaction stream, the feature maps of all streams are updated at the interaction points. Park

et al. [271] proposed the multi-modal feature fusion module in order to combine information from different modality-specific branches. Valada et al. [383] proposed a fusion module (SSMA) that emphasizes areas and modality specific feature maps according to the feature map contents, and thus leveraging common and modality specific features.

Finally, some researchers defined data specific solutions. For example, the work of Georgiou et al. [371] evaluates three different modality-processing strategies specific for CFD simulation output, which consist of 4 different modalities over 6 channels of information. Gupta et al. [117] propose a data transformation for the depth channel in RGB-D data, called HHA. Mainly, they introduced two more channels. Although the values of those channels are computed from the depth map itself, they are transformations that are easily learnable, by convolutional kernels, namely height from ground and surface angle to gravity vector.

The benefits of using this transformation are two-fold. First, the network gets more relative information to its input, and second with the depth information transformed to a three channel representation it is possible to use pre-trained networks on ImageNet for this modality as well. Eitel et al. [78] proposed three more encodings that transfer the depth data to a three channel representation and compared them to each other and HHA. Their intuition was that since in object classification, all objects have similar elevation not all channels of HHA are interesting. The projections they proposed are (i) copy the depth values to all channels, (ii) transform to the surface normal vector field and (iii) apply jet colormap of depth values to RGB, ranging from red (near), through green to blue (far). They argue that since the networks are pre-trained on RGB data, transforming depth to RGB might result in a more stable fine tuning of the networks. The last method showed the best results on object classification. Nonetheless, they do not perform a comparison in the case where the elevation makes a difference and thus there is no objective comparison between their method and HHA. For a visual comparison of the four different schemes, the reader is referred to [78].

2.3 Traditional methods

Traditional methods vary a lot depending on the application and the type of data they are applied on. For example, when dealing with semantic segmentation the most common, non-deep, approach is to apply a graph model like a Conditional Random Field (CRF) [334, 174, 65, 352]. On the other hand a large group of works utilize template matching approaches [135, 137, 113, 292] in order to tackle object detection. Although there is a large diversity on the applied methods, there are some

common practices between most of them. The data are not processed in their raw format, but they are transfered in a feature space in which they are represented and then processed by any machine learning pipeline.

Building from the very successful work of feature representation of images in many applications of computer vision, a lot of methods are developed that generalize them to be applicable to higher dimensional data as well. The main idea is to describe the content of an image using a number of points or neighborhoods instead of the whole image. The type of description can vary, from raw values to histograms of gradients and point wise comparisons. In order to get a good content description and not background description, researchers develop specialized detectors which detect points according to several characteristics. This very well known pipeline is extended and applied to higher dimensional data.

The most common types of higher dimensional data that people are dealing with are objects represented by surfaces and/or color, volumetric representation of the world, videos or sequences of images, or in the extreme scenario four dimensional data, a three dimensional representation evolving in time. A large group of works try to generalize the interest point detectors and descriptors of images to the data available. Because of the different nature of different data types the definition and development of features change accordingly. The main categories of such features are surface features, volumetric features and spatio-temporal features.

2.3.1 Object surface features

Many people have tried to derive heuristics and encodings of 3D shapes and objects that help to process them in an efficient way. The first approaches date back to 1984 with the work of B. Horn, Extended Gaussian Images [144]. Since then numerous approaches and features have been developed. The main common objective is to have a low dimensional yet discriminative description of three dimensional objects and shapes. There are many ways one can separate these methods according to their characteristics. A common distinction is global and local features. Global features describe the whole object whilst local ones describe a small neighborhood around a point on the object. The final description of the object is comprised by a collection of such local descriptions.

2.3.1.1 Global features

Global features usually try to aggregate low-level structural and geometric statistics of the complete objects like point pair distances, surface normals and curvature. Their advantage is the very low dimensional representation they offer in comparison with local descriptors that make object retrieval much faster. Unfortunately they require

the whole object to be available and fully separated from the environment [114]. Thus they are very limited in real world scenarios where objects are partially occluded and usually blended in their environment. Some examples of global methods are the Extended Gaussian Images [144], shape distributions [266], the light field descriptor (LFD) [46], the spatial structure circular descriptor (SSCD) [93] and the elevation descriptor (ED) [329]. For a more comprehensive review of global features the reader is referred to [93, 329, 114].

2.3.1.2 Local features

Local features describe some properties of the local neighborhood of an object's surface points. In order to describe a complete object, a set of these local descriptors have to be used. Depending on the needs of an application a different scheme of accumulating these local features is used. For example, for object recognition the local features of an object in the repository are added to a feature library. These features are searched for candidate correspondences with the features of a scene, which vote for specific objects and poses [112]. Bronstein et al. [37] incorporated the well established "Bag of Features" model of computer vision to 3D shape retrieval, in which the local features are translated to "visual words", or in this case "shape words", in order to obtain a global compact description of the full object. When tackling the scene semantic segmentation task, these features are considered as the data primitives in order to construct geometric unary potentials that are considered in an CRF pipeline [334, 335].

As mentioned above, local descriptors encode information of a neighborhood around a point. In order to exclude points that do not carry enough information, feature detectors are introduced. These detectors usually find points whose neighborhoods exhibit large variance of some property, e.g. fast and multiple changes of the surface normals. Given a detector, a set of "highly informative" points is detected. Then, one can extract local descriptors only for those points and describe an object or scene only using these points neighborhoods. Since most real world applications deal with varying scales of objects, as well as a variety of occlusions and deformations, feature detectors and descriptors must be invariant to scaling, rigid and non-rigid deformations as well as illumination changes. Moreover, they need to be repeatable, and unique. A very comprehensive study on surface detectors and descriptors has been published in [112]. In this work we will give a brief overview of the available detectors and descriptors.

2.3.1.2.1 Detectors. Interest point, salient or *keypoint* detectors are a classic first step to object description, since they define which points of the surface are the most

important for describing the object. A generic and popular division of detectors depends on whether they are scale invariant or not [378, 112]. Although scale invariance is an important feature, not all detectors have that ability. Some of them take the scale or neighborhood size, in which they will detect keypoints, as an input. Consequently, detectors are classified as fixed-scale or adaptive-scale keypoint detectors.

Most fixed-scale keypoint detectors have two common steps [378]. They first compute a quality measurement across all points. Then, the points are checked for saliency by checking whether they are local maxima of the quality measurement. As an example we describe the detector defined by Mokhtarian et al. [249]. A point is declared as interest point if its curvature is larger than the curvature of every 1-ring neighbor, where the k-ring neighbors are defined as the neighbors that have k edges distance. On the other hand, adaptive-scale detectors, inspired by the works of image detectors, first construct a scale-space and then search for local maxima of a defined function along the scale-space [378]. For example, Zaharescu et al. [438] build a scale-space by applying Gaussian filters directly on the 3D mesh and detect points as the extrema of the DoG space. For an extensive review of keypoint detectors the reader is referred to [378, 112].

2.3.1.2.2 Descriptors. Local surface descriptors can be subdivided according to different factors. For example, they can be subdivided according to the invariance properties, i.e., invariant to rigid or non-rigid transformations, invariant to scaling etc. The most common division for surface features is according to their encoding, i.e., histograms, point signatures and transformations [112, 376], which we will follow in this work as well.

Histograms are a broadly used type of feature description, not only in describing 3D surface features but also in image and video analysis. Histograms accumulate different measurements of the neighborhood of a point and use that as a feature. Histograms have been very popular due to their simplicity combined with high descriptive capabilities. Three dimensional surface histogram descriptors can be subdivided to spatial distribution histograms (SDH), geometric attribute histograms (GAH) and oriented gradient histograms (OGH) [112].

SDH accumulate in histograms the spatial relationship, e.g. pair point distances, of points in a neighborhood. One of the first examples of SDH descriptors is the spin images (SI) [166, 164]. The spin image is a two dimensional histogram. First, all the neighboring points are transfered to a cylindrical coordinate system starting from the interest point. The points are expressed with the radial distance α and the elevation distance β. The 2D histogram accumulates the number of points existing in a square of the $\alpha - \beta$ plane. Other examples include the extensions of the SI, Scale Invariant SI

(SISI) [63] and Tri-SI [114, 109], the generalization of shape context (SC) [22], 3DSC [91], and the Rotational Projection Statistics (RoPS) [113]. More recent examples are the TOLDI [426], RSM [283], BRoPH [450] and the MVD [107].

GAH accumulate geometric properties of the neighborhood of a point, e.g. angle between surface normals. Some examples are the Local Surface Patch (LSP) [47], the THRIFT [90], the point feature histogram (PFH) [305], its fast counterpart the fast point feature histogram (FPFH) [304] and the Signature of Histograms of Orientation (SHOT) [376].

OGH accumulate gradients of various metrics of the surface. This kind of descriptors are closely related and inspired from image descriptors like SURF [19] and SIFT [225, 226]. Some examples are the 2.5D SIFT [223], the meshSIFT [231], the meshHOG [438], 3DLBP [238], 3DBRIEF [238] and 3DORB [238].

Yang et al. [425] proposed a descriptor (LFSH) which combines SDH and GAH. Specificaly, they use histograms of a depth map, point distribution and deviation angle between normals.

Signatures describe the local neighborhood of a point by encoding one or more geometric measures computed individually at each point of a subset of the neighborhood [376, 112]. Some examples of signature descriptors are the Exponential Map [259] and the Binary Robust Appearance and Normal Descriptor (BRAND) [67], a binary descriptor that encodes geometrical and intensity information from a local patch. This is achieved by fusing intensity variations with surface normal displacement.

Transforms. These descriptors perform a transformation of the surface to a different domain and describe the neighborhood according to the characteristics of the surface on that domain. For example, Rustamov [303] performed a Laplace-Beltrami transform whilst Knopp et al. [178] performed a Hough transform on a voxelized representation of the surface. Other examples of transform descriptors are the Heat Kernel Signature (HKS) [355], its scale invariant variation (SI-HKS) [38] as well as the more recent Wave Kernel Signature (WKS) [11].

A collection of the most important, according to this study, surface features is shown in Table 2.1. The features are shown together with what, in our opinion, is their most important contribution to the field.

2.3.1.2.3 Rotation Invariance. A common goal for most descriptors is to achieve rotational invariance. In order to achieve that they try to find a repeatable and unique Reference Angle (RA) or local Reference Frame (LRF) to which the local patch or neighborhood is rotated before they describe it [164]. The first approaches used the surface normal as a reference vector in order to achieve rotation invariance. Although

Method	Year	Comments
SI [166, 164]	1998	Most sited surface descriptor
PFH [305]	2008	captures multiple characteristics
FPFH [304]	2009	improved computational efficiency of PFH
2.5D SIFT [223]	2009	SIFT for depth images
HKS [355]	2009	invariant to non-rigid transformations
mesh-HOG [438]	2009	extension of HOG [62] descriptor for triangular meshes
3D-SURF [178]	2010	extension of SURF [19] descriptor for triangular meshes
SI-HKS [38]	2010	scale invariant extension of HKS
SHOT [376]	2010	signatures of histograms, balance between descriptiveness and robustness
CSHOT [377]	2011	extension of SHOT descriptor to incorporate texture information
WKS [11]	2011	invariant to non-rigid transformations, scale invariant, outperforms HKS
TriSI [109]	2013	rotation, scale invariant and robust extension of SI descriptor
RoPS [113]	2013	unique and repeatable LRF, robust to noise and mesh resolution
3DLBP [238]	2015	Generalization of LBP to 3 dimensions
3DBRIEF [238]	2015	Generalization of BRIEF to 3 dimensions
3DORB [238]	2015	Generalization of ORB to 3 dimensions
LFSH [425]	2016	combines depth map, point distribution and deviation angle between normals.
TOLDI [426]	2017	robust to noise, resolution, clutter and occlusion LRF. Multi-view depth map descriptor
RSM [283]	2018	uses multi-view silhouette instead of depth map. Outperforms RoPS
BRoPH [450]	2018	binary descriptor, combines depth map and spatial distribution.
MVD [107]	2019	Extremely low dimensional. Performs similar to SotA descriptors in Object Recognition.

Table 2.1: A collection of surface descriptors with the most influence on the field, according to our study. The table shows the most important contribution of the work to the field. For a more comprehensive study of surface descriptors the reader is referred to [112].

the surface normal is easy and fast to compute, it is very sensitive to noise. Other methods use the Singular Value Decomposition (SVD) or Eigenvalue Decomposition (EVD) [259, 448, 34]. Unfortunately these methods do not produce a unique LRF and in order to tackle that multiple descriptors are extracted per point. A good overview and comparison of these methods is given in [376]. Moreover, they propose their own method which is more robust to noise and tackles the limitations mentioned above. To do that it computes the EVD of a weighted N-nearest neighbor covariance matrix, in combination with the sign swapping of [34].

2.3.2 Volume features

In some applications, the data of interest are not represented by surfaces, but by volumes. Some examples include voxelized representation of the objects, as well as 3D images, mainly medical images, like 3D ultrasound, CT scans and MRI scans [50, 256]. In some cases, videos are considered as three dimensional data where the time dimension is considered equivalent to the two spatial ones [321]. In order to describe the content of these kind of data, scientists generalized one of the known Interest Point detector and descriptor of 2D images to 3D, namely Lowe's SIFT detector and descriptor [225, 226].

Scovanner et al. [321] were one of the first that tried to generalize the SIFT descriptor to the three dimensional case. Although they did extend the SIFT descriptor they did not generalize the detector as well. The method picks random points in the volume as salient points and then describes them in a similar fashion to the SIFT. Orientation invariance is achieved by computing the dominant solid angle of the gradient and rotating the neighborhood around the point so that the solid angle is equal to zero. Finally, the neighborhood is split into eight sub-regions and a gradient orientation histogram is computed per region. The final descriptor is the concatenation of these histograms, which results in a 2048-D vector. They tested their descriptor on action recognition and showed that their method performs better than the regular 2D-SIFT.

At the same time, Cheung and Hamarneh [50] developed independently their own generalization. In contrast to Scovanner et al.'s work [321], they generalized both the descriptor and the detector. Moreover, instead of generalizing to the 3D case, they generalized to the nD case making their method applicable to many more datasets and applications. They use $n-1$ directions, with β bins for each, resulting in β^{n-1} bins in total. The gradients are computed using hyperspherical coordinates. They tested their method on 3D MRI of the brain and 4D CT scans of a beating heart.

Allaire et al. [9] focused on the 3D case. They observed that the aforementioned methods failed to account for the tilt that a neighborhood can have, resulting in the

Method	Data Type	Comments
Scovanner et al. [321]	Video	first 3D SIFT
Cheung and Hamarneh [50]	3D MRI & 4D CT	detector & nD
Allaire et al. [9]	3D CT, MRI, CBCT	detector & account for tilt
Ni et al. [256]	3D Ultrasound	US noise filter and smoothing

Table 2.2: Extensions of the SIFT descriptor to 3D volumetric data.

need for an extra angle in order to have full orientation invariance. For detecting points they extended Lowe's method by computing the Difference of Gaussians (DoG) in a manner similar to Lowe. The local minima/maxima of the DoG in the scale-space are picked as interest points. After detection in the scale-space, feature points are filtered and localized. The remaining points are described as follows. First, they find the dominant solid angle and for each angle with magnitude above 80% of the maximum they calculate the tilt. As with the solid angle, every angle that has a magnitude more than 80% of the maximum is considered as a different interest point. They evaluated their method on 3D registration and segmentation of clinical datasets such as CT, MR and CBCT images.

Ni et al. [256] used a similar method to the one developed by Allaire et al. [9] and adapted it for optimal description of ultrasound content, which is very noisy. They used the same filtering techniques at the detection stage with different thresholds, necessary due to the increased noise of ultrasound images. Besides the extension of Lowe's detector, they also applied the Rohr3D detector developed by [296]. It first defines the cornerness as the determinant of the matrix C, given by equation 2.5.

$$C = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{2.5}$$

where I_{ij} are the second order intensity gradients of a voxel. The local maxima of the cornerness are then detected as interest points. For description they do not use all three angles defined by [9] but only the two constituting the solid angle, like in [321]. They evaluate their method on 3D ultrasound (US) registration and compare it to the original 3D SIFT of Scovanner et al. [321].

An overview of the aforementioned methods, together with the milestone of each work, is given in Table 2.2.

2.3.3 Spatio-temporal features

As with images and three dimensional representation of objects, traditional approaches that deal with videos follow the same regime. First, a number of points is defined as interest points. These points are either detected through some saliency measurement, which means that their neighborhood is considered as very informative, or they are densely sampled, as in [171]. These points are then used to describe the whole sequence of frames (either 2D or 3D). There are many methods that try to detect and describe these kind of interest points.

First traditional approaches dealing with time dependent data, like video, either used a collection of 2D features, i.e., image features, to describe the clip or consider time as an extra dimension equivalent to the spatial ones and thus represent the clip as a 3D volume. As such, simple extensions of the image features to the 3D case are used to describe the volume [321]. Although this method produced good results at the time, the different nature of the time dimension as well as the large variance in sampling frequencies by different sensors, i.e., frame rate, motivated scientists to develop methods that describe spatio-temporal volumes whilst regarding time separately. These features are called spatio-temporal features. The new interest points are known as space-time Interest Points (STIPs).

2.3.3.1 STIP detectors

The first STIP detector was proposed by Laptev [190]. It's an extension of the Harris corner [123], called Harris3D. The Harris3D operator considers different scales in the space and time dimensions. To achieve that it convolves the video sequence f with a Gaussian kernel g given by equation 2.6.

$$L(\cdot;\sigma_l^2,\tau_l^2) = g(\cdot;\sigma_l^2,\tau_l^2) * f(\cdot) \tag{2.6}$$

where the spatio-temporal Gaussian kernel is given by:

$$g(\cdot;\sigma_l^2,\tau_l^2) = \frac{1}{\sqrt{(2\pi)^3\sigma_l^4\tau_l^2}} \cdot \exp\left(\frac{-(x^2+y^2)}{2\sigma_l^2} - \frac{t^2}{2\tau_l^2}\right) \tag{2.7}$$

where σ_l^2, τ_l^2 are the spatial and temporal variances respectively and x, y are the spatial coordinates whilst t is the temporal one. Given a space and a temporal scale, a corner or interest point is found by finding the local maxima of the corner function given by equation 2.8.

$$H = det(\mu) - k\ trace^3(\mu) \tag{2.8}$$

where μ is the 3 by 3 second-moment matrix weighted by a Gaussian function, given by equation 2.9. In a later work, Laptev and Lindeberg [192] extended the detector

in order to be velocity adaptable, which provides invariance to camera motion. In order to achieve that they considered the transformation caused by camera motion as a Galilean transformation, which is computed iteratively. This approach was later used by [191] for motion recognition. Schuldt et al. [318] combined the feature size adaptation of [190] and the velocity adaptation [192] in a single framework.

$$\mu = g(\cdot; \sigma_i^2, \tau_i^2) * \begin{bmatrix} L_x^2 & L_x L_y & L_x L_z \\ L_x L_y & L_y^2 & L_y L_z \\ L_x L_z & L_y L_z & L_z^2 \end{bmatrix} \tag{2.9}$$

Another very popular spatio-temporal detector is the one developed by Dollár et al. [68], known as cuboids. The motivation behind their detector lies in the observations that i) corners are very sparse in images and even sparser in videos and ii) there are movements, like opening and closing of a jaw, that do not include corners and thus if only corners are chosen to represent a video clip, many actions will not be recognizable. STIP are detected at the local maxima of the response function given in equation 2.10.

$$R = (I * g * h_{ev})^2 + (I * g * h_{od})^2 \tag{2.10}$$

where $g(x, y; \sigma)$ is a 2D Gaussian smoothing function applied only on the spatial dimensions and h_{ev} and h_{od} are a quadrature pair of 1D Gabor filters, given by equations 2.11, applied temporally. The scale of the feature in the spatial dimensions is defined by the Gaussian (σ) whilst in the temporal dimension by the quadrature pair ($\tau, \omega = \frac{4}{\tau}$).

$$\begin{aligned} h_{ev}(t; \tau, \omega) &= -\cos(2\pi t\omega) e^{-\frac{t^2}{\tau^2}} \\ h_{od}(t; \tau, \omega) &= -\sin(2\pi t\omega) e^{-\frac{t^2}{\tau^2}} \end{aligned} \tag{2.11}$$

Bregonzio et al. [33] observed that the aforementioned detector has some drawbacks. The Gabor filters applied in the temporal dimension are very sensitive to noise and produce many false detections in textured scenes. Moreover, it fails to recognize slow movements. In order to deal with these drawbacks they propose their own STIP detector which works in two steps. The first step is simple differencing between consecutive frames in order to produce regions of interest in which there is motion. The second step is to apply, spatially, a 2D Gabor filter.

Oikonomopoulos et al. [261] followed a different approach. They extended to the spatio-temporal case the approach of Kadir and Brady [167]. They first defined a measure of saliency based on the amount of information change in a neighborhood, which they expressed by the entropy of the signal in the neighborhood. The extension

to the spatio-temporal case is done by considering a cylindrical neighborhood instead of a two dimensional circle.

Wong and Cipolla [415] argued that all the above methods detect interest points using only local information, which produces a lot of false positives in the presence of noise. In order to counter this drawback they proposed an alternative approach which uses global information in order to detect interest points in a video sequence. In order to do so they applied non-negative decomposition of the sequence, which is represented by a two dimensional matrix, in which each column is a frame of the video. The result of the decomposition is a number of subspaces ϕ and transitions χ. By applying Difference of Gaussians (DoG) on the subspaces and the transitions, they detect spatio-temporal interest points. They compared their method with the aforementioned approaches on gesture recognition using the same description for all detectors, and showed that their method outperforms the rest.

Inspired by the work of Laptev [190], Willems et al. [413] proposed an new detector which instead of utilizing the second moment matrix μ (given by equation 2.9) they utilized the Hessian matrix H given by equation 2.12. The points are detected at the local maxima of the saliency measurement S given by equation 2.13. Unlike the 2D case [20], maxima of S do not ensure positive eigenvalues of H which means that saddle points will also be detected.

$$H = \begin{bmatrix} L_{xx} & L_{xy} & L_{xz} \\ L_{xy} & L_{yy} & L_{yz} \\ L_{xz} & L_{yz} & L_{zz} \end{bmatrix} \tag{2.12}$$

$$S = |\det(H)| \tag{2.13}$$

Yu et al. [433] developed a generalization of the FAST [298] detector to the spatio-temporal case, which they call V-FAST. For each candidate point they considered three 2D planes, the XY, XT and YT planes. They applied the FAST detector in each plane. If the point is detected as interest point in the spatial domain (XY plane) and at least one of the time comprising planes (XT or YT) then the point is considered as a STIP.

Cao et al. [42] observed that from all STIPs detected by Laptev's [190] detector, only 18% belong to a specific action whilst the rest belong to the background. Inspired by this phenomenon, Chakraborty et al. [44] proposed an new pipeline for STIP detection. They initially detect spatial interest points (SIPs) using the Harris detector [123] and then apply background suppression and other temporal and spatial constraints in order to keep only features relative to the motion in the sequence.

Finally, Li et al. [211] proposed a new detector, the UMAM-detector. The video is transfered to a Clifford algebra based representation. There, a vector is extracted for each pixel which contains both motion and appearance information. In this new space

Method	comments	Year
Harris3D [190]	first STIP detector	2003
Harris3D + velocity adaptation [192, 318]	limit camera motion detections	2004
Cuboids [68]	more dense point detection	2005
Bregonzio et al. [33]	limit false detections & detect slow movements	2009
Oikonomopoulos et al. [261]	information based saliency	2005
Wong et al. [415]	use of local and global information	2007
V-FAST [433]	efficient computation	2010
Chakraborty et al. [44]	limit background detections	2012
Li et al. [211]	unified motion and appearance	2018

Table 2.3: Existing spatio-temporal detectors. The left column shows the name of the descriptor together with the paper that proposes it, the middle column the contribution of the method to the field and the right column the year the method was published.

they apply a Harris corner detector to detect STIPs. According to their experiments the UMAM-detector outperform all the aforementioned detectors and some deep learning methods, in classification performance.

All the above detectors are summarized in Table 2.3, together with their contribution to the field.

2.3.3.2 STIP descriptors

In order for the STIPs to be represented in an optimal form for machine learning pipelines, special descriptors are defined that try to capture important information for the neighborhood of the STIP. Most proposed descriptors can be categorized depending on the type of measurements they contain or the way they quantize that information. More specifically, the most typical measurements taken to describe a STIP are the N-jets [179], Gaussian gradient field (similar to HoG and SIFT [225, 62]) or optical flow field [24]. These measurements are usually quantized or vectorized by histogramming or Principal Component Analysis (PCA) [193, 191].

The N-Jets represent a collection of point derivatives (up to Nth order) at a specific scale of the scale-space representation L, given by equation 2.14.

$$J(g(\cdot;\sigma_0,\tau_0) * f) = \{\sigma L_x, \sigma L_y, \tau L_t, \sigma^2 L_{xx}, ..., \sigma\tau^{N-1} L_{yt..tt}, \tau^N L_{tt..tt}\} \qquad (2.14)$$

The Gaussian first order gradient field is also computed on the scale-space represent-

ation L, in order to make the descriptors invariant to scaling and noise. The optical flow field represents the movement in a clip at each pixel by a velocity vector field. There are a lot of methods that try to efficiently and accurately extract that vector field. For a good overview of the optical flow estimation field the reader is referred to [354].

As mentioned above there are many ways to accumulate information over the spatio-temporal neighborhood. The most common ones are histogramming and applying PCA. Histogramming is either applied globally, i.e., one histogram over the STIP neighborhood, or on several small neighborhoods around the STIP. In the later case the separate histograms are concatenated in order to constitute a single descriptor. PCA is usually applied on a number of IP of a train set in order to obtain D most significant dimensions defined by the eigenvectors.

Laptev et al. [193, 191] tested a number of different descriptors both in terms of measurements accumulated and in the type of accumulation. Their study showed that, on average, local histograms on adaptive scales perform better than the rest of the approaches. Moreover, methods based on the first order gradient field outperform both optical flow and the N-Jets.

In a parallel work, Dollár et al. [68] performed a similar comparison. They tested normalized pixel values, first order intensity gradients and optical flow values. They tried all the above measurements by flattening the cuboid and within global or local histograms. Finally, on all descriptors, they applied PCA to reduce the dimensionality. According to their experiments histogramming did not benefit performance and thus concluded to the flattened values with PCA. As with Laptev et al.'s experiments, the gradient based descriptors showed higher overall performance than the rest.

Niebles et al. [257] extended the aforementioned descriptor. They first smoothen the image at a specific scale and then extract the intensity gradients. The apply this function for several scales and then apply PCA to get the final descriptor. Their method indeed outperforms Dollár et al.'s [68] method but it is still outperformed by Laptev et al.'s [191] histogram of gradients, with velocity adaptation.

Laptev et al. [194] proposed a combined histogram of gradients with a histogram of optical flow. Their descriptor together with the non linear SVMs managed to outperform all previous methods on the KTH dataset [318]. Willems et al. [413] extended the known SURF descriptor [19] to the spatio-temporal case. Their implementation differentiates between the spatial and temporal dimensions by setting a different number of bins, as well as different scales (σ and τ). They evaluated their method on the mouse behavior dataset as well as the KTH and they achieve comparable to the state of the art results.

Klaser et al. [177] designed a new 3D HoG descriptor. They introduced a generalization of the orientation binning of the known SIFT descriptor by introducing a

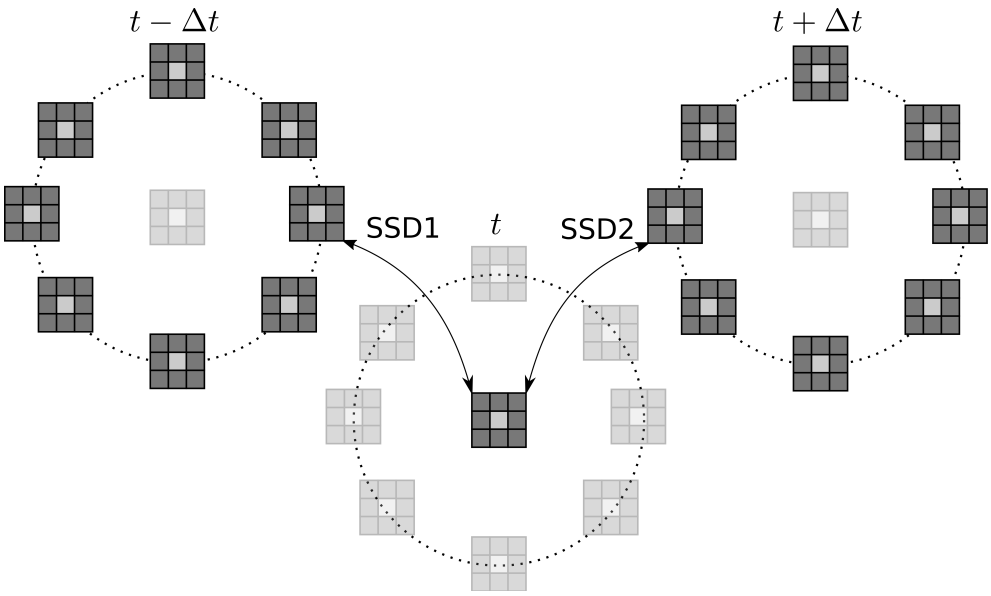

Figure 2.7: An illustration of the encoding process of LTP. For each of 8 different locations at time $t - \delta t$ and the same locations at $t + \delta t$ SSD distances of 3x3 patches to a central patch at time t are computed [429].

normal polyhedron, dodecahedron or icosahedron, and considering each face of the polyhedron as a bin. The angle of the gradient vector to the surface normals of the faces is computed and if its smaller than a threshold, the projection of the gradient vector to the surface normal contributes to the respective face's bin. Moreover, they generalized the integral image method of [388] to the integral video method. The integral video is a representation of the video volume that helps the fast computation of average gradients. Given a video volume $\nu(x, y, t)$ and its three first order partial derivatives $\nu_{\partial x}, \nu_{\partial y}, \nu_{\partial t}$, the integral video, $iv(\cdot)$, of direction j is given by:

$$i\nu_j(x, y, t) = \sum_{x'<x, y'<y, t'<t} \nu_{\partial j}(x', y', t') \tag{2.15}$$

A block of video **b** is first divided into SxSxS sub-blocks. For each sub-block the average gradient and its contribution to the histogram bins are calculated. The final descriptor is a concatenation of several such histograms computed on MxMxN blocks around the STIP. Willems et al. [412], inspired by the quantization of Klaser et al. [177] extended the method of [413] to quantize the gradient orientations in the same way.

Yeffet and Wolf [429], inspired by the Local Binary Pattern descriptor [262], pro-

posed the Local Trinary Pattern (LTP) a spatio-temporal motion descriptor. The main idea of the descriptor is to compare patches between frames instead of pixels within an image. Eight patches neighboring the pixel in question in the previous and next frames are defined, as well as a "central" patch which includes the pixel in question, as seen in Figure 2.7. A so-called trit is calculated for each spatial location (i, j) according to the following rule:

$$\begin{aligned} -1 \; if \; SSD1 < SSD2 \\ 0 \; if \; SSD1 = SSD2 \\ +1 \; if \; SSD1 > SSD2 \end{aligned} \tag{2.16}$$

where SSD is the sum of squared differences between the patches (Figure 2.7). A global descriptor is calculated by combining the trinary patters for all available pixels in histograms. First spatial histograms are created by splitting each frame in $(m \times n)$ patches. The resulting histograms are then merged temporally to create one global spatio-temporal descriptor.

2.3.3.3 3D space

Due to the inexpensive available sensors, scientists extended the STIPs to the 3.5 and 4 dimensional cases as well. To the best of our knowledge, the first to define detectors and descriptors for higher than 2 + Time dimensional data are Xia and Aggarwal [420]. Their detector is similar to Dollár et al. [68]'s Cuboids. The motivation behind their method is that due to the nature of depth images, detectors developed for color based STIP detection tend to find many points in the background and thus introducing a lot of noise in the description of a clip. In order to avoid that they introduced a correction function that smoothens out depth map specific type of noise. After the detection of the Depth-STIPs (DSTIPs) the information of the spatio-temporal neighborhood is described by a occupancy histogram.

In later work Oreifej and Liu [265] generalized the Histogram of surface Normals (HON) [361] to four dimensional surfaces (HON4D) and applied it on 3D Action Recognition. Finally, Rahmaniet al. [285] proposed the Histogram of Oriented Principal Component (HOPC). Their descriptor calculates the principal components of the scatter matrix of spatio-temporal points around an interest point and create a histogram of principal components for all points in a neighborhood. In a later work, they also proposed a detector in order to filter out points that are irrelevant [284]. Their method first computes the ratio of sequential eigenvalues. If the surface is symmetric then at least one of these ratios is going to be one. Thus they define a threshold, and if a ratio is below that the point is excluded. Otherwise the neighborhood of that point is considered informative enough to be of interest.

2.3.3.4 Trajectories

Driven by the poor generalization performance of the aforementioned approaches, researchers proposed a new strategy for handling the time dimension [237, 244, 356]. Instead of describing the change in the temporal dimension in a local manner as with the spatial ones, researchers tried to describe motion using trajectories of spatial interest points and their spatial description.

More specifically, Matikainen et al. [237] track features in a video using the standard KLT method [227, 328]. For every tracked feature they keep a vector of frame-by-frame position derivatives. The resulting vector is the trajectory-feature. These features are then clustered and the Bag of Words (BoW) model is implemented. The final action classification happens using an SVM. In parallel work, Messing et al. [244] proposed a very similar feature which they call velocity history. The difference with the aforementioned method is that they quantize the velocities in eight directions and five magnitudes. Moreover, the classification is done by a generative mixture model instead of the BoW approach. Sun et al. [356] proposed a different approach, but in the same direction. Instead of the KLT method, they find trajectories by applying frame-by-frame SIFT feature matching. According to their results, this is a more robust approach for feature tracking. Then, the visual characteristics of each trajectory is described by the average SIFT descriptor tracked. In order to describe the temporal dynamics of the trajectory, a Hidden Markov Chain (HMC) is employed that is trained on the spatial development of features. Finally, the inter-trajectory context is encoded with their proximity descriptor.

Wang et al. [395, 396], inspired by the success of the aforementioned methods as well as the dense sampling of features in images [260], proposed a combination, the dense trajectories. The trajectories are sampled on multiple scales on a spatial grid via dense optical flow. Finally, the area around the trajectories is described by the HOG-HOF spatio-temporal descriptor. Their method achieved state of the art results at the time, on many benchmarks. In later work, Wang and Schmid [397] proposed an improvement on the dense trajectories. They tracked camera movement and used it to reject trajectories caused by it. Moreover, they applied the estimated camera movement as a correction to the optical flow, in order to extract camera motion invariant trajectories.

2.4 Datasets and benchmarks

One of the main motives behind the research on higher than two dimensional data is the large availability of datasets comprised by such representations. Depending on the application and the type of data different datasets and benchmarks are proposed,

both small and large scale. In this section we will give an overview of the well known and current benchmarks and large datasets for the domain of computer vision in higher dimensions and we categorize them according to their intended application. To be more precise, numerous small scale datasets and benchmarks exist that are meant for very specific applications. Nonetheless, for each type of data, i.e., 3D scene, action in video, objects etc., there are some large scale datasets that help evaluate the data representation methods that can be applied on many different tasks. These are the datasets that are presented here and are categorized according to the type of data they deal with, namely object understanding, scene understanding and video understanding. More specific concepts can be added, like video retrieval, but due to the small number of datasets, they are grouped together in a category called "other datasets".

2.4.1 Object understanding

There is a large collection of datasets with various 3D models of objects used for object understanding tasks, like detection and classification, shape understanding and more. These datasets either contain 3D images or scans of real objects, e.g. [313, 330] or they might contain designed objects like CAD models [419]. Moreover, different datasets are used for different tasks. For example, the LINEMOD dataset [136] is used for object detection, classification and pose estimation, whilst the Princeton shape benchmark (PSB) [330] focuses on different classification themes. Besides these state of the art datasets, there are also smaller but well known datasets. Some of these are Lai et al.'s [188] dataset, the big bird [339] and the SHREC [206]. For a good overview of all these benchmarks and datasets the reader is referred to [89]. Table 2.4 gives a comparison of the state of the art datasets.

The largest datasets available, to date, are datasets that contain designed models and objects instead of real scans, largely due to the longstanding graphics communities. Some of the well known datasets are the Princeton shape benchmark [330], which consists of 161 object classes and a total of 1814 models. The ModelNet [419], a dataset which consists of 151,128 3D CAD models in 660 categories. ShapeNet [45] is also a recent database, which tries to make even more detailed annotations than just object labels. The raw dataset consists of roughly 3 million models, from which 220,000 have been classified into 3,135 categories. Besides the raw dataset the authors also made two subsets. The first, called shapeNetCore, consists of 51,300 models in 55 common categories, with extra alignment annotations and the second, shapeNetSem, consists of 12,000 models from 270 categories. In addition to manually verified category labels and consistent alignments, they are also annotated with real-world dimensions, estimates of their material composition at the category level,

Dataset	Data Type	#Images	#Objects	#Object Cat	6DoF pose
PSB [330]	PSG	-	1 814/6 670	161/1 271	-
ModelNet [419]	CAD	-	151 128	660	-
ShapeNet [45]	CAD	-	3M/220K	- /3 135	-
shapeNetCore [45]	CAD	-	51 300	55	-
shapeNetSem [45]	CAD	-	12K	270	-
YCB [40]	RGB-D	600	75	-	No
Rutgers APC [289]	RGB-D	10K	24	24	Yes
Redwood [52]	RGB-D	23M	> 10K	44	No

Table 2.4: Large scale datasets and benchmarks for object understanding. PSG stands for polygonal surface geometry. DoF stands for degrees of freedom.

and estimates of their total volume and weight [45, 314].

As mentioned above there are also datasets with scanned real life objects instead of designed models. One example is the YCB object and model set [40]. It consists of every-day object scans from 75 object categories. For each object the dataset includes 600 RGB-D images coupled with 600 high resolution RGB images, segmentation masks as well as calibration information and texture-mapped 3D mesh models. The Rutgers APC RGB-D dataset [289] consists of more than 10 thousand RGB-D images. In total it contains 25 objects along with their 6DoF pose. Choi et al. [52] created a dataset of scanned 3D objects with an RGB-D camera. The dataset provides a variety of different objects, from bottles of shampoo to sculptures and even an howitzer. They grouped these objects in 44 categories. Besides the raw RGB-D videos they also provide 3D reconstruction for some of the objects. Some example 3D reconstructions can be seen in Figure 2.8. For more information about the reconstruction technique and the number of objects reconstructed we refer the reader to the original paper [52]. All the above datasets are summarized in Table 2.4.

2.4.2 Scene understanding

Scene understanding is a domain that refers to machine learning pipelines that are able to perform several tasks given a scene, such as object detection and localization, scene semantic segmentation, scene classification and more. In general it includes all methods that increase the understanding of a scene through visual means. Due to the significant qualitative difference in terms of applied sensors and the structure of indoor and outdoor scenes, they are considered as separate problems.

One of the first “bigger” datasets is Berkley’s B3DO dataset introduced by Janoch

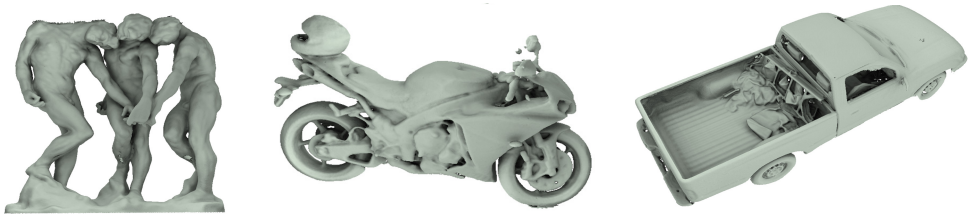

Figure 2.8: Example scans of real objects from Choi et al.'s [52] dataset. Original Figure from [52].

et al. [158]. It is comprised by 849 images from 75 scenes captured by an RGB-D camera. Overall it includes more than 50 object classes. One of the most known datasets and most used benchmarks for indoor scene understanding is the NYUv2, created by Silberman et al. [335] in 2012. It is comprised by a set of indoor videos taken with RGB-D camera, resulting in 795 labeled images with 894 object classes. Xiao et al. [421] tried to provide a richer dataset, in the sense that the segmentation is not pixel wise but there is a better 3D representation of the objects. The result is the SUN 3D dataset [421] which also provides point cloud segmentation produced by Structure from Motion (SfM). Song et al. [344] realized that existing datasets were limited in (i) the number of scenes and sequences they include and (ii) they have sequences from a single RGB-D camera type. They created a more large scale and generic dataset, the SUN-RGBD dataset. They achieved that by taking images from existing datasets and also introducing their own. The result was a dataset with 10,335 RGB-D images of a total of 47 scene categories and 800 object classes. Hua et al. [145] created sceneNN, a dataset that contains 100 scenes with per-pixel annotation of objects. The scenes are 3D reconstructed on triangular meshes.

Most of the scene understanding datasets suffer from small variation of well annotated scenes and limited number of objects. Handa et al. [121] created a method for dataset creation in order to tackle these problems. They claimed that their system is able to create a virtually infinite number of scenes with various objects in them and perfect per-pixel annotation. They accomplish that by using computer graphics to artificially create scenes. They also acquired a large number of 3D CAD models, from some of the datasets mentioned in Section 2.4.1, and randomly placed them in the scenes. The resulting dataset can be used in order to properly pre-train a CNN which can be then fine tuned on a real world dataset. McCormac et al. [241] continued this work with the goal to create a dataset, called SceneNet RGB-D, with annotation not only for semantic segmentation, object detection and instance segmentation but also scene trajectories and optical flow. For comparison, example real scenes from the

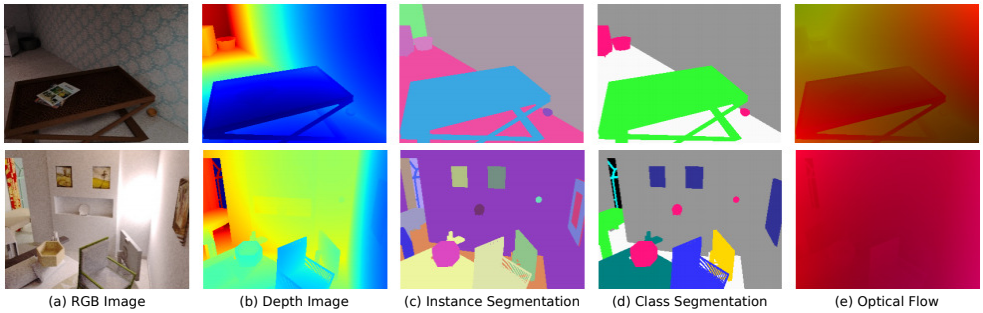

Figure 2.9: Example Images from the SceneNet RGB-D dataset [241]. (a) RGB Image, (b) Depth Image, (c) Ground Truth Instance Segmentation, (d) Ground Truth Class Segmentation, (e) Optical Flow

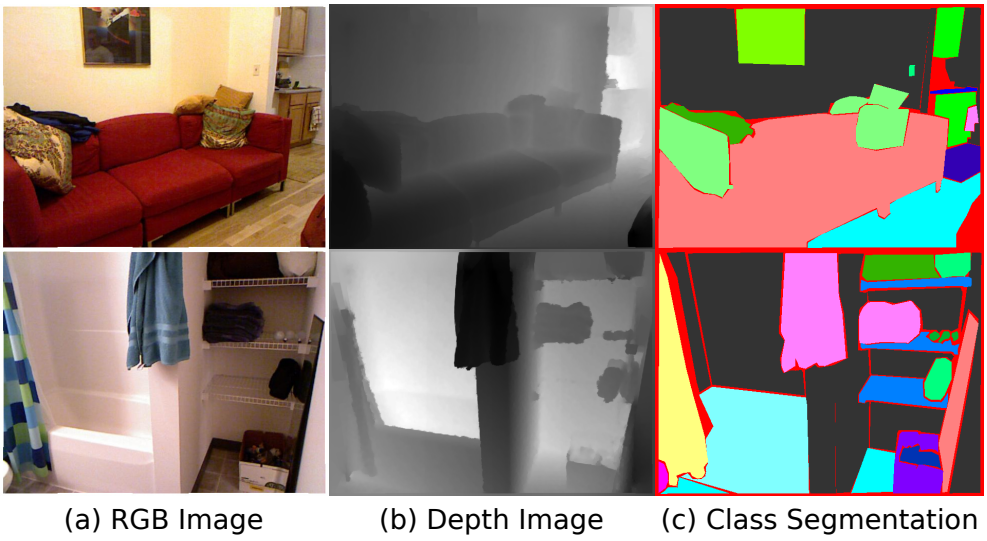

Figure 2.10: Example Images from the NYUv2 dataset [335]. (a) RGB Image, (b) Depth Image, (c) Ground Truth Segmentation

NYUv2 are shown in Figure 2.10 and some artificial scenes from the SceneNet RGB-D in Figure 2.9. Similar to their work, Song et al. [346] created a synthetic 3D scene dataset called SUN-CG, which contains 45,622 synthetic scene layouts created using Planner5D [346]. Dai et al. [61] introduced a much bigger dataset with real world scenes than all the aforementioned. It consists of 1,513 scenes with overall 2.5M RGB-D frames and more than 36K object instances. All scenes have been reconstructed and labeled manually.

For a good comparison the datasets, together with their features and details, are shown in Table 2.5. As with the object datasets of the previous section, we can see that the artificial datasets are orders of magnitude larger than the datasets that contain images and videos of real scenes.

The aforementioned datasets focus only on indoor scenes and objects. When considering outdoor scenes, the availability of datasets decreases significantly. One of the reasons is the low quality of the RGB-D sensors in open space. Most of the existing datasets are limited to 2D RGB images, for example Richter et al.'s [290] dataset and the SYNTHIA dataset [297]. Nonetheless the KITTI dataset [98], although built for pedestrian, car and cyclist detection on images, it also includes Velodyne 64E range scan data with 2D and 3D bounding boxes for 7500+ frames. Moreover, the Sydney Urban Objects dataset [282] contains labeled Velodyne LiDAR scans of 631 urban objects in 26 categories.

2.4.3 Video understanding

Video understanding is a broad field. There are several research areas that can be considered subfields of video understanding, for example action recognition, video retrieval as well as object detection and tracking. Object detection and tracking is highly related to object and scene understanding which are covered in the previous sections. The most active areas in video understanding are action recognition and video retrieval. Most of video understanding related research focuses on action recognition and more specifically human action recognition. Action recognition is the main research area for which new representation approaches and video understanding methods are developed and tested on. There is a large collection of datasets and benchmarks whose content relates a lot to the evolution of the "action recognition" research. Good overviews of these benchmarks and their historic value are given by Hassner [124] and Idrees et al. [151]. In this section we will give an overview of the state of the art datasets and benchmarks.

One of the well known and used benchmarks today is the Human Motion Data Base (HMDB51) [186]. It consists of 6,766 video clips, each representing one out of 51 "every day" actions collected from various sources on the Internet. The annotation

Dataset (Reference)	RGB-D video	Per-pixel annotation	traj. GT	RGB Texture	# scenes	# layouts	# object classes	3D Models avail.
B3DO [158]	No	Key frames	No	Real	75	-	> 50	No
NYUv2 [335]	Yes	Key frames	No	Real	464	464	894	No
SUN 3D [421]	Yes	3D point cloud + Video	No	Real	254	415	-	Yes
SUN RGB-D [344]	No	Key frames	No	Real	-	-	~ 800	No
sceneNN [145]	Yes	Video	Yes	Real	100	100	≥ 63	Yes
SceneNet [121]	No	Key frames	No	non-pr	57	1 000	-	Yes
SceneNet RGB-D [241]	Yes	Video	Yes	pr	57	16 895	255	Yes
SUN-CG [346]	Yes	Video	Yes	non-pr	45 622	45 622	84	Yes
ScanNet [61]	Yes	3D + Video	?	Real	1 513	?	≥ 20	Yes

Table 2.5: Big scale datasets and benchmarks for indoor scene understanding. The first column shows the name of the dataset, the second column shows whether the dataset provides RGB-D video of the scenes, the third one the level of the annotation, the forth one whether trajectory ground truth is included and the fifth whether the data are real, or synthetic. “pr” means photorealistic whilst “non-pr” means non-photorealistic. Sixth, seventh and eighth columns show the number of scenes, layouts and object classes respectively and the ninth, last, column shows whether the dataset provides 3D models of the objects present in the dataset.

is done in a redundant way (each label is verified by at least two humans) in order to ensure its quality. Moreover, every video has some extra meta-data such as camera view-point and motion. Although, for todays standards, this is a small to medium scale dataset, it is still widely used due to its very accurate ground truth. A similarly popular dataset is the UCF101 [348] dataset. It consists of 13,320 clips which belong to one of the 101 action classes of the dataset. These classes are single person actions as well as person to person interactions. Caba Heilbron et al. [39] proposed the ActivityNet, a dataset of human activities. It contains about 20 thousand videos from 203 different human activities. Most videos are between 5 and 10 minutes long with a maximum of 20 minutes. In these videos the classes are manually annotated and specified in time. This results in about 30 thousand human-annotated clips of a specific human action. Recently, Kay et al. [170] proposed the Kinetics dataset, the largest human action dataset to date. It consists of 306,245 trimmed clips from YouTube that include human-object and human-human interactions. The clips are classified to one of the 400 possible classes and were annotated using Amazons Mechanical Turk (AMT) [170].

One of the largest datasets at the time of this paper is the Sports 1M dataset [169]. It consists of 1 million YouTube videos assigned to one of 487 classes. These classes are sport actions such as road bicycle training, track cycling, monster truck etc. These videos have been automatically annotated according to the video tags. Moreover, these are five minutes videos so the class might be a small proportion of the whole video. Due to the above reasons, the labeling of the data is very weak and thus hard to properly evaluate different algorithms. Jiang et al. [161] released the Fudan-Columbia Video Dataset (FCVID), a dataset that contains over 90 thousand videos from 239 categories. Most of these categories are actions like “making cake” whilst there are some object and scene categories as well. The videos are collected from YouTube and are manually labeled. Abu-El-Haija et al. [4] released the largest to date video dataset, the YouTube-8M. It consists of about 8 million videos with 4 thousand labels in total. Each label is supposed to shortly explain the content of the video. For example, a video of biking on dirt roads and cliffs would have a central topic/theme of Mountain Biking, not Dirt, Road, Person, Sky [4]. Possible labels are also filtered out according to some characteristics. For example a label must be visually recognizable and should not require specialized knowledge.

Barekatain et al. [18] introduced an aerial view video dataset for human action recognition it consists of 43 videos with varying camera position and motion. The videos are staged and include multiple actors that perform several actions out of the 12 defined classes. Goyal et al. [104] introduced the ”something - something” dataset. It is an action recognition dataset where the labels are of the form ”something” action ”something”, for example Dropping [something] into [something]. The dataset

Dataset	#Videos	#Clips	#Classes	m-l	trimmed	m-ann
HMDB51 [186]	3 312	6 766	51	No	Yes	Yes
UCF101 [348]	2 500	13 320	101	No	Yes	Yes
Sports 1M [169]	1M	-	487	No	No	No
ActivityNet [39]	19 994	28 108	203	No	Both	Yes
FCVID [161]	91 223	91 223	239	No	No	Yes
YFCC100M [374]	0.8M	-	-	-	No	-
YouTube-8M [4]	$\sim$ 8M	-	4 800	Yes	No	No
Kinetics [170]	306 245	306 245	400	No	Yes	Yes
Okutama - Action [18]	43	43	12	Yes	Yes	Yes
Something-Something [104]	108 499	108 499	174	No	Yes	Yes
Moments in Time [250]	1M	1M	339	No	Yes	Yes

Table 2.6: Big scale datasets and benchmarks for video understanding. First column depicts the name of the dataset, the second third and forth the number of videos, clips and classes respectively. The fifth column (m-l) specifies whether each video can have multiple labels. The sixth column depicts whether the videos are trimmed and the final column (m-ann) shows whether the videos/clips are manually annotated.

is manually annotated and consists of about 108K short videos ($\sim$ 4 seconds) with 174 action classes and more than 23K object names. Monfort et al. [250] introduced the "Moments in Time" dataset, a big dataset of one million 3-second clips with 339 classes of verbs, which are picked from the VerbNet.

A summary of all the above datasets can be found in Table 2.6. For a more comprehensive review on Human Action Recognition datasets the reader is referred to [340].

2.4.4 Other datasets

Besides the scene understanding, object and action classification datasets mentioned in the previous sections there are also datasets for a big variety of applications. For example the Cornell dataset [162] is a dataset built with the goal of training robotic grasp detection on various objects. It contains 1035 RGB-D images with 280 graspable objects annotated with several positive and negative graspable rectangles. For the goal of shape deformation, Yumer et al. [436] created a dataset, containing objects from various categories and their deformations scales, that was later also used for other research purposes, for example [435]. Garcia and Vogiatzis [95] proposed the MovieDB, a dataset for different image-to-video retrieval tasks [94]. The TACoS

dataset [286], with action labels on videos as well as natural language descriptions with temporal locations, and the Charades-STA [92] have been used for text-to-clip video retrieval. The DiDeMo dataset [10] has been introduced for temporal localization given natural language, but has also been used for the purpose of text-to-clip video retrieval [423]. Recently, the Hollywood 3D dataset was proposed [118] which contains 650 stereo clips with 14 action classes, together with stereo calibration and depth reconstruction.

2.5 Research areas

2.5.1 Object classification and recognition

A very well researched topic that includes three dimensional representation of the world is 3D object classification and recognition. Given an object with a 3D representation, a system has to classify the category or the instance of the object. Although, conceptually, a straight forward task, it constitutes a very complex problem because it requires efficient and complicated representation methods that are able to capture the high level content from the raw representation. Moreover, it is a fundamental step in understanding the three dimensional world. As a result, it is considered a very good benchmark for 3D world representation methods. Two large clusters of object classification and recognition methods are identified, depending on the data they process. These are methods that try to classify full 3D objects, usually available as CAD models, and methods that classify RGB-D images of objects.

2.5.1.1 RGB-D object recognition

The first methods applied for this task are inspired by the imaging community. Researchers were trying to develop hand-crafted descriptors that were then used to discriminate between different objects. One of the first examples of such methods is the work of Lai et al. [187], which extracts spin images from the depth map and SIFT features from the RGB values. They create two different vocabularies using the Efficient Match Kernel (EMK) method. The resulted representation is fed into a linear SVM (linSVM), a Gaussian kernel SVM (kSVM) and a random forest (RF) and compare their performance on their RGB-D object dataset [187, 188]. Other works apply the well know Kernel descriptors (KDE) [27] on several characteristics of an RGB-D image whilst other use the Hierarchical Kernel descriptor (HKDE) [25], which applies the kernel descriptor also on the kernel representation instead of only on the pixel level, creating a hierarchy of kernel descriptors.

With the recent success of deep convolutional neural networks (Deep CNN) in image analysis tasks, researchers try to extend these methods to the three dimensional representations as well. One of the first approaches towards training features from data from more than two dimensional representations was done by Bo et al.[28] who learned features in an unsupervised manner from RGB-D data and Socher et al. [342] who trained a Convolutional-Recursive neural network. Alexandre [8] proposed a transfer learning method where different networks are used for each channel (three color channels and depth map). Instead of training each network from scratch they take as initialization method the weights of the best performing network trained so far. Since their experiments aim to test the increase of performance using the transfer learning method, they do not compare to other methods. Unfortunately, they also use a subset of the original dataset which makes the comparison to other methods impractical. Eitel et al. [78] propose a fusion architecture, in which two networks are trained, one on the RGB data, pre-trained on ImageNet [302] and an other on the depth map. The two networks are combined with a late fusion to produce the final result.

Method	Category	Instance
linSVM [187]	81.9 ± 2.8	73.9
kSVM [187]	83.8 ± 3.5	74.8
RF [187]	79.6 ± 4	73.1
KDE [27]	86.2 ± 2.1	84.5
HKDE [25]	84.1 ± 2.2	82.4
Upgraded HMP [28]	87.5 ± 2.9	92.8
CNN-RNN [342]	86.8 ± 3.3	-
Fus-CNN [78]	91.3 ± 1.4	-

Table 2.7: Performance of object recognition methods on the RGB-D Object recognition dataset [187]. The performance is measured by classification accuracy. Left column describes the method, the middle column presents the results on the category level classification benchmark and the right the Instance level classification performance.

We summarize the performance of all the above methods, on the RGB-D Object recognition benchmark [187, 188] in Table 2.7. The benchmark used for this comparison provides two different tasks. One is the category level classification, where a classifier is supposed to label the type of object. The second is instance level classification, where the classifier is supposed to identify the specific object from different views and in different environments.

2.5.1.2 3D object classification

As mentioned in Section 2.2.2.1, early deep learning approaches on learning from a three dimensional representation define two design concepts. The first approach is to

train CNNs straight from a 3 dimensional representation of voxel grids [419], while the second one applies 2D projections. In the context of 3D object classification the projection is done via a multi-view approach [353]. In order to compensate for the increased computational complexity of three dimensional convolutional kernels, the networks are trained on very low resolution inputs (usually do not exceed 30^3 voxels) and they are comprised by a small number of layers, restricting the complexity of the features learned. Most of the proposed methods for 3D object classification belong to one of these two categories.

Both strategies have received a lot of attention. The 3D kernel approach was first applied in this research area by Wu et al. [419]. They utilize a 3D Convolutional DBN, which is trained on their newly proposed ModelNet. The idea of 3D convolutional kernels is further explored with the works of Maturana and Scherer [240], who introduced a 3D CNN as well as a new representation approach. They argued that the voxel representation usually used disregards a lot of information. The reason is than the value of a voxel can only be "occupied" or "empty". Thus, they propose a representation in which the voxel value represents the posterior probability of the cell being occupied. Later, Qi et al. [280] tried to improve the 3D CNN approach in three stages: 1) new network structure, 2) data augmentation, 3) feature pooling. They proposed two new architectures. The first improves the performance by training on an extra auxiliary task, i.e., object classification from sub-volume, and the second mimics the behavior of the multi-view CNN. A specialized convolutional layer performs a 2D projection of the 3D volume and then an image CNN performs the classification. The object is presented to the network in various orientations. The network performs orientation pooling to obtain a orientation invariant representation. Sedaghat et al. [323] argued that the network should, at some point, be able to estimate the pose of an object in order to be able to classify it. Thus, they added an auxiliary task, namely pose estimation and increased the performance of the tested networks by a significant margin on a variety of benchmarks. Hegde and Zadeh [128] fused multi-view and 3D CNNs, whilst Brock et al. [35] defined blocks of layers based on the inception [358] and ResNet [127] architectures, namely Voxception, Voxception-downsample and Voxception-ResNet.

The projection to lower dimensions has also received a lot of attention. As mentioned above, Su et al. [353] proposed a multi-view approach, where pictures of the object are taken from 20 different views and processed by a pre-trained, on ImageNet, network. Shi et al. [327] proposed the projection of the shape on a cylinder, described in Section 2.2.2.1 and Qi et al. [280] improved the multi-view approach by introducing a multi-resolution extension of data augmentation. Wang et al. [392] argued that the view pooling approach of the multi-view strategies fails to take into account important information from different views since only one survives the pooling. In order

Method	Type	ModelNet10	ModelNet40
shapeNet [419]	3D	83.54	77.32
MV-CNN [353]	2D proj.	-	90.1
VoxNet [240]	3D	92.0	83.0
DeepPano [327]	2D proj.	88.66	82.54
MVCNN-MultiRes [280]	2D proj.	-	91.4
MO-AniProbing [280]	3D	-	89.9
ORION [323]	3D	93.9	89.4
FusionNet [128]	Both	93.11	90.8
VRN [35]	3D	93.61	91.33
VRN-Ensemble [35]	3D	**97.14**	**95.54**
Wang et al. [392]	2D proj.	-	93.8

Table 2.8: Performance of object classification methods on the ModelNet 10 and 40 benchmarks [419]. The performance is measured by classification accuracy. Left column describes the method, the middle column presents the results on the ModelNet10 classification benchmark and the right one the performance on the ModelNet40 classification benchmark.

to alleviate this issue they introduced a recurrent clustering and pooling layer based on graph theory. With their approach they achieved state of the art performance on the ModelNet 40 dataset.

The performance of the above methods is summarized on Table 2.8. Although, for the most part multi-view approaches were outperforming the voxel-based approaches, the work of Brock et al. [35] with the Voxception-ResNet approach managed to outperform all multi-view approaches. Nonetheless, their strategy needs to train multiple big networks from scratch whilst the work of Wang et al. [392] only needs to fine tune the networks lowering the training time by multiple orders of magnitude while still having competitive performance.

2.5.2 Semantic segmentation

An important research area using such three dimensional datasets is semantic segmentation. Semantic segmentation or scene labeling is the procedure of labeling every pixel, or voxel, in an image, as seen in Figures 2.9 and 2.10. Most methods tackle this problem by utilizing only RGB images. Since depth sensors became widely accessible, people started to use this extra information in order to make better predictions. The

methods that utilize these features are heavily influenced by their RGB-only counterpart. In this work we will only focus on the methods that utilize the depth information since we are interested in applications and methods that deal with higher than two dimensional data. Most traditional methods tackle this problem by utilizing hand-crafted features, introduced in Section 2.3, in a Conditional Random Field (CRF) or Markov Random Field (MRF) model. The usual pipeline is to over segment the image in super pixels, extract features from the superpixels and then use them to construct unary and pair wise potentials for the CRF or MRF model. With the success of deep learning in image classification, researchers try to adapt these methods for three dimensional semantic segmentation as well.

2.5.2.1 Traditional approaches

The first to tackle this problem in the higher than two dimensional representations are Silberman and Fergus [334]. In their work they use a CRF-based approach and define unary potentials encoding spatial location and pairwise potentials encoding relative depth. The unary potentials are learned from a neural network using local descriptors. They evaluate their approach on their NYUv1 dataset, which they construct for the purpose of their project. Moreover, they test different descriptors, both image and depth descriptors, and compare their performance. They extended their work [335], by introducing a new extended version of NYU, NYUv2, which is still one of the most used datasets for benchmarking scene segmentation algorithms. Couprie [57] explored other CRF-like approaches in order to improve the computational complexity of the algorithm. Ren et al. [288] improved the segmentation performance by using kernel descriptors [26, 27] and by combining superpixel MRF with segmentation trees for contextual modeling.

Koppula et al. [180] oversegmented a 3D pointclound [80] based on smoothness and surface continuity. The segments are then labeled using an MRF. Gupta et al. [116, 115] introduced gravity direction prediction and combine it with appearance and depth contours coupled with either RDFs or SVMs. Hermans et al. [133] proposed an RDF classification which is refined using a Dense CRF.

Deng et al. [65] argue that although hand-crafted features have achieved much progress, the geometric relationship of local and global object spatial configurations have not been studied, leaving room for improvement. They propose a method that jointly considers local and global spatial configurations by adding mutex constraints on Gupta et al.'s method [116].

[351, 352] proposed a method for real time semantic segmentation of RGB-D videos. Their method estimates camera pose by using simultaneous localization and mapping (SLAM) and recognize and segments object classes in the image, using a

GPU implementation of random forests. Müller and Behnke [251] used the output of this method as a feature for unary node potentials on a CRF model. For pairwise depth-sensitive features they use a number of features such as contrast, vertical alignment, depth difference and more. The unary and pairwise functions are learned using structural support vector machines (SSVM).

[174] propose a new CRF model that combines appearance and geometric information in various levels of the models hierarchy. They introduce a new region growing algorithm to extract fundamental geometric planes, and extract appearance and geometric unary potentials from these planes. More information is included by adding pairwise potentials between these planes and extra higher order potentials defined on cliques, which encompass planar patches.

2.5.2.2 Deep learning

As mentioned above, a lot of methods that utilize deep learning have been also developed. Within this category we can identify two clusters of methods. The first represents a transition from the aforementioned traditional methods to the pure deep learning ones. In these the networks are used in order to extract features that are then used to classify segments or superpixels either using graph models like CRF and MRF or some other classifiers. Some examples are the works of Couprie et al. [58] who adopted a multi-scale approach by adapting previous work in semantic segmentation [85, 86], Höft et al. [142] and Wang et al. [391] who proposed a multi-modal unsupervised method that would automatically learn rich high and low-level features from an auto-encoder.

The second cluster is initiated by the work of Long et al.[224], who introduced the Fully Convolutional Networks (FCN) in order to produce per pixel, dense, classifications. These networks are end-to-end trainable and do not rely on other methods. Eigen and Fergus [77] trained a multi-scale convolutional neural network to predict the depth-map, surface normals and provide semantic segmentation. Wang et al. [401] designed two convolutional and deconvolutional networks, one trained on depth values and one on RGB values. These networks explicitly try to learn common features between different modalities (see Section 2.2.2.2). Li et al. [213, 212] proposed an LSTM-CNN approach called LSTM-CF. They first transform the depth data to the HHA form [117]. They use three LSTM networks, namely one for each modality, that explore vertical contexts. The results are concatenated and fed into the last LSTM network which performs bi-directional propagation along the horizontal direction. Hazirbas et al. [125] argue that the aforementioned method is very complicated, resulting in inefficient training. Moreover, they argue that the HHA representation of the depth information is very computationally expensive to compute whilst it does

not provide any new information over the depth channel. In order to gain as much information possible from the depth channel without increasing the preprocessing computational complexity or the complexity of the model they propose the FuseNet. They extended the work of Noh et al. and Badrinarayanan et al. [258, 14] to also utilize depth information, by having two encoder parts, one processing the RGB data and one the depth data. Every few layers of the encoders, the feature maps of the depth data are inserted to the RGB processing network.

Park et al. [271] adapted the very successful work of Lin et al. [215], RefineNet, to use RGB-D data. They do that by introducing the multi-modal feature fusion (MMF) block which fuses feature maps from an RGB specific and a depth specific network. These fused representations are used as input to the refine blocks of RefineNet [215]. Valada et al. [383] used the SSMA (Section 2.2.2.2) module to fuse geometric and color features, while Deng et al. [64] used the interaction stream that they introduced, described in Section 2.2.2.2 as encoders. The outputs of the streams are fused together and sent to a decoder to predict the class labels. Qi et al. [281] introduced a method which combines the two methodologies. They do that by utilizing graph neural networks (GNN) instead of a CRF or MRF. They experiment with unary potentials extracted from a pretrained VGG as well as a ResNet. Moreover, as an update function for the GNN they try both MLP and an LSTM.

The performance of the aforementioned methods on the NYU benchmarks [334, 335] can be seen in Table 2.9. For all benchmarks, the highest performance is reported by deep learning methods, and more specifically the second cluster of the deep learning methods. Nonetheless, the best performing traditional approaches still outperform the first cluster of the deep learning approaches. Table 2.10a shows the performance evaluation of the methods on the SUN-RGBD dataset and 2.10b on the ScanNet. From both tables, it can be seen that the RDF-Net of Park et al. [271] outperforms all other methods by a large margin, on every benchmark tested.

2.5.3 Object detection

Object detection is a conceptually similar task to scene semantic segmentation. The main difference between the two fields is that object detection does not classify all data points in a scene. Instead, the scene is parsed in order to detect the position of specific objects. Traditional methods that tackle these problems mainly depend on template matching [112, 366, 367].

2.5.3.1 Traditional methods

Template matching methods are very popular due to the absence of need for a large annotated training set and training times. On the other hand, a system that is able

Method	Year	Shallow/ Deep	NYUv1	NYUv2					
				4 Classes		40 Classes			
			pixacc	pixacc	clacc	fwavacc	avacc	pixacc	clacc
SIFT+MRF [334]	2011	Shallow	56.6 ± 2.9	-	-	-	-	-	-
Silberman et al. [335]	2012	Shallow	-	58.6	-	-	-	-	-
KDES [288]	2012	Shallow	*76.1 ± 0.9	-	-	-	-	-	-
Gupta et al. [116]	2013	Shallow	-	-	-	45.1	26.1	57.9	*28.4
Hermans et al. [133]	2014	Shallow	59.5	69.0	-	-	-	-	-
RF + SP + CRF [251]	2014	Shallow	-	*72.3	*71.9	-	-	-	-
Khan et al. [174]	2014	Shallow	-	69.2	65.6	-	-	-	-
Gupta et al. [115]	2015	Shallow	-	-	-	45.9	26.8	58.3	-
Deng et al. [65]	2015	Shallow	-	-	-	*48.5	*31.5	*63.8	-
Stückler et al. [352]	2015	Shallow	-	70.9	67.0	-	-	-	-
Couprie et al. [58]	2013	Deep	-	64.5	63.5	-	-	-	-
R-CNN [117]	2014	Deep	-	-	-	47.0	28.6	60.3	35.1
FCN [224]	2015	Deep	-	-	-	49.5	34.0	65.4	46.1
Eigen and Fergus [77]	2015	Deep	-	**83.2**	-	**51.4**	34.1	65.6	45.1
Wang et al. [401]	2016	Deep	**78.8**	-	**74.7**	-	-	-	47.3
RDF-152 [271]	2017	Deep	-	-	-	-	**50.1**	**76.0**	**62.8**
3DGNN [281]	2017	Deep	-	-	-	-	43.1	-	59.5

Table 2.9: Performance evaluation of different methods on the NYU datasets (v1 & v2). First column refers to the methods and the papers that present them. The second column is the year that the methods where published. The third column shows whether the method is follows a traditional approach, or shallow learning, or a deep learning approach. Fourth column shows the per pixel average accuracy on the NYUv1 dataset using all 13 classes. The rest of the columns show performance results on the NYUv2 dataset. The fifth and sixth column refer to the 4-class segmentation task whilst the rest on the 40-class segmentation task [335]. pixacc refers to the average per pixel accuracy, clacc refers to the average per class accuracy, fwavacc is the frequency weighted average accuracy and avacc refers to the meanIU, or the mean Intersection over Union [125]. We highlight the per category (shallow or deep) best performance with a * and the over-all best with **bold** (in which case the asterisk is omitted).

Table 2.10: Semantic segmentation performance evaluation of different methods. The first column refers to the method and the second shows the year the method was published. The rest of the columns show the performance results on the respective benchmark. pixacc refers to the average per pixel accuracy, clacc refers to the average per class accuracy and avacc refers to the meanIU, or the mean Intersection over Union [125]. We highlight the best performance with **bold**. It should be noted that all methods shown in this table are deep learning methods.

Method	Year	SUN-RGBD		
		clacc	avacc	pixacc
*FCN [224]	2015	41.13	30.46	68.35
LSTM-CF [212]	2016	48.1	-	-
FuseNet-SF5 [125]	2016	48.3	37.29	76.27
RDF-152 [271]	2017	**60.1**	**47.7**	**81.5**
SSMA [383]	2018	-	38.4	-

(a) Performance evaluation of different methods on the SUN-RGBD 37 class benchmark [344]. *FCN refers to the work of [224] but the performance on the SUN-RGBD is reported by [383].

Method	Year	avacc
SSMA [383]	2018	57.7
RFB-Net [64]	2019	**59.2**

(b) Performance evaluation of different methods on the ScanNet dataset [61] as reported by the benchmark website.

to detect objects from many classes has to try and fit many templates which makes the algorithms slow for real time applications such as robotics [135]. The pipeline of most of template matching method can be divided into four steps [112]: **1) feature extraction**, where local features are extracted from template object points, **2) feature matching** where features from the new scene are corresponded to the library (template) features. There are several matching strategies, such as threshold based, nearest neighbor approach (NN) based and nearest neighbor distance ratio (NNDR) based [247, 112]. An important step of the matching step is the strategy with which the feature library is searched, such as the naive brute force search, kd-trees [113, 109], hash tables [91] and more. Then, each match is used for a **3) hypothesis generation** where the match is voting for an object position and pose. Popular methods include pose clustering [245, 72, 113, 109, 448], the RANSAC algorithm [269, 270], hough transform [178, 375] and more. Finally, all hypotheses are leveraged in the final **4) hypothesis verification** step. For a more comprehensive study on traditional methods that perform object detection and recognition the reader is referred to [112].

In recent years, learning based methods started getting attention for the problem of 3D object detection and 6 DoF pose estimation. These methods rely on the size of the dataset to provide both positive and negative examples for objects, contrary to template methods that only require positive examples, and thus are more difficult to generalize to new types of scenes. For example, Rios-Cabrera and Tuytelaars [292] extended Hinterstoisser et al.'s [135, 134] method and learn the templates in a discriminative fashion. Since the pipeline of this method is mostly a template method we consider it a "learning boosted" template method. More representative works of learning based method are Song et al.'s [345] sliding shapes method, where a sliding window is fed into an Exemplar-SVM for each object and Bonde et al.'s [30] work who followed a "sliding cuboid" approach coupled with a random forest classifier. Tejani et al. [366, 367] proposed a novel method using latent-class hough forests to detect objects in a 3D scene. The descriptions of objects are used in order to train a multi class hough forest. Recently, Hinterstoisser et al. [137] extended the point pair feature (PPF) [72], by introducing novel sampling and voting schemes that are not influenced as much by occlusion.

2.5.3.2 Deep learning

Besides the traditional methods, people have tried to adapt the deep learning methods to perform object detection and 6 DoF pose estimation as well. One of the first approaches was introduced by Wohlhart et al. [414]. They adapted the template approach to take advantage of the discriminative power of CNNs. In their approach they

trained a CNN to produce a patch descriptor. The training is done by their triplet error function which leverages between similar and dissimilar objects. This is accomplished by having two terms on the error function. One term punishes descriptor difference of the same object with different background noise and clutter whilst the other punishes small distance of descriptors of different objects. This descriptor is then used to describe both the templates and image patches. The detection and pose estimation is done by the k Nearest Neighbors approach. In order to introduce scale invariance, they adjust the size of the patch according to the distance of the center of the patch to the camera so that the "real world" size of the patch is always the same. Balntas et al. [16] extended the aforementioned method by adding one more term to the error function that takes into account the pose of the object as well, punishing similar/dissimilar descriptors for different/same pose respectively. With this method they achieved state of the art results on a modification [414] of the LINEMOD dataset [135].

Krull et al. [185] proposed an analysis-by-synthesis method for 6DoF pose estimation. Their method uses a CNN to calculate an energy function which compares an scene object estimate with a rendered version of the same object from the estimated view point. Doumanoglou et al. [71] followed a similar approach to that of [366, 367], but instead of the LINEMOD [135] descriptor, they trained a sparse AE to learn a discriminative representation. Kehl et al. [172], inspired by [414], introduced a new method, in which they applied a similar approach but instead of training a CNN on positive and negative objects, they used unsupervised learning with a CAE. They argue that by using unsupervised learning, negative examples are not needed any more and the method becomes less dataset-dependent.

Although there are some datasets that are commonly used, such as [135], a lot of different variations of it as well as different datasets are proposed and used for evaluation, making a global comparative analysis very difficult.

2.5.4 Human action classification

To the best of our knowledge, human action classification is the most researched area concerning image sequences, or videos. Given a short video clip that contains humans performing an action, an automated system has to be able and classify the given action. Depending on the dataset these actions might be single human actions, like standing up or opening door, single human actions in a sport environment, or person to person actions, like hugging or kissing. Like with many fields that deal with visual data, early approaches include template matching while a bulk of traditional approaches define interest points in order to describe small clips and using these interest point and special descriptors try to classify the actions. More recent approaches

try to apply deep learning methods to this field as well.

2.5.4.1 Traditional methods

As stated above the very early approaches are based on templates [29, 326, 325]. Unfortunately these methods can not define single templates for each activity which renders them insufficient [295]. Thus, researchers turned their attention to other models, like the Hidden Markov Model (HMM), Hidden Semi-Markov Model (HSMM), Conditional Random Fields (CRF) and support vector machines (SVMs). Another group of methods extract a representation that is derived using the STIP detectors and descriptors introduced in Section 2.3.3. Finally, a group of works exploit trajectories of points in order to describe and classify actions [237, 244, 356, 395, 396, 397], as described in Section 2.3.3.4.

Yamato et al. [424] were the first to apply HMM on the action classification problem. The method first extracts the mesh feature vector [424] for every frame I. Then all features are quantized to codewords. Each frame codeword is then used as input of the HMM. Oliver et al. [263] follow a different approach. They first extract the human positions and their trajectories and utilize a Coupled HMM (CHMM) in order describe pairwise human interactions. Wang and Mori [406] utilized the hidden CRF (HCRF) in order to classify actions. The optical flow based descriptor of [76] is used to describe each frame. Song et al. [347] proposed a hierarchical recursive sequence representation coupled with a CRF model for sequence learning. In order to retrieve the summarized representation for the next level in the hierarchy, the samples are grouped adaptively, i.e., observations are grouped together when they have similar semantic labeling by the CRF model in the previous layer. Fernando et al. [88] tried to model the evolution of the actions in an video. In order to do that he used the "learning to rank" framework on the Fisher Vector representation of each frame.

As mentioned above, many methods followed the classical approach for image classification, utilizing interest points. Schuldt et al. [318] proposed a local SVM approach combined with the BoF representation in order to classify single human actions in videos. Later Laptev et al. [194] test both HoG and HoF to describe the STIPs, as well as a number of different grids for accumulating histograms, and created a BoF representation of the clips. Finally, the clips are classified with a local-SVM [441]. From the combinations that they tested the best performing one was the HoF features in combination with a three way vertical split on the spatial dimensions and no temporal split of the movie clip.

Sun et al. [356] were one of the first to explore trajectories. They extract SIFT trajectories from the clips and measure the average SIFT descriptor along those trajectories. Wang and Schmid [397] used dense trajectories with corrected camera mo-

tion, encodes them using Fisher Vectors and finally classify them using a linear SVM. Kovashka and Grauman [182] proposed a hierarchical feature approach. They created different vocabularies for a BoF representation for multiple scales. From all the aforementioned methods, the only approach that still stands out today and can be compared to the state of the art deep learning methods is the trajectory based improved Dense Trajectories (IDT) of Wang and Schmid [397] and thus it is the only for which we report results.

2.5.4.2 Deep learning

Many deep learning approaches have been proposed for tackling the HAR task. The main bulk of works can be divided in three schemes, namely full 3D CNNs, two-stream networks and CNN-LSTM approaches. Regardless of the class of the method, besides a small number of works, the input to the networks is a small part of the video, usually referred to as clip. The length of these clips can vary from five to sixteen frames. A more detailed overview of the methods is given bellow.

To the best of our knowledge the first to apply deep learning on HAR were Taylor et al. [363]. In their work they proposed a special RBM, the convolutional gated RBM (convGRBM), which is a generalization of the gated RBM (GRBM) [242]. Their method alleviates a limitation of GRBM, the fact that it can not scale up to large inputs. Their method shares weights in all locations of an image and thus can scale to large inputs. As an old approach, this work does not fit with our classification scheme.

Ji et al. [160] proposed the first 3D CNN for action recognition. Their network has five 3D convolutional layers, one 2D convolutional layer and the output, classification layer. Since their network takes as an input only seven frames, it can not take into account actions that span for a longer period. Thus, they use a feature vector from a long span of frames, i.e., the BoW representation using SIFT features of the frames, as auxiliary input through a hidden layer. In a later work, Tran et al. [379] delved into optimizing the architecture of 3D convNets for spatio-temporal learning. Their experiments indicated that uniform kernels (3x3x3) give the best overall performance. Karpathy et al. [169] did a detailed research on what architecture can exploit the time-dimension better. They tested four different strategies, namely single frame network, early, late and slow fusion networks. Interestingly enough, the single frame network has similar performance to the rest, which means that these first approaches towards spatio-temporal understanding using deep CNNs are not able to exploit the temporal dimension as well.

Baccouche et al. [13] also proposed a 3D convolutional neural network. They deal with the long-term actions by building an RNN-LSTM network which takes as input the output of the 3D CNN network. Donahue et al. [70] proposed a very similar

architecture, stacking an LSTM on top of a CNN network and calling the complete architecture Long-term Recurrent Convolutional Neural network (LRCN). The two main differences with the model of [13] are that they train their network end-to-end and that the CNN is pre-trained on ImageNet.

Simonyan and Zisserman [337] proposed a new strategy, the two-stream networks. In this architecture one network processes the RGB values of a single frame whilst an other processes 10 stacked frames of optical flow fields. The spatial network is first pre-trained on ImageNet and thus increasing the performance of the approach. The final decision on the class of a clip is done by averaging the classification results of the separate networks. Wang et al. [403] identified as drawbacks of deep learning approaches on HAR the lack of large data and the limitation of the complexity and depth of the networks applied. In order to alleviate these issues they proposed some "good practices" for training very deep two-stream networks. The first important step is that the temporal network is also pre-trained on images and thus able to be much deeper. Second they utilized state of the art very deep networks, (VGG19 [338] and GoogleNet [359]) for both streams. Furthermore they proposed more data augmentation techniques for the videos and applied smaller learning rates. Feichtenhofer et al. [87] identified two drawbacks with the two-stream strategy as applied until then. (i) It was not able to learn correlations between spatial and temporal features since the fusion happened after the classification and (ii) the temporal scale was limited since the temporal network only considered 10 frames. Also inspired by the work of [254] they proposed a temporal fusion two-stream network. They applied feature map fusion before the last convolutional layer. They fused the two streams and activations from several frames with a 3D convolutional layer followed by a 3D pooling layer. Carreira and Zisserman [43] proposed to inflate existing architectures from images to three dimensions. They do that not only in terms of architecture but also inflate the trained parameters. Given this starting point they trained two networks, one on RGB values and one on optical flow. Finally, they averaged the outputs in order to provide a unified prediction.

Ng et al. [254] followed a different approach, where they make predictions while processing the whole video sequence rather than short clips. They tested several architectures including two-stream networks, LSTM and other temporal feature pooling mechanisms. Applying max-pooling over the temporal dimension in the last convolutional layer (i.e., convPooling) and the LSTM are the two best performing strategies for temporal handling. Their convPooling network takes as input 120 frames whilst the LSTM 30 and both give similar results. In similar work, Varol et al. [384] proposed a Long-Temporal Convolutional network (LTC). Their network is processing 60 frames per video clip. They defined a number of 3D convolutional networks, each processing different resolutions and modality, i.e., RGB and optical flow. The classification scores

Method	Year	+IDT	RGB	Flow	UCF-101	HMDB-51
IDT [397]	2013	-	-	-	86.4	61.7
Two-Stream [337]	2014	No	Yes	Yes	88.0	59.4
Karpathy et al. [169], Sport 1M pre-train	2014	No	Yes	No	65.2	-
TDD [402]	2015	No	Yes	Yes	90.3	63.2
C3D ensemble [379], Sport 1M pre-train	2015	No	Yes	No	85.2	-
Very deep two-stream [403]	2015	No	Yes	Yes	91.4	-
Two-stream fusion [87]	2016	No	Yes	Yes	92.5	65.4
LTC [384], Kinetics pre-train	2017	No	Yes	Yes	91.7	64.8
Two-stream I3D [43], Kinetics pre-train	2017	No	Yes	Yes	**97.9**	**80.2**
(2+1)D [380], Kinetics+Sports 1M pre-train	2018	No	Yes	Yes	97.3	78.7
TDD + IDT [402]	2015	Yes	Yes	Yes	91.5	65.9
C3D ensemble + IDT [379], Sport 1M pre-train	2015	Yes	Yes	No	90.1	-
Dynamic Image Networks + IDT [23]	2016	Yes	Yes	No	89.1	65.2
Two-stream fusion + IDT [87]	2016	Yes	Yes	Yes	93.5	69.2
LTC+IDT [384], Kinetics pre-train	2017	Yes	Yes	Yes	92.7	67.2

Table 2.11: Performance evaluation of different methods on the UCF-101 [348] and HMDB-51 [186] datasets. The first column refers to the method and the second shows the year the method was published. The third column specifies whether IDT is used in combination with the networks. The forth and fifth column show whether the method is utilizing RGB and optical flow inputs respectively. The sixth and seventh columns show classification accuracies of the methods on the UCF-101 and HMDB-51 datasets, respectively.

of all networks are averaged in order to produce the final prediction.

Wang et al. [402] proposed the trajectory-pooled CNNs (TDDs). Inspired by the work of [397] and the lack of CNNs in exploiting long term temporal relationships, they proposed the Trajectory pooled Deep-Convolutional Descriptors (TDDs), where they compute descriptors by computing trajectories of CNN features maps using the method of [397] and encoding them using Fisher Vectors.

Tran et al. [380] proposed to decompose the spatial to the temporal convolution, and thus creating the (2+1)D convolution which is a 2D spatial convolution followed by a 1D convolution exploiting the temporal dimension. Their top performing network is a (2+1)D, two stream network which has a much lower complexity than the top performing 3D networks, while keeping the performance competitive.

We summarize the results of some of the above methods on Table 2.11. There are several conclusions we can derive from these results. Simple 3D networks seem to be outperformed by CNN-LSTM as well as two stream networks, but the combination of them outperform the 'single solution' networks. Moreover, pretraining on large datasets with not very accurate annotation, such as Sports 1M [169], benefit the quality of the networks. Last but not least, as with many applications, the best performing traditional approach, IDT [397], is outperformed by most deep recent deep learning approaches. Nonetheless, the combination of IDT and networks produces better results, by a constantly large margin, driving us to the conclusion that the high-level hand crafted features seem to capture information that is not learned by the networks, rendering them complementary.

2.5.5 Other areas

There are numerous more research areas and applications that deal with high dimensional data. Some examples are:

2.5.5.1 Outdoor object detection

Outdoor object detection is a very well studied research topic with many real life applications, like autonomous vehicles and security. Some more specific examples of object detections are pedestrian detection, vehicle detection, like cars motorcycles and bicycles. Traditional methods first segmented the input point cloud and then classified the segments with various methods [365, 364, 393, 21]. For example, Behley et al. [21] used the BoW model to describe each segment and used it to classify it. State of the art methods take advantage of deep neural networks. Some examples are [278, 82]. Qi et al. [278] uses the pointnet++ as a base, whilst [82] utilizes 3D convolutional kernels and [207] utilizes a 2D FCN with the depth data as an extra

modality. To the best of our knowledge [278] achieves the state of the art performance on the KITTI benchmark [98].

2.5.5.2 Landing zone detection.

This is a very important task for autonomous rotorcrafts. It provides the ability to localize possible landing locations. Methods that perform such a task usually depend on LiDAR [320] data. Traditional approaches depend simple geometric properties such as terrain roughness and slope [410, 315, 165, 407]. The performance of such methods lowers significantly once the terrain is covered by low vegetation. Maturane and Scherer [239] propose a deep learning approach which utilizes a CNN with three dimensional convolutional kernels which predicts which parts of the seen terrain are safe for landing and which are not.

2.5.5.3 Structure from motion (SfM) and Simultaneous localization and mapping (SLAM)

These are very challenging tasks. SLAM is the process where the algorithm is trying to identify the position of the camera or sensor in the environment while constructing a map of the environment. SLAM is a very challenging while very interesting and important task in the field of robotics as well as augmented reality. Traditionally people were trying to match new environment parts to the constructed map by matching features (usually hand crafted) and RANSAC like algorithms. Some representative work can be found in [350, 80, 173, 81, 411, 252]. SfM is the process of building a 3D representation of a scene/environment of a camera by using multiple views, and more specifically views from the same camera as it moves in the space. It usually is part of SLAM since it tries to built a 3D representation of the local environment of the camera. A comprehensive survey on SLAM and SfM was recently published by Saputra et al. [312].

2.5.5.4 Action Recognition in 3D videos

This is a relatively new research field. As with video action recognition the target of the task is the classification of human actions in different kind of categories. The methods applied in this field can be divided into two categories depending on the type of data they process. More precisely, they process skeleton data or depth data [284]. Also methods that process color-data have been proposed but since these are much closer to the 2D Video action recognition, described in Section 2.5.4, than the rest of these methods we do not consider it as part of this section. Skeleton-based approaches first extract the joints positions, usually using the OpenNI tracking framework [332],

and then either use them [427], or information from the area around them [400, 399], to describe the motion. Depth-based approaches use either silhouettes [208, 385] or 4D histogram descriptors [284, 265, 428] in a BoW framework to describe each action and then try to classify them. In recent years plenty of DL approaches have been proposed as well. They usually utilize an RNN-LSTM on joints and skeletons [324, 73, 219] or process directly the depth data in time [404]. For a good overview of deep learning approaches the user is referred to [446].

2.6 Discussion

Although this field has come a long way, there are still a lot of challenges that the researchers face. Since most of these methods are generalized from successful methods developed for two dimensional images, all limitations and problems that arise when dealing with two dimensional images exist here as well. For example, when it comes to deep learning, the models are typically not understood and treated as black boxes [110]. Although researchers know how these models update their parameters and learn from the data, retrieving the information that they have learned is still an open research area. More specifically, although there has been done research on feature visualization [440, 336, 434], it is still unknown how to discover or understand what the networks learn and how they behave. Another inherent limitation is the typical lack of rotation invariance of the models, although some methods try to work around it. For example, Cheng et al. [49] train a specific layer to be orientation invariant. They do that by adding a penalty term to the loss function to force the layer to become rotation invariant. Although the result of the specific layer is rotation invariant the rest of the network is not. In cases where information from multiple layers is needed, such as semantic segmentation, this solution does not suffice. Another example is the work of Marcos et al. [234]. They rotate the kernels and convolve with the rotated kernels and thus obtain responses from all possible orientations. The rotation invariance of this strategy is also limited since the information of the orientation is getting lost during the orientation pooling operation.

Besides the inherited difficulties from the two dimensional case, other problems arise when trying to extrapolate to more dimensions, either when the increase is an increase of physical dimensions or if it is an increase of available modalities. A common limitation to all state of the art methods that deal with higher than two dimensional data is the high demand of resources. This limits the possible size of the deep learning methods. Moreover, as shown from the two dimensional case, these methods highly depend on the complexity and size of the resulted models [127, 358, 110, 146], which combined with the increased complexity of the data as well as the

increase of demand renders it very difficult to efficiently apply them.

According to the results shown in this chapter, the state of the art performance on volumetric data is achieved using deep learning models. As described above, these methods have many drawbacks, both inherited from the drawbacks of deep learning in general as well as drawbacks regarding computational complexity. Moreover, it is still unclear which strategy for dealing with the higher dimensionality of the data is better. To be more precise it is still unclear whether reducing the dimensionality to two is better than using three dimensional kernels. In the later case it is still unclear which representation of the data works best. All these questions are left unanswered whilst the computational complexity of the models together with the lack of very large scale, high dimensional, diverse and well annotated datasets make the unbiased comparison between approaches very hard.

Difficulties arise when processing spatio-temporal data as well. Although current results show that methods that utilize optical flow outperform methods that do not, it is still unclear how to optimally include this information. Moreover, the difference of space and time is still a challenging concept. It is still not clear how to process them in order to acquire as much information as possible from both spatial contexts as well as their temporal interactions. Furthermore, most approaches process only short term interactions and only a few process more that 16 frames long clips, and thus encoding long term interactions [384]. Processing many frames though becomes very computationally expensive and thus the question of how to optimally perform temporal and spatial pooling arises. Although there has been significant development in the field the long term impact and directions for continued advances are still unclear. Some of the limiting factors being the fundamental theory for understanding the strengths and limitations of the networks, approaches for learning with small training sets and/or the availability of accurately annotated, diverse and large scale real life datasets.

2.6.1 Major challenges

In summary, the major challenges as described by the research community are:

- Deep learning in high dimensional data is very computationally and memory expensive, limiting the capabilities of the applied approaches.
- Deep learning approaches lack invariance in many transformations, such as scale and rotation, which are usually tackled by very computationally expensive approaches.
- There exist many competing strategies for handling high dimensional data and it is still not clear which approaches are suited better for which type of data and more importantly why.

- For many applications there are not enough labeled data to properly train and test methods. Nonetheless, the past few years, in some research areas this issue has been slowly tackled by introducing large scale datasets such as the ScanNet [61] and the Moments in Time [250].

2.6.2 Future work

According to this study, there is significant room for improvement in all research areas covered in this chapter. Nonetheless, we can identify some common issues to most of them. In most cases deep learning approaches are too computationally expensive for many real world applications, whilst the traditional counterparts have much lower performance. It is important to get as high performing approaches while minimizing computational complexity and memory demands. Moreover, being able to leverage information from different modalities without performing unnecessary computations for common features whilst not missing modality specific information is very important to the whole field. Although there are similarities in the type of dimensionality increase in different research areas, the solutions applied are usually unique to the research area. It would be interesting to acquire knowledge from multiple approaches and create unified solutions.

2.7 Conclusions

This chapter presents a comprehensive review of methodologies, data types, datasets, benchmarks and applications of computer vision on high dimensional data (higher than 2D). Based on the recent research literature we identify four main data sources, namely image videos, RGB-D images and videos, and 3D object models, such as CAD models. Moreover, we identify common practices between methods that are applied on all data types despite their qualitative difference. For example, deep learning approaches and hand crafted features, such as histograms, are developed and applied on all data types and research areas considered. Most of the methods are inspired by previous work in computer vision on 2D data.

Regarding deep learning methods, we discuss the interrelationships and give a categorization of generalization of methods to higher dimensions, namely generalization in case of increase of physical dimensions and generalization in case of increase of modalities, or information per physical position. Finally, we review and discuss the state of the art methods on the most researched areas using these data, such as 3D object recognition, classification and detection, 3D scene semantic segmentation, human action recognition and more.

According to this study, some conclusions can be drawn regarding the top performing approaches. Deep learning approaches seem to outperform hand crafted feature based approaches when it comes to recognition performance in all tested settings (i.e., object classification, recognition and detection, semantic segmentation and human action classification). Nonetheless, hand-crafted feature based approaches have much lower time complexity. In some cases they can produce similar performance to the state of the art deep learning method, as shown in object detection by Tejani et al. [366, 367]. As shown in Human action Recognition, with the IDT approach [396], the hand crafted features can provide complementary information to the deep learning features increasing the overall performance of a system by a significant margin. When the number of physical dimensions is increasing, although early experiments showed that projecting information to lower dimensions and taking advantage of large available systems outperformed the raw processing of the high dimensional data, nowadays, we see an opposite trend. For example, the work by Brock et al. [35] on object detection as well as Carreira and Zisserman [43] on HAR, outperform 2D projection methods. Finally, late fusion seems to be the best performing naive strategy across the board for combining different modalities, whilst fusion in multiple levels and fusion on multiple stages of the process seem to outperform all other methods, e.g. Wang et al. [401] and Park et al. [271].

Understanding the world around us is a difficult task [222]. Although there is a lot of progress in this area, there is still a lot of room for improvement. For most data types there is no clear solution or approach that properly handles the extra dimensions. For example, even in the well studied area of video understanding, there is not a definitive way to handle the difference between space and time. Similarly, in the three dimensional static world even the optimal raw format of the data, e.g. point cloud, 3D mesh or voxelized, is unknown.