Zurich University
of Applied Sciences

School of
Engineering

InIT Institute of Applied
Information Technology

Bachelor thesis (Computer science)

Active Scene Understanding from Image Sequences for Next-Generation Computer Vision

Author	Ralph Meier Dano Roost
Main supervisor	Dr. Giovanni Toffetti Carughi
Sub supervisor	Prof. Dr. Thilo Stadelmann

Date	19.06.2020

School of Engineering

Declaration of Originality
Bachelor's Thesis at the School of Engineering

By submitting this Bachelor's thesis, the undersigned student confirms that this thesis is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members is not considered as third party).
The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the Bachelor thesis have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

City, Date:	**Name students:**
Winterthur, 19.06.2020	Dano Roost
Winterthur, 19.06.2020	Ralph Meier

Zusammenfassung

In den vergangenen Jahren konnten enorme Fortschritte im Bereich der digitalen Objekterkennung erzielt werden. Ansätze basierend auf Convolutional Neural Networks sind in der Lage, Katzen und Hunde voneinander zu unterscheiden oder zu erkennen, ob ein Bild an einem Strand oder in einem Wald aufgenommen wurde. Die meisten dieser Techniken arbeiten jedoch mit einzelnen, unabhängigen Bilder und können Informationen aus verschiedenen Blickwinkeln nicht kombinieren, weshalb sie auch kein Verständnis von Objektpermanenz haben, also dass ein Objekt immer noch existiert, auch wenn es gerade nicht sichtbar ist. Ebenfalls sind die meisten dieser Ansätze nicht in der Lage zu entscheiden, von welchem Winkel die Szene als nächstes betrachtet werden sollte, um ein möglichst komplettes Bild zu erhalten. Diese Arbeit untersucht Ansätze, um diese Mängel zu beseitigen und Computern ein tieferes Verständnis einer Szene zu ermöglichen. Dazu benutzen wir synthetische 3D-Szenen und rendern Bilder aus 36 Blickwinkeln. Unter Verwendung von Sequenzen dieser Bilder muss das System Fragen zur Szene beantworten, wie zum Beispiel die Auswertung von Form und Farbe für ein Objekt an einer bestimmten Position im 3D-Raum oder die Auflistung aller gefundenen Objekte. Wir evaluieren drei verschiedene Netzwerkarchitekturen (CNN, CNN + RNN, ConvLSTM), von denen zwei in der Lage sind, Informationen über mehrere Zeitschritte zu aggregieren. Die Kamerapose steht dabei nicht als Input zur Verfügung, so dass das System selbst lernen muss, Informationen von verschiedenen Blickwinkeln zu kombinieren. Als besonders erfolgreich zeigt sich dabei eine Kombination von Convolutional und Recurrent Neural Networks. Eine noch bessere Leistung wird erzielt, wenn das System zudem lernt, die Kamera selbst zu steuern. Mittels dieser Methode werden im Durchschnitt nur noch fünf verschiedene Blickwinkel benötigt, um bessere Resultate zu erreichen als mit jeder anderen getesteten Art der Kamerasteuerung. In Szenen mit einem Hindernis-Objekt übertrifft die Architektur, die Convolutional und Recurrent Neural Networks kombiniert, den reinen Convolutional-Ansatz um bis zu 97%. Wir können zeigen, dass unser Ansatz in der Lage ist, viele gängige Unzulänglichkeiten traditioneller Objekterkennungsansätze zu lösen, darunter das Verständnis von 3D-Okklusion, die Fähigkeit, Informationen über viele Einzelbilder hinweg zu aggregieren oder die aktive Steuerung der Kamera, um die Szene von optimalen Blickwinkeln zu erfassen.

Abstract

In recent years, computational object recognition has made huge progress. Convolutional neural network-based approaches are able to successfully distinguish between cats and dogs, or detect whether a picture was taken on a beach or in a forest. Most of these techniques, however, operate on single independent snapshots and cannot combine information from different viewpoints, which is why they do not have an understanding of object permanence (knowing that an object exists, even if it is currently not visible). Furthermore, most of these approaches are not capable to determine from where to look at a scene to acquire all available information. This work addresses these issues and creates a prototype computer vision system for acquiring a deeper scene understanding. To do so, we generate synthetic 3D scenes and render circular camera trajectories around them with 36 images each. By processing image sequences from this trajectory, the system learns to answer questions about the scene, like locating an object with a given shape and color in 3D space, or enumerating all objects present in the scene. We evaluate three different neural network architectures (CNN, CNN + RNN, ConvLSTM) of which two have the capability to aggregate information over multiple timesteps. We do not provide the camera pose as input, so the system must learn to combine information from different camera angles on its own. A combination of convolutional and recurrent neural networks proves the most successful solution to the task. This performance can be improved even further by actively controlling the camera using reinforcement learning. By doing so, the system only requires an average of five different viewpoints to reach better success metrics than with any other type of camera control. In scenes with added occlusion, the architecture that combines convolutional and recurrent neural networks proves to be very effective and outperforms the convolutional-only approach in enumerating present objects by up to 97%. We can show that our approach is capable of solving many common shortcomings of traditional object recognition approaches including the understanding of 3D occlusion, the ability to aggregate information over many frames, the understanding of object permanence or the active control of the camera in a beneficial way.

Preface

We would like to give special thanks to Dr. Giovanni Toffetti Carughi and Prof. Dr. Thilo Stadelmann for their great support during this Bachelor thesis, for their helpful tips and constructive discussions. We are very grateful to have been given the opportunity to explore and commit to the ongoing research topic on the next generation of computer vision.

Contents

1. Introduction

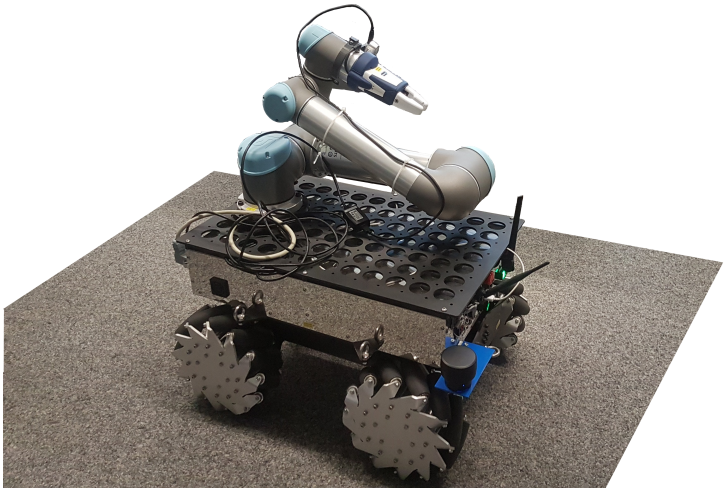


Figure 1.1.: Summit XL Steel mobile robot from ZHAW InIT Cloud Computing Lab

1.1. Use Case: Helping a Robot to Understand the World

The InIT Cloud Computing Lab (ICCLab) at Zurich University of Applied Sciences (ZHAW) is focused on research on cloud computing and cloud robotics [1]. As part of its cloud robotics initiative, research is being undertaken using a modified *Summit XL Steel mobile robot* with an attached *Universal Robots UR5* robotic arm (see Figure 1.1). This arm is equipped with an *Intel® RealSense™ D435 depth camera* which is able to perceive Red-Green-Blue-Depth (RGB-D) images using active infrared stereo technology [2].

With the goal of serving as a mobile service robot, the system is currently capable of navigating in an office environment and performing very simple grasping tasks. However, these grasping attempts often fail due to an insufficient understanding of the objects present, their geometry and their relations (such as one object obstructing access to another one). The current setup merely maps the inputs of the two RGB-D cameras into a point cloud and does not have any capabilities to segment this data into separate objects or even to classify it semantically. The point cloud acquired by the robot is of relatively low quality as it only relies on a single viewpoint. One possible way of solving these problems would be to acquire a more complete point cloud and segment it semantically using approaches like PointNet++ [3]. Without countermeasures, however, this process can be both computationally intensive and very memory consuming as there are usually large amounts of data that need to be mapped [4]. When comparing this process to how the human brain works, one realizes that it is unlikely that we humans operate with such large numbers of data points as these 3D mapping approaches do, but instead we use a much more compact scene representation [5].

1.2. Towards Neural Scene Understanding

The computer vision revolution of the past years has been driven in great parts by convolutional neural network (CNN)-based object recognition approaches which can find and label entities in images [6, 7, 8, 9]. While often applied on video sequences, most successful approaches operate on independent frames and do not take the *temporal correlation* of images into account [8]. By assuming that information for each input frame is independent and identically distributed (i.i.d.), one could lose valuable preliminary information when observing subsequent frames. This is especially problematic when observing a scene from multiple viewpoints where it would be necessary to combine independent predictions into one large representation. Gori [10] even proposed that the inability to process temporally correlated images together makes the task unnecessarily complicated, since interesting properties of such correlated images are not exploited.

As a consequence of relying on independent images, most algorithms do not have a concept of space, time or occlusion, but operate according to the principle "out of sight, out of mind". This makes it easier to acquire the skills during the training period due to lower input dimensionality, but might not lead to the desired result in the long term. Processing sequences as independent images leads to a number of shortcomings:

- **Shortcoming Nr. 1:** The system might recognize the class of an object but has no understanding of object permanence in presence of occlusion. Object permanence is the cognitive ability to know that an object exists, even if it is currently out of sight [11]. Human infants develop such capabilities in their early months [12]. Understanding object permanence could be very helpful for tasks like observing a scene from different angles. For computer vision systems, the problem of object permanence would also need to be extended to permanence across frames, so an object can be recognized as the same when processing subsequent frames.
- **Shortcoming Nr. 2:** We expect that a computer vision system would perform much better if the algorithm was aware of the things it *cannot* currently see, and actively navigated the camera to a viewpoint that reduces this uncertainty. Most popular object detection algorithm operate in a static way by analyzing what they see, but do not actually understand the geometry and thus the present occlusion of the scene.
- **Shortcoming Nr. 3:** The system has only one chance to make the right guess about the class of an object. In the real world, however, it is often necessary to accumulate information over multiple timesteps to identify far away or occluded objects. If the algorithm could combine the existing and the new information, the chance for a successful prediction would be highly significantly.

Inspired by these shortcomings, we want to explore how to employ more human-like perception in computer vision as has been encouraged in recent publications on vision theory [10, 13].

1.3. Problem Description

In this work, we want to answer the question **how computer vision systems could acquire an actual understanding of a scene** by **aggregating information from image sequences**. This includes not only identifying objects in sight, but also inferring their 3D positions and remembering them, even if they are temporarily occluded. Additionally, we examine **how active vision could improve such a system**.

We start by briefly analyzing the current state of object detection and localization in images and how it can be extended to videos. We discuss publications on the visual systems of the human brain and identify areas that could serve as inspiration for neural scene understanding systems. This is followed by an overview of existing approaches to neural scene understanding. Using the gathered information, we create a prototype system which can successfully understand simple scenes with occlusion and accumulate information over multiple frames. The system is able to localize objects in 3D space and determine their shapes and colors, even in scenes that are too complex to be captured in a single image. We extend this system with the capability to actively change its camera position to minimize

uncertainty and improve inference performance. In different experiments we can show that it is possible to create computer vision systems that do not suffer from the shortcomings presented in section 1.2. Finally, we discuss the outcome of the conducted experiments and analyze how to interpret the resulting conclusions, for example in connection with work on vision theory or with regards to sample efficiency when training computer vision systems.

With strong existing solutions for grasping tasks [14], we do not focus on this aspect in our work, but instead improve the perception system on which grasps could later be based on. We do not attempt to solve the specific use case of the ZHAW Summit XL Steel mobile robot, but aim to create an approach which is helpful in a large range of possible scenarios, similar to the way 3D mapping systems already enable robotic interaction today.

2. Literature: From Object Recognition to Deep Scene Understanding

In this chapter, we summarize literature that shows how to move from a object recognition system such as Mask R-CNN towards actual scene understanding from image sequences. We analyze what makes humans so good in scene understanding, and which tools computer science has to offer to achieve a comparable level. Finally, we summarize the most important takeaways.

2.1. Traditional Object Recognition Systems and their Limitations

With the rise of CNNs, the foundation has been set for well-performing image classification approaches such as ResNet [6] or VGG [7]. In recent years, this development has gone even further towards systems which are capable of localizing or even masking out objects in images. Popular approaches with these capabilities are for example Mask R-CNN [9] or YOLO [8]. While using different approaches, both systems have the capability of localizing and semantically classifying objects in images. Mask R-CNN is even able to mask the pixel-areas that are belonging to the object by using polygonal shapes. In the background both approaches still rely on ResNet for feature extraction.

While systems like these are highly impressive and sufficient for many cases, they are still far away from human visual perception and actual scene understanding [10]. Especially the shortcomings presented in section 1.2 apply to this kind of systems. The resulting polygon-masks or bounding boxes are not projected into 3D space but merely mapped onto the 2D coordinates of the image. If employed on sequential data, both systems would also need to reevaluate the whole frame on each timestep as they are not able to remember or track objects. This is problematic when understanding a scene, as it is often not likely for the whole scene to be visible at once, but only partially. **Therefore we believe, it is crucial for a system not to forget but to accumulate information over timesteps**.

2.2. Going Sequential - From Static Object Recognition to Dynamic Tracking

To alleviate the problems of the object recognition systems from section 2.1, new approaches have been found which still use these algorithms as backbone, but also have an additional component which allows them to preserve certain information from previous timesteps. Such a system has been presented by Jiang et al. in "Multiobject Tracking in Videos Based on LSTM and Deep Reinforcement Learning" [15]. Their approach uses a YOLO backbone to find all objects present in the image. Each of these objects is then tracked by a separate, recurrent reinforcement learning (RL)-controlled network that can move and scale the bounding box so that it follows the target object in the movie as more frames are processed. This gives the advantage of retaining object instances across frames, even if an object is temporarily occluded. Using a YOLO backbone helps the system to overcome one of the fundamental problems when processing videos using neural networks, which is the inherently high dimensionality of videos with $\#\text{frames} \times \text{height} \times \text{width}$ and therefore the high computational power required. Focusing attention on extracted regions can ease this problem as it removes a large part of this dimensionality. While the approach is capable of tracking objects in screen space, it still misses a real understanding of 3D space.

This shortcoming is addressed by Marban et al. [16]. In their work, the authors show that it is possible to infer the position and velocity of surgical instruments in 3D space from monocular video sequences. To achieve this, they use a network that first extracts features from the frames using a CNN, followed by a recurrent neural network (RNN) for spatio-temporal reasoning. We believe that these capabilities are crucial to understanding 3D space when attempting to observe a scene from multiple angles and aggregating information while doing so.

An alternative approach used for object detection in videos is 3D convolutional neural networks (3D CNN) as in [17, 18]. This approach exploits the fact that videos are basically stacks of images which can be processed by 3D filters and thereby extract not only features in 2D images but also changes along the time dimension. A problem of systems like this is that the sequence length needs to be known at the beginning of the training because variable length inputs are not possible. Also, all data needs to be available to start the inference process, which makes the approach less attractive for real-time processing.

Generally, we believe that approaches based on RNNs like the one discussed by Marban et al. [16] are better suited to solve the shortcomings presented in section 1.2, since the difference in the number of frames required to observe and understand a scene is potentially large. This approach is capable of understanding 3D space, which we see as a must when striving for scene-understanding capabilities.

2.3. The Internal Scene Representation of the Human Mind

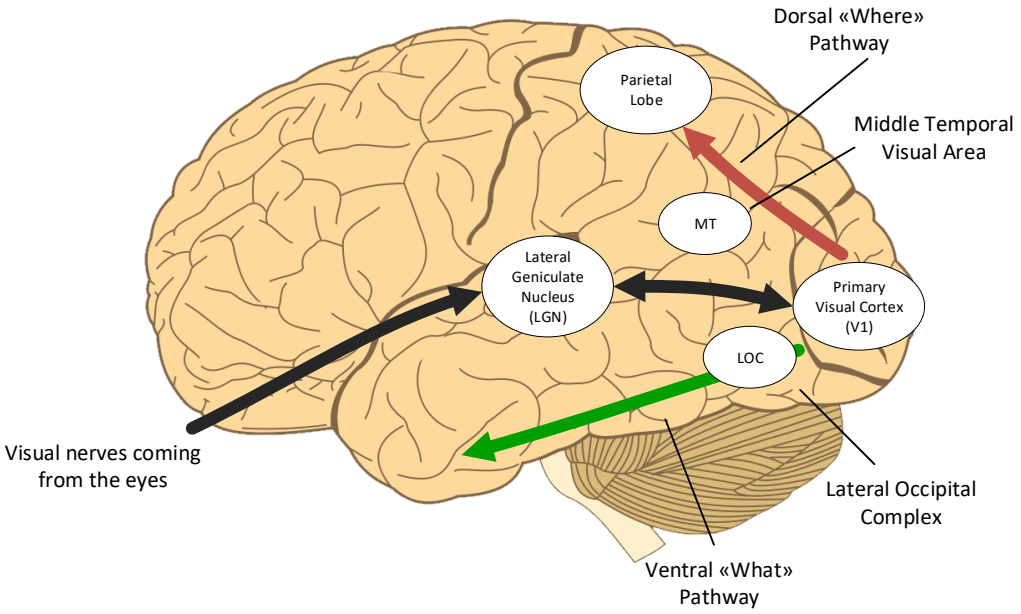


Figure 2.1.: Simplified visualization of the human visual pathways. Perceived information is passing through the lateral geniculate nuclei into the primary visual cortex. From there, the two dorsal and the ventral pathway extract more specialized information. Brain image source: [19], other sources: [20, 21, 22].

In an attempt to improve computer-driven scene understanding, we believe it is vital to analyze what tools the human brain employs to solve the task. While deep learning is inspired by the way real neurons fire together, artificial neural networks and real neural structures still inherit major differences (such as the proposed lack of machine learning-like back-propagation in the brain [23]). With large parts of the brain still unknown, we are not able to directly copy the architecture. This lack of understanding is mainly due to the immense complexity of the human brain. However, higher level similarities in the

way neural networks and real brains work can still be found. This has been shown for example in the work of Cichy et al. [24], which compares neural activation in the human brain with the activation in a neural network in a task of identifying objects. By using functional magnetic resonance imaging (fMRI) and millisecond resolved magnetoencephalography (MEG) to analyze active regions in the brain, both a spatial and a time correspondence between the observed neural regions and the layers of the artificial neural network could be shown. Furthermore, it was possible to identify connections in the neural network which corresponded to the two visual pathways whose proposed existence is known as two-stream-hypothesis. As a popular model for neural processing of vision, this hypothesis suggests that two separate streams (or pathways) of the brain are involved in processing visual information: The *dorsal* stream, leading from the primary visual cortex (V1) through the middle temporal visual area (MT) into the parietal lobe, is responsible for localizing sighted objects in space. Usually, human grasping actions are based on signals from the dorsal stream, which is why it is often described as "vision for action". The *ventral* stream on the other hand leads from the visual cortex, or more precisely from the lateral occipital complex (LOC), towards the temporal lobe. This stream can be seen as "vision for perception", respectively *what* is an object [20]. The ventral stream is built in a hierarchical way, which allows to classify a large number of different shapes including previously unseen objects. Since its first discovery (notably in hamster brains and not in human brains) in 1969 [25], the two-stream-hypothesis has been highly influential for large parts of research on human visual processing. While recent research challenges the belief, that the two streams are completely functionally independent [26], it is generally accepted that at least a rough physical separation of tasks takes place.

One interesting property of the dorsal stream is the quicker decay of memory than within the ventral stream[27]. This makes sense, as object classes need to be learned and remembered but object positions are usually only relevant for a limited time. When following the ventral/"what" stream, it is also important to note that the LOC contains no information regarding what class an object belongs to, but merely what shape the specific object has [21, 28]. This means, that the ventral stream can be seen as a hierarchical construct that contains different levels of abstractions when going from the V1 to the LOC and further into the areas that actually use the extracted information. This is similar to the way a CNN extracts features from an image by applying multiple levels of filters, followed by fully connected layers for reasoning. Potentially useful for computational scene understanding might be the concept that the two-way connection between the lateral geniculate nucleus (LGN) and the V1 creates a compressed representation of the scene [22], which is accessed by different pathways that need to solve different tasks. These pathways then extract more and more specific information from this representation, which can then be used for reasoning.

With the human brain being extraordinarily successful with object classification and localization, it does not surprise that attempts have been made to utilize this architecture by building a two-stream neural network for object detection and classification. Jang et al. [29] show that this approach can be highly successful. In their work, they present a solution for robotic object detection and grasping which is based on a ventral and a dorsal network. As in the human brain, the ventral stream recognizes the class of an object, while the dorsal stream evaluates geometric relationships/objects positions to successfully execute grasps. The system is reusing a semi-supervised grasping approach from previous work [30]. The authors evaluate their work using eight different setups on a cluttered scene and compare it to three single-pathway baselines. Jang et al. show that using two separate streams can lead to an improvement of about three times compared to their baseline from mentioned previous work. However, the approach uses only independent frames and has therefore no capabilities of combining information from different angles.

Ebrahimpour et al. present their ventral dorsal neural network (VDNN) in [31]. The network is split into two separate streams, where the ventral stream detects interesting parts of the current image and masks out all non-essential parts. The dorsal stream subsequently attempts to recognize the object that is still present on the partially masked-out image. While this is inspired by the two-stream-hypothesis, it deviates heavily from the human archetype by applying the two streams in a sequential way. The object recognition is executed on static frames, while one of the qualities of having separate pathways would be to have different memory capabilities as discussed above.

We think that this discussed difference in memory-span is essential when building a system to observe a scene. Inferring the class of an object requires long-term memory, because the class needs to be

known in advance. This applies also for classifying objects never seen before because not knowing an object still requires checking it against all known classes to determine that it does not fit into any of the existing categories. Transferred to a neural network, the long-term memory of the ventral stream could therefore be seen as the learned weights, while the dorsal information is rather stored in the accumulated state.

2.4. Scene Understanding for Computers

In the work "Attend, Infer, Repeat: Fast Scene Understanding with Generative Models" [32] by Eslami et al., a system is presented which is actually capable of observing and reconstructing a scene using a type of variational autoencoder architecture. The authors demonstrate capabilities of their system to understand both 2D and 3D scenes that include counting, localizing and classifying objects. The encoder-part uses a RNN which outputs a description of one found object on each step. This description is used by the decoder to recreate the scene. This works for 2D scenes as well as for simple 3D scenes, but it is important to note that the system needs fully specified 3D models in advance to perform the reconstruction, rendering and training. We see that as a potential problem for domains which do not allow for previously specified models, but have, for example, more unpredictable and organic shapes. To obtain a gradient from the rendered image, the authors use finite-differencing as an approximation. While this work operates on single independent images, an extension exists which is capable of taking image sequences into account [33]. However, its focus is less on acquiring a more complete image of a scene and more on dealing with moving objects in 2D space.

A kind of scene understanding that does not need pre-specified 3D models is presented in the follow-up work "Neural scene representation and rendering" [34] also by Eslami et al.. With only a few 2D input images and their corresponding viewpoints in an unknown, simple 3D scene, the presented Generative Query Network (GQN) is capable of synthesizing renderings from previously unseen viewpoints within the scene. By doing so, the authors show that neural networks are capable of acquiring an internal model of observed geometry. With an increasing number of input images, the quality of the synthesized output image improves and its uncertainty decreases. The approach uses two networks, where the first network aggregates the data of the input images into a compressed scene representation. The second network produces a rendering based on this scene representation and an additional input, which is the desired camera position for rendering by stochastic sampling from the scene representation vector. Eslami et al. also successfully use the compressed representation as input vector for RL-based robotic grasping. By doing so, they achieve a four times higher sample efficiency than when relying on raw pixel input. The resulting renderings of the system are very impressive, but they come at the price of requiring a lot of training data. For each of the four presented experiments, at least 2 million training scenes have been used. Collecting comparable numbers of real-world data would hardly be achievable in a reasonable time.

When comparing this highly impressive result to the process of human vision, one can argue however, that the process is simplified by providing the coordinates from where the additional input image was taken. The human visual process on the other hand needs to integrate each new perception with respect to the previous one into the acquired scene representation. Gori [10] even claims that animals would not be capable of seeing in such a world of shuffled frames. For this reason, we think that a human-like scene understanding system should be capable of inferring its own position and thus the translation and rotation between consecutive frames.

2.5. Looking at it from a Different Angle - On Active Vision and Visual Attention

While a large number of use cases in computer vision assume a stationary camera or given image sequences [15, 17, 18, 35], the area of *active* vision deals with cases where the system is embodied in an active agent and can manipulate the viewpoint in order to perceive more complete information of the scene [36]. In many cases where the initial camera position does not capture all the information available in the scene this is highly useful for a large number of tasks [36]. It is natural for humans to catch glances from as many different angles as necessary or available in order to obtain the most complete mental representation possible of a scene or an object. Of course, this can also work for robots. Cheng et al. [37] for example, apply RL-based mechanisms for a combined control of a robotic gripper and an active camera. By using actor-critic algorithms [38], the authors train a system to pick up a red cube even in the presence of distractor objects which sometimes occlude the direct line of sight to the target. By training to move the camera and pick up the cube, the system implicitly learns to understand how relevant a certain feature in the scene is. This process of focusing on important things while discarding less relevant input information is also known as visual attention [39]. We think that visual attention and active vision are closely related because attention can be focussed on areas which are not known well enough. This is evident in human sight, which deals with areas of missing information by using both eye movement and change of viewpoint as forms of active vision.

Traditionally, attention in computer vision and machine learning has been divided into *soft attention* and *hard attention* [40]. Soft attention limits the influence each component (e.g. pixel) of the input can have, hard attention crops out only a small part, similar to the small foveal region of human vision. While soft attention mechanisms help object detection algorithms to extract important features, it does not reduce the computational power required to detect objects in an image. It has been suggested that the advantage of the very limited human foveal region lies exactly in this reduced cost, which is also favored by the fact, that the limited region requires the brain to process perceived objects in a sequential way [10]. In cases where computing power is limited, hard attention mechanisms can therefore help. By focusing on changing areas with a cropping window, the perception process can be sequentialized. This removes the need for parallel computation perception as it required when processing whole images at once.

A successful application of hard attention can be found in "Recurrent Models of Visual Attention" [41] by Mnih et al.. In their work, the authors perform digit classification on the MNIST database [42]. Using a RNN, the focused window is set to different points on the input image. Each input consists of k square images, where each image has twice the width and height of the previous image. All images are then down-sampled to the resolution of the inner-most image, which can be considered as the equivalent of the human foveal image. The outer images can be seen as peripheral vision. To avoid the need for a sliding window that scans the *whole* picture, the approach uses RL to find the next region on which the moving window should focus.

As mentioned above, we believe that the process of visual attention and active vision are deeply intertwined. In our opinion, it is likely that one needs the other to successfully minimize unknown information when observing a scene. In chapter 4, we perform experiments using active vision and discuss the influence that viewpoint-altering actions can have on the quality of the scene observation.

2.6. Summary

While static object recognition algorithms remain very popular in computer vision [8, 9], much of the research is already focused on how to teach systems a better understanding of space, time or occlusion [15, 16, 43]. The human brain is able to solve these challenges with great ease. That is why we analyzed what components could potentially be copied for a system like ours. The literature describes two different visual pathways which both operate on the information aggregated in the V1. We think that this aggregate-then-specialize approach could potentially be very helpful when attempting to understand a scene. While the two-stream idea has been applied in static object recognition, all

work that we have seen attempts to completely separate the two streams. However, we are critical of applying such a strict separation for scene understanding, because we believe, that localization and classification are inherently connected and need to operate on the same low-level signals. While early literature assumed a strict separation[20], recent reviews are unsure about the independence of the two pathways [26]. Instead, we think that the system would need to learn such a separation by itself, like it is the case in the presented work of Cichy et al.[24].

In terms of existing scene understanding solution, especially the GQN [34] presented by Eslami et al. is able to acquire a highly detailed representation of the scene. However, we think that a system should be capable of inferring its own viewpoint, which is not the case in this work. The benefit of being able to synthesize renderings comes at the cost of 2 million required training samples. With most humans also not being able to synthesize renderings, it might not be a critical task to solve a system like ours, where the main objective is to understand the scene. We think a solution that would achieve this objective using a reasonable number of training samples could already be a real benefit to a large number of tasks. We build a system that solves these shortcomings in chapter 3.

Lastly, we discussed the topic of active vision and its connection to visual attention. By approaching viewpoints that are more beneficial, a system could potentially reduce the number of required camera angles while assuring that the images captured are of great quality. However, having non-uniform camera movement would require, that the system can integrate new images into the current representation. In chapter 4, we evaluate whether it is feasible to extend our approach from chapter 3 with such capabilities. These experiments show that active vision is highly beneficial for the systems performance.

3. Scene-Understanding Network

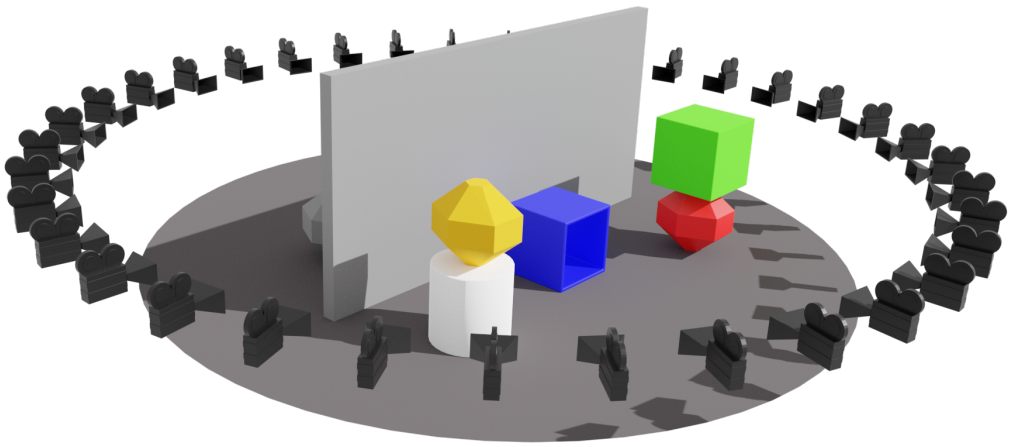


Figure 3.1.: Visualization of an example scene with occlusion, including the designated camera positions, which are used for observing the scene.

In the following sections, we attempt to combine the points discussed under chapter 2 into a single system that processes video input inspired by the functioning of the human brain. As a simple use case, we use scenes with randomly placed objects as visible in Figure 3.1. Each object can have one of 5 different shapes and 7 color options. After processing sequential perceptions from such scenes, the system should be able to answer questions about the scene, such as "where is the red cone?" or "is there any blue cylinder in the scene?" (for reason of simplicity, we encode these questions in a one-hot form which is easy to understand for neural networks). For these different types of questions, we want to have separate sub-networks that are able to extract the desired information from the accumulated scene state which we call *scene embedding vector*.

As we do not find any public benchmark which includes the proposed capabilities, we do not perform direct comparisons to approaches from other work. Instead, we compare our recurrent approaches against a network that relies solely on processing independent pictures and cannot aggregate information over multiple timesteps. We structure our training data in such a way that multiple angles are required to fully capture all available information in the scene due to occlusion. In theory, this means that only a system which can combine information from different viewpoints should be able to successfully solve the task.

The goal of the system is therefore to show whether it is possible to have a system that does not directly search for a trained class of objects, but builds an internal representation of the scene which can later be queried with questions like those described above. Such a system would have the advantage that it would primarily extract important features of the scene which are more likely to be relevant later, even though the network does not know beforehand what question will be asked.

3.1. A Computer Vision System Like the Human Mind

The human visual system is performing extraordinarily well for scene understanding. It is therefore an obvious step to copy the concepts that make the brain so successful (see discussion in section 2.3). With the visual input passing through LGN into the V1, one can assume that visual information is accumulated in this region [22]. For a human-like computer vision system it might therefore be advisable to have a similar component which aggregates information and provides it to different pathways. Instead of the ventral and the dorsal pathways present in human, we introduce different sub-networks which we call "streams". They should be able to perform specialized tasks with input from the aggregated scene embedding vector. These tasks include answering questions like the following:

What objects are present in the scene? This question includes being aware how many objects are present in the scene. The number and the combination of object shapes and colors is different from scene to scene, which is an additional challenge when answering this question.

Where is an object with a given color and shape? Given color and shape, the system should find an object and output its location in relative coordinates as described below.

What color and shape does an object at a certain position in space have? This question can be seen as the inverse of the previous one. Given only a relative position from the camera position projected to the ground plane, the goal is to evaluate color and shape of the object.

Is there any object stacked above/below an object with a given color and shape? As described in section 3.2, it is possible for two objects to be stacked on top of each other. This type of question should be able to evaluate if such a relation between two objects exists.

With humans not using a global coordinate system for object localization, it would be unrealistic to use such in a task like ours. Instead, we rely on a self-centered relative coordinate system. Its origin is located at the camera position mapped to the floor and it is facing in the same direction as the camera (along the Z-axis). While this approach is not based on scientific publications, we think it is intuitively close to the way human would localize objects.

3.2. Setup and Network Architecture

We skip the step of learning human-like depth perception by directly relying on RGB-D-input image sequences, with each image having a resolution of $4 \times 128 \times 128$ pixels. We perform a circular camera trajectory with a radius of 1.2 meters around a randomly arranged group of primitive objects and capture 36 images, all with the camera pointed towards the center of the group. To simplify matters, we assign a unique combination of color and shape to each placed object, so it can be clearly identified during the training period. We place some objects on top of each other (but not more than two objects). It is important to note that we do not provide any information regarding the camera position to the system. Instead, we solely rely on the system to infer its own position based on previously seen images. To evaluate the system, we use two different scene setups and three different network architectures:

No occlusion: We randomly place four to ten objects in the scene that need to be identified, but we do not place any occlusion objects in the scene. On this setup, both algorithms that operate on static as well as on sequential images should be able to perform well.

With occlusion: We also place four to ten objects randomly in the scene. As an additional difficulty, we add an obstacle in the middle of the scene that occludes parts of the scene when viewed from certain angles. In this environment, sequential approaches should theoretically outperform approaches that just rely on single images.

We use these sequences of captured images from these different scene setups as an input to three different architectures of neural networks. All proposed network architectures should fulfill the tasks described in section 3.1. We use the same output streams for all three network setups (see section 3.3).

1. Single image CNN (Baseline): This setup consists of a simple CNN followed by multiple fully connected layers that attempt to create a representation of the scene from a single image. The scene embedding vector is then passed to the different streams, which have different tasks and receive different additional inputs. No information is aggregated over multiple frames. While highly simplified, this baseline should represent currently popular object recognition approaches without aggregation capabilities. With this baseline, we want to assure that sequential information acquisition does actually help the process.

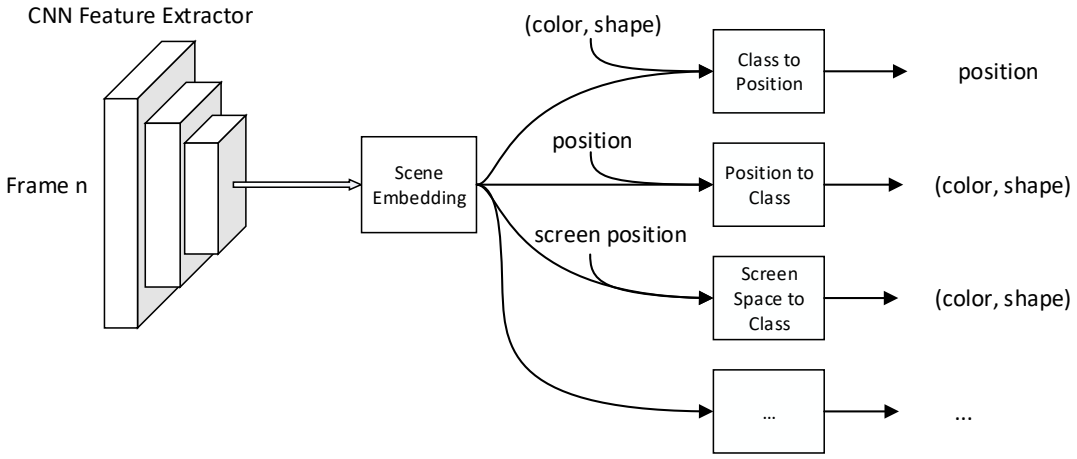


Figure 3.2.: Network architecture of baseline approach that does not accumulate information over time.

2. CNN + RNN: For this network, we sequentially pass images of the captured sequence to the network. Each frame gets processed by a CNN first, followed by a Long Short-Term Memory (LSTM)-layer [44], which can aggregate the perceived scene information. A number of fully connected layers follow this recurrent state. This is a standard approach for a large number of successful video classification algorithms. The fully connected layers are followed by separate streams.

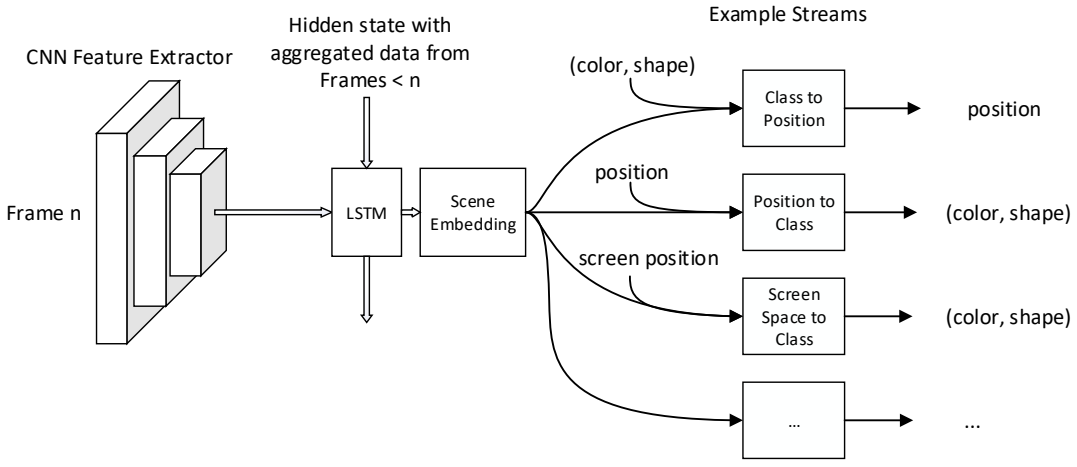


Figure 3.3.: Network architecture of approach 2. CNN + RNN with a limited number of example streams.

3. Convolutional LSTM (ConvLSTM): This setup uses a ConvLSTM, a combination of 2D convolutional layer and LSTM [45] to process spatiotemporal correlations of the input frames. Like the other two setups it is followed by a series of fully connected layers that produce the scene embedding. This vector is then again used by the different streams.

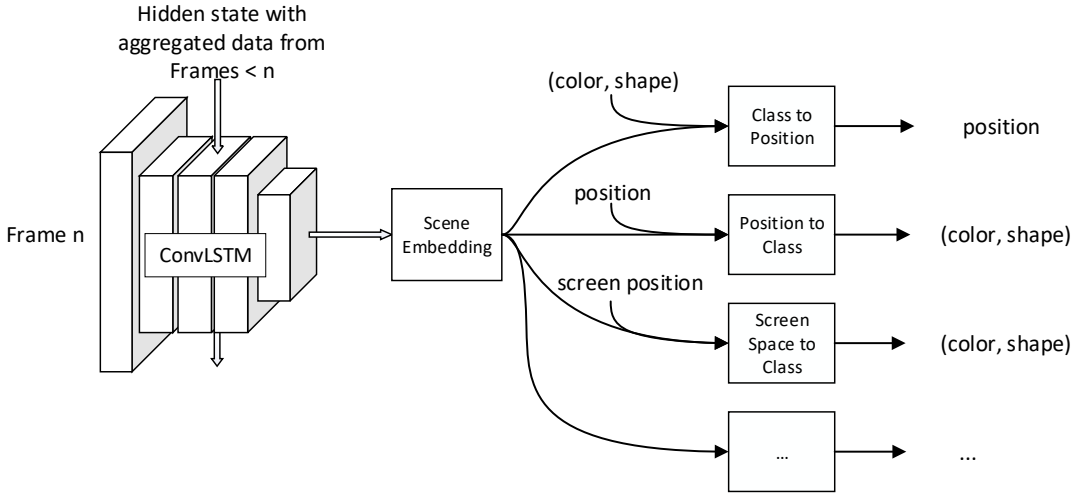


Figure 3.4.: Network architecture of approach 3. ConvLSTM with a limited number of example streams.

While the baseline approach using just a CNN is not even in theory capable of accumulating enough information of the scene to answer all questions posed in section 3.1, it is instead much easier to train a neural network which does not use image sequences but independent images only as input, due to its lower dimensionality. Each architecture setup is composed from a number of sub-networks for better reusability. The *CNN + RNN* version for example uses the same feature extraction layer as the baseline version and all three setups rely on the same number of fully connected layers for creating the scene embedding vector. The backpropagation of error is performed end-to-end through all different sub-networks. The streams that output information regarding a single object are evaluated multiple times on each timestep using randomly selected objects and their loss is accumulated. The enumerating stream which is designed to output all objects is evaluated once each frame.

3.3. Proposed Output Streams

For each network, we use the same streams as outputs. We define the class as a combination of color and shape. In cases where the object identity is not clearly defined by the class, it would be logical to convert all streams to become dynamically unrolled (compare Enumerating stream). For simplicity reason however, we limit ourselves to unique color-shape combinations.

- Class to Position: The stream needs to evaluate the relative position of the object, given the accumulated scene embedding vector and the objects shape and color. The output consists of 3 floating point numbers representing the 3D coordinates in a system relative to the base of the camera.
- Position to Class: Given the accumulated state vector and the objects position, the stream should evaluate the objects shape and color. For both of them, the stream outputs a softmax probability distribution.
- Screen Space Coordinates to Class: This stream should evaluate the objects shape and color, given the screen space coordinates of the object for the current frame. The stream outputs a softmax probability distribution for both properties.

- Has Object Below or Above: Given the accumulated state vector and the objects shape and color, the stream should determine if the object stands alone or has another object below or above it. The stream outputs a softmax probability distribution over the options "has something below","has something above" or "stands alone".
- Neighboring Class: The stream should evaluate the color and shape of the object below or above the target object as softmax probability distributions. For both of these distributions, one option is added for the case that the object does not have an adjacent object but stands alone.
- Enumerating: This stream returns a sequence that should include all objects. We use a recurrent network with a multi-head output. One of the heads represents a flag that tells whether there are more objects to unroll. The other heads return position, color and shape of the detected objects. With no naturally given order to the objects, we let the network decide which object to return first. To train this system, we use the output of the position-head to find the ground-truth object that is closest in space. Subsequently, we can train all streams with this selected ground-truth as target. To find the closest object, there are two feasible solutions:

  1. Closest first: We keep a set of remaining ground truth objects and always select the closest remaining object for the next prediction. This object then gets removed from the remaining objects.
  2. Linear Sum Assignment: We pair the predicted positions and the actual object positions in a way that minimizes the overall distance. For this purpose we solve a linear sum assignment optimization problem with the so-called Hungarian method [46].

  We briefly compare the performance of the two approaches in section 3.6.

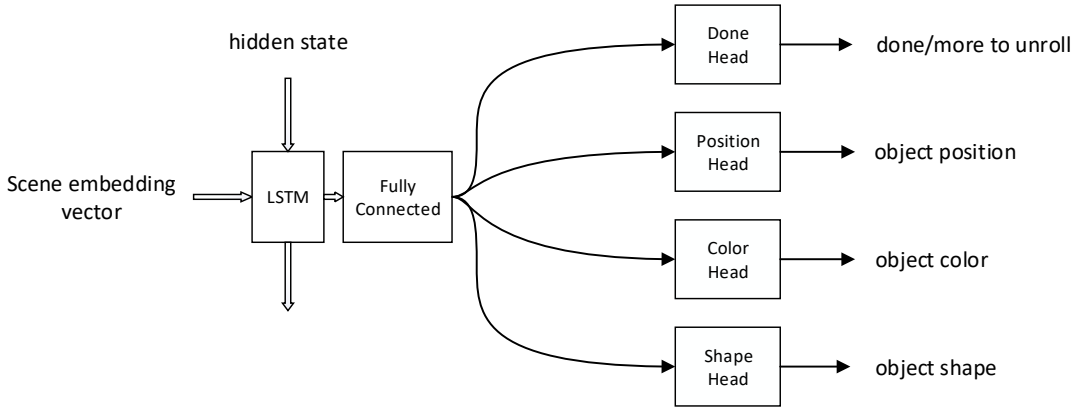


Figure 3.5.: Enumerating stream architecture. The LSTM network gets unrolled until the Done-Head returns *done*

3.4. Training Data Collection

To create large amounts of training data, we pre-generate all image sequences synthetically using the 3D rendering software Blender [47]. This enables us to acquire the corresponding ground truth data, which can be used for training. However, we do not save specific labels but only information regarding the scene in general, such as object positions, colors, shapes, relations (is above/below), etc. This allows us to query the same input sequence on a large number of different questions and therefore force the system to acquire a higher order scene understanding than just simple input-to-output mapping. It is important to note, that the whole data generation process is performed before training. As we are using predefined camera positions, all possible viewpoints are pre-rendered and the camera trajectory is defined by sequentially selecting images from this set. While no viewpoint-pose information gets passed to the network, we still save the transformation matrix of the camera base, so all global object positions can be transformed to relative positions as discussed in section 3.1.

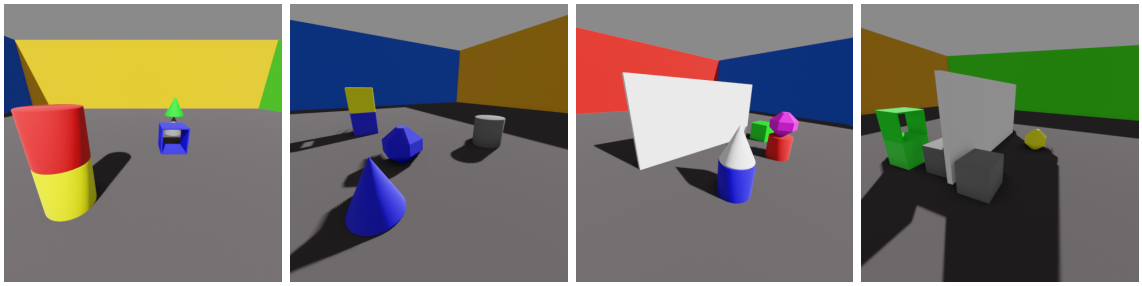


Figure 3.6.: Two example scenes *with* obstacles (right) and two *without*.

We acknowledge that synthetic data does not reflect real camera data in regards of noise or different lighting conditions. However we are confident that the approaches used can be transferred to real-world data if sufficient labeled data (see discussion in section 5.4) are available, as has been shown in different other publications [48, 49, 50].

For the generated scenes, we attempt to reach a visual level that includes many real-world properties such as shadows, specular highlights, etc., but is still simple enough to make sure that potential shortcomings of the algorithm cannot be accounted to scenes that are too complicated or to noisy. We also include uniquely colored walls to enable the algorithm to internally assign global positions to the objects (e.g. "The yellow cylinder is located towards the corner of the red and green wall"). Due to time constraints, we do not evaluate the influence of these colored walls as part of this work but leave this question open to follow-up research. To add additional variance to the input images, the camera gets randomly rotated for each scene along the horizontal axis with an angle between $+12.5°$ and $-12.5°$. As a consequence, this rotation needs to be considered when inferring positions of objects or transforming stored object positions on frame changes.

3.5. Implementation and Stream Training

All architectures are implemented in Python 3.7 and use PyTorch as machine learning framework [51]. To train the system, we use a combination of $20,000$ scenes with and $20,000$ scenes without occlusion. We let the system draw a random sample from this set for each batch. The used batch size varies for each architecture and is determined by the memory requirements of the approach. For the baseline version, we use 128, the CNN + RNN version uses 32 and the ConvLSTM version 64 elements per batch. The learning rate is the same for all approaches at 0.0001. We did not use any form of hyperparameter search, but relied only on parameters that seemed to provide a stable training performance with a few test runs. It can therefore be assumed that other parameter combinations exist which would improve the training performance even further. All used network architectures are listed in Appendix A.

We train the CNN + RNN and the ConvLSTM with about $600,000$ gradient updates each. The baseline version learns much faster but stagnates at around $200,000$ gradient updates. To improve training performance, we split the system into a data-loading and a training part. These two components run in separate processes and communicate via multiprocessing queue to increase performance with the help of a producer-consumer pattern. The data-loading component loads ZIP-files created by Blender. They include the 36 rendered images and a JSON-file with a description of the scene. These ZIP-files are loaded, unpacked and passed into the queue, from where the training-component can pick them up to use them for training.

All training is performed on a NVIDIA Titan Xp Graphics Processing Unit (GPU) with 12 GB VRAM and uses CUDA [52] to speed up the process.

3.6. Stream Evaluations with Different Architectures

We evaluate all networks on scenes with and without occlusion. For both setups, we use 1,000 test scenes each, which have *never been seen on training time*. We only list a subset of all stream evaluations here. The evaluation of remaining streams can be found in the Appendix A. For evaluation of the enumeration stream, we use the following metric (which we call success rate):

$$S = \frac{\#\text{correctly identified}}{\#\text{correctly identified} + \#\text{incorrectly identified}}$$

As "correctly identified" we only count objects which are present in the scene with the matching color and shape. As incorrectly classified objects we count objects which are duplicate, do not exist at all or have not been found by the network. This is similar to accuracy, with the difference that the number of correctly and incorrectly classified objects do not need to add up to the number of ground truth objects. In the Table 3.1, we evaluate a subset of the streams presented in section 3.3 on the evaluation scenes without occlusion.

Stream	**Position to Class**	**Neighboring Class**	**Class to Position**	**Enumeration**
Metric	Accuracy	Accuracy	L2-Error	Success rate∗
Baseline (CNN)	0.81	0.82	0.106	0.71
CNN + RNN	0.94	0.87	0.093	0.89
ConvLSTM	0.83	0.76	0.119	0.71

Table 3.1.: Evaluation of selected streams using *non-occluded* scenes.
∗The success rate is calculated as described above in section 3.6

As visible in Table 3.1, the CNN + RNN approach outperforms both the baseline and the ConvLSTM version remarkably for all streams. While it could be assumed that the baseline would not perform as well as CNN + RNN due to a lack of accumulating capability, the mediocre performance of the ConvLSTM version is relatively surprising. We are unsure of the cause and cannot recognize a specific pattern in which the ConvLSTM is under-performing. However, we assume that the fewer convolutional layers before the ConvLSTM do not allow for such a good feature extraction as this is the case in the CNN + RNN version. This recurrent layer therefore needs to operate on less abstracted information, which could prove difficult. CNN + RNN on the other hand, does exceptionally well, and if it struggles, then mainly in cases where the two shapes "Cube" and "Cube-Hollow" are present, which do look similar by nature. The following Table 3.2 lists the results of evaluating scenes *with* occlusion.

Stream	**Position to Class**	**Neighboring Class**	**Class to Position**	**Enumeration**
Metric	Accuracy	Accuracy	L2-Error	Success rate∗
Baseline (CNN)	0.54	0.63	0.207	0.43
CNN + RNN	0.92	0.86	0.11	0.85
ConvLSTM	0.69	0.70	0.18	0.60

Table 3.2.: Evaluation of selected streams using scenes with *occlusion*.
∗The success rate is calculated as described above in section 3.6

CNN + RNN also outperforms the other versions for scenes with occlusion. Even though the problem is much more difficult, this version reaches performance measures similar to scenes without occlusion. While the ConvLSTM is still not reaching this performance, it at least clearly outperforms the baseline. It is capable of accumulating information, which was not clear after evaluation in scenes without occlusion.

While the *Class to Position* stream cannot really be compared to the streams that output a probability distribution, we do need to acknowledge, that this task seems particularly hard. With an average object size of 0.2 meters, the best performing architecture still makes an average error of 0.11 meters, which would still need to be improved for applications in robotic grasping.

As described in section 3.2, the process of pairing predicted with ground-truth objects on training the *enumerating* stream can be solved with the "closest-first" or the "linear sum assignment" approach. To evaluate which one is better, we train the system with each of the these algorithms and find, that using the "linear sum assignment"-approach is slightly outperforming "closest-first". After $200,000$ network upgrades, the "linear sum assignment" is performing approximately 12% better. However, both versions show a stable training performance.

An evaluation with fine tuning a pre-trained ResNet-18 as feature extractor did not lead to any noticeable improvements. We assume that our scenes are generally too simple to actually use advantages of such approaches. For more complex scenes, this needs to be re-evaluated.

3.7. Towards Neural Symbolic AI

While large parts of research in artificial intelligence are currently focused on neural network-based approaches, symbolic AI still excels in areas, where search, reasoning or predictability are important [53, 54]. One obstacle that has prevented symbolic AI from widespread use is the difficult connection between real-world (noisy) data and their symbolic representation [55]. A system like ours could be useful in solving this problem, since it would be possible to convert the outputs of the streams to symbols. These could then be used for symbolic AI approaches such as logic based systems or imperative programming. While this makes only limited sense in a scenario like ours, it might be highly beneficial for more complex scenes such as agricultural or medical environments. In the following we show two simple example of how a system like ours could be used. The first pseudo code shows how to extract all red objects that lie below a blue cone:

```
for frame in input_sequence:
  network.perceive(frame)

scene_embedding = network.scene_embedding
objs = get_all_objects(scene_embedding)

target_objects  = []
for obj in objs:
  if has_something_above(scene_embedding, obj) and obj.color == red:
    obj_above = get_obj_above(scene_embedding, obj)
    if obj_above.shape == cone and obj_above.color == blue:
      target_objects.append(obj)

return target_objects:
```

An example which would be closer to a real-life use-case could be the picking of fruits from a fruit plant (e.g. tomatoes). A robotic arm could perform a trajectory around the plant to find all fruits. This could be followed by deciding whether the fruit is ripe already and big enough for harvesting. While this would be possible with other approaches, the presented system closely connects semantic information (is the fruit ripe) with spatial information (where in the 3D space can the fruit be found).

```
for frame in input_sequence:
  network.perceive(frame)

emb = network.scene_embedding
fruits = get_all_fruits(scene_embedding)
for fruit in fruits:
  if is_ripe(emb, fruit) and get_size(emb, fruit) > 0.1:
    pick(fruit)
```

The great advantage such a system is that the trained streams can easily be rearranged if a different use case occurs. The output of the streams can be combined like building blocks and the system does not need to be retrained every time a new task needs to be solved.

4. Improving Scene Understanding with Active Vision

In this chapter, we build on the discussion from section 2.5 and focus on the connection between embodiment and self-determined actions in a scenario like the one used in chapter 3. While most computer vision algorithms simply analyze static images from predetermined perspectives, humans tend to observe unknown objects from a larger angle of viewpoints and attempt to identify an object and its relation to its surroundings. We discuss how the performance of our scene understanding system can be improved by using active vision instead of a static trajectory. For all evaluations we use the CNN + RNN model as it delivers the best performance according to the evaluation of section 3.6.

4.1. The Requirements for an Active Vision System

In this section, we evaluate whether active vision is beneficial for the knowledge acquiring process, or if a system like ours performs better when following the same trajectory every time. By answering this question, we gain a deeper insight into the approach the system is taking when confronted with new frames. Two possible ways of working come to mind:

1) The system learns to integrate new images into the existing scene embedding by learning a fixed transformation: Given the input RGB-D frames f_i and f_{i+1} the system assumes a static transformation matrix M which can be learned. Accumulating information would now only require a transformation by M of all up to point i acquired geometric information plus the additional information acquired at timestep $i+1$.

2) The system is capable of dynamically embedding the next image into the existing scene embedding, independently of the change of view point: If the system has acquired this capability, it would be able to determine the transformation matrix M dynamically for each new frame instead of relying on some static transformation. This would be preferable over the first option because it would make the system a better fit for non-uniform camera trajectories.

While the training process significantly influences the type of transformation that the system applies, it is still possible that even a system that trains on non-uniform camera trajectories only learns a static transformation matrix. However, such a system would only be able to use a subset of all input frames, namely those that are transformed the same way as the learned transformation. Assuming that the system is actually capable of learning dynamic transformations for the next timestep, it makes sense to train on these non-uniform trajectories to get the network used to a large variety of different changes between possible camera angles.

Having *active* vision would only make sense in a system that can handle such dynamic transformations. If changing the camera viewpoint in non-uniform ways would negatively affect the capabilities of the system, then a system which actively selects these next input-angles would be unlikely to improve the performance. In fact, it would most likely converge to always use the same trajectory that uses the same step size for each frame change. In this case, the overhead of a system with active vision would be greater than its benefits. Luckily, we see in section 4.3 that this is not the case for our system. Instead it is capable of successfully performing dynamic trajectories and accumulate information quicker than with predefined image sequences.

4.2. Active Vision as a Reinforcement Learning Problem

With the task defined, we can now formulate a RL problem with the goal of reducing uncertainty as much as possible. In our case, we define uncertainty as the sum of the loss of all streams.

$$\delta_i = \sum_{n=0}^{S} L_n^i \qquad \text{where} \qquad \begin{aligned} S &= \text{Number of streams} \\ L_n^i &= \text{Loss of stream n at timestep i} \\ \delta_i &= \text{Uncertainty on timestep i} \end{aligned}$$

A reward function in RL is used as a success metric. In this case, it should reflect the fact that less uncertainty is better. We specify the reward as the difference of uncertainty between timestep $i-1$ and timestep i, or in other words as a gain in certainty.

$$\mathcal{R}_i = \delta_{i-1} - \delta_i$$

From the captured trajectories of chapter 3 we now select a random starting point and give the RL algorithm 7 different actions to chose from:

$$U = \{\text{5 left, 2 left, 1 left, stay here, 1 right, 2 right, 5 right}\}$$

For example, if the agent chooses to go 5 right, the next input frame will be the one 5 steps to the right, relative from the current input image. We include the "stay here" option to give the system the opportunity to focus on different aspects of the same input. In our experiment we only allow a maximum input sequence length of 14 frames. We use the scene embedding vector s_i (see section 3.1) as input state for the RL algorithm and use Q-learning to predict the expected reward for taking an action [56]. To achieve this, we add another stream with output size 7 (our action size) for the q-values $q(s, u)$. This function should estimate the cumulative future reward that can be expected if taking the action u given the system is in state s. Thus, the action with the highest q-value u is the most promising. At each step of a training episode, we therefore either select the best action in terms of the q-function or randomly, based on a decaying ε value. This ε is initially 1.0 and gets multiplied with 0.9999 after every episode. We use the resulting reward values from Monte-Carlo roll-outs to update the q-function after every episode. All states s, actions u and rewards $\mathcal{R}$ are recorded during the episode. For each step i we can calculate the cumulative discounted reward using the following formula, which (when correctly learned) should approximate the q-function by using all received rewards up to the terminal timestep T:

$$q(u_i, s_i) \approx \sum_{f=0}^{T-i} \gamma^f \mathcal{R}_{i+f}$$

To update the network, we use the squared difference between the predicted q-value and the Monte-Carlo roll-out reward as loss value. The training process now takes place simultaneously for the supervised and the RL stream(s). The calculated loss of the supervised streams is not only used for their training using backpropagation, but also serves as input for the reward calculation to improve the RL stream. With ongoing training of the supervised streams, this loss changes potentially, which means that the RL algorithm needs to deal with a non-stationary environment. However, this should not be a problem for the algorithm as it does not rely on experience replay [57]. On the downside, this leads to a lower sample efficiency since the possiblity of reusing the seen episodes is missing [57]. Furthermore, the evaluation of training time is more demanding, because it is difficult to say whether the improvement of the supervised streams needs to be accounted to the improving RL policy or the ongoing training of the streams themselves. As a consequence, it is necessary to perform different types of trajectories and compare their success rates.

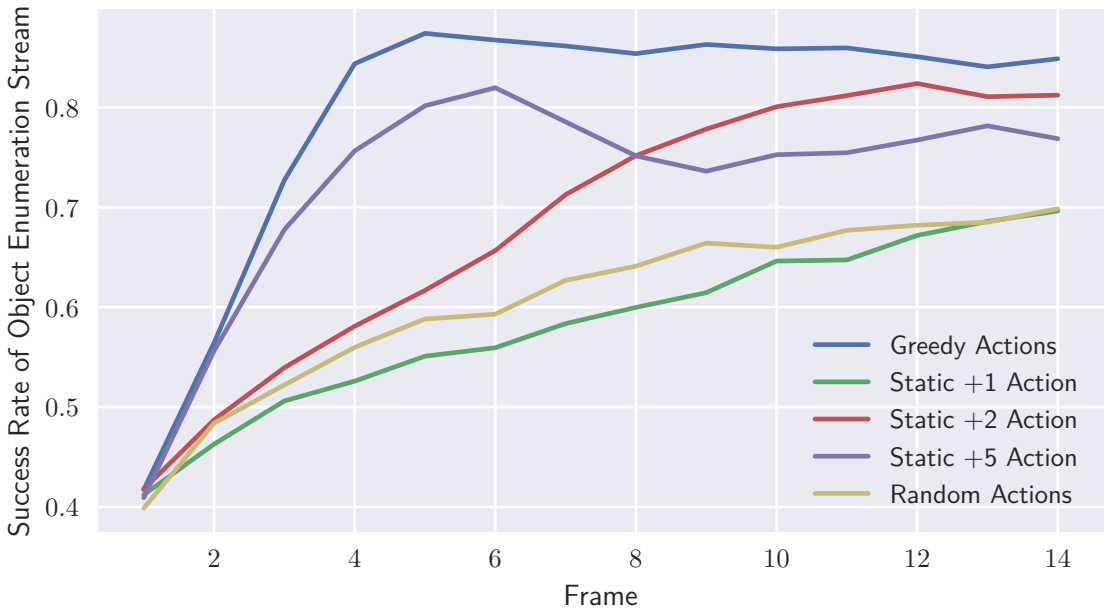


Figure 4.1.: Mean success rate (section 3.6) of the Enumeration stream over 500 episodes, when using 5 different ways of controlling the current viewpoint.

4.3. Evaluation: Active vs. Passive Vision

In this section, we perform evaluations of the approach discussed in section 4.2. For this purpose we perform different types of trajectories using the 36 captured pictures for each scene.

- *Random Actions:* Randomly sampled actions from U.
- *Static Actions:* On every timestep, the camera moves a fixed number of steps to the right.
- *Greedy Actions:* The action with the highest Q-value (estimated by the active vision stream trained by using RL) is used.

All trajectories start at a random point and acquire information over a total of 14 frames. Figure 4.1 shows that the success rate of the system is improved by using active vision, especially during the first few frames. While especially the static actions with $+2$ and $+5$ also perform decently, the active vision allows a head start and reaches the maximum success rate after only 5 frames. Interestingly, the performance of the $+5$-actions drops after the 6th frame. This could be because the $+5$-actions actually perform more than one circle around the scene, which might confuse the system. The most constant performance gain is achieved by the $+2$-action. The active vision approach however is outperforming all of them without relying on a single action for each timestep. From this result it can be concluded that the system is indeed capable of dynamically calculating the transformation matrix given the new frame and does not rely on any statically learned matrix (see section 4.1). Further research would however be required to evaluate whether a similar performance would be possible by using a trajectory that does not have the circular shape that the presented experiment is using. With data from the experiment, one can therefore conclude, that given enough time and a static trajectory, the system can accumulate most of the desired information. When using active vision, however, an even better performance can be achieved with less frames.

While the active vision approach could outperform all other types of camera control in our experiment, we think that the benefit of such a system would be even greater in more complex scenes where the camera can be moved not only on a 1-dimensional trajectory, but in 2 or even 3 dimensions instead.

The current approach does not (and does not have to) care about objects, which could block the path of the camera. However, the reality works differently and we believe that approaches that allow more degrees of freedom should also include ways to prevent collisions with the environment.

One noteworthy finding of the evaluation is the fact that a larger number of frames does not necessarily decrease performance. This becomes clear when comparing the success rate of the $+1$-action with that of the $+2$-action. One could assume that seeing frames with no or very little additional information could prevent the system from focusing on the important features. This does not seem to be the case, since the performance of the $+1$-actions after 12 frames is about the same as the performance of the $+2$-actions after 6 frames.

4.4. Towards Two-Layer Visual Attention

When striving for human-like active vision, simply moving the camera does not do justice to the more complicated real world archetype. Humans are usually able to change their point of view. An equally important component of human visual attention is focusing on an object through eye movement. Due to time constraints, we do not implement a two-layer active vision system in the course of this work. Nonetheless, we can outline how such a system could work. Potentially crucial for the success of such a two-layer active vision would be the chronological order of actions. This means that first the camera viewpoint would need to be moved, followed by one or more steps of a simulated eye movement. While the viewpoint determines from what angle to look at the scene, the eye movement could determine at what object to look in the scene. This ability could be similarly structured to the hard attention mechanism of the discussed work by Mnih et al. [41], where a small portion of the available image gets cropped out and used as actual input. It would need to be empirically evaluated whether there is a performance gain and if so, whether it is worth the additional complexity.

Approaches like the proposed two-layer visual attention would allow to incorporate a cost function to speed up real-world systems. While the delay of changing viewpoints might not be an issue when using synthetic data, the motion planning might be computationally expensive and the execution slow for real robots. This could be taken into account when training the RL-model, so that viewpoints are being preferred that allow to see a high number of objects which then can be focused one-by-one by the simulated eye-movement.

In his work Gori states the hypothesis that the main reason for the limited foveal region of human vision is the reduced need for expensive parallel computation [10]. Recent research even suggests that humans are in fact not capable of any real parallel perception at all [58]. Consequently, implementing such a two-layer visual attention might allow to reduce the required number of parameters to understand a scene, as a large part of the computation that currently happens in parallel would be sequentialized.

5. Discussion and Conclusion

5.1. Concluding Discussion of the Finished System

In this work we could show that is it possible to acquire a deep scene understanding from sequential data with supervised learning. In a simplified use-case such as the presented one, the system can successfully acquire a spatial scene understanding that includes objects, their shape, color and even their relationship with other objects. Although it is limited to the type of scenes the system was trained on, its versatility exceeds by far current mainstream object recognition systems such as ResNet [6]. Unlike encoder-decoder-based scene understanding approaches such as [32], we do not need specified 3D models for training but only a simple scene description that specifies present objects including their positions and the camera poses of the captured images. We could show that our system is capable of sequentially integrating information from new frames into the existing scene embedding vector. This capability is indeed highly remarkable because it includes the achievement of multiple non-trivial steps: 1. The change in camera position relative to the previous frame needs to be evaluated. The system does not receive any information about the camera position or rotation in space, but needs to extract this information based on the difference between the already perceived and the new input frame. 2. This extracted transformation of the camera position must be used to transform all remembered object positions. Some objects might be occluded by the obstacle or other present objects, and the system is not able to perceive these items in the current frame. Nonetheless, it is necessary to transform their remembered positions so that the system is able to return the correct position when queried. The precision of this process however, would need to be further improved when used for robotic grasping, as there is still an average deviation of 0.11 meters (about half the diameter of the object) from the ground truth when using the best performing approach.

While our proposed system does not use separated streams for ventral and dorsal pathways, our information aggregation process is inspired by the quicker decaying dorsal memory and the more persistent ventral memory. This is represented by learned weights versus aggregated temporary information. This architecture seems to work great, especially with respect to the 3 different shortcomings that were the focus of this work (see section 1.2). In the following, we discuss how our system overcomes them:

Shortcoming Nr. 1 | Lack of object permanence in traditional object recognition: The presented approach is capable of identifying objects and remembering them, even when the object is temporarily occluded. The system can update the knowledge about an object upon receiving more or better viewing angles. This is reflected in the evaluation of active vision (see section 4.3), which clearly shows that the system constantly improves when it receives more input frames. We interpret this result to mean that the system is aware of the permanent identity of each object, since it not only adds newly identified objects to the list of discovered entities, but actually expands the collected information of already identified objects, even if an object reappears after occluding.

Shortcoming Nr. 2 | Changing the viewpoint for better results: By training one stream using RL, the system could acquire capabilities of predicting which following viewpoint will be more promising. The scene observation process using active vision greatly outperforms all other types of camera control in terms of speed. It makes it possible to obtain a largely complete image of the scene with only a few frames. Additionally, not only the number of required images is improved, but also the overall observation quality. After only 5 frames, the active vision version can output 87% of all present objects, which is better than any other camera control method.

Shortcoming Nr. 3 | The need to accumulate information over multiple frames: In hindsight, this capability is closely coupled to solving *Shortcoming Nr. 1*. As evident in the evaluation of the streams in section 3.6, the capability to aggregate information can be used most efficiently by the

CNN + RNN approach. Naturally, the consequences of not having these capabilities are smaller in scenes without occlusion. While the baseline solution only required a fraction of the training episodes of the other two approaches, it was not capable of filling missing information from frame to frame. As mentioned in the discussion of the first shortcoming, we think that the understanding of object permanence is one of the key achievements of this work.

With our baseline solution representing traditional CNN-only approaches, we can observe nearly double the performance by using the CNN + RNN method. It is likely that this difference in performance would be smaller in a less constructed case. We are confident, however, that in most scenarios a significant performance boost could be expected compared to single-image approaches.

As discussed in section 4.3, the addition of active camera control does in our case not only improve sample efficiency but also the overall stream performance. We assume that the system was able to learn not only from what angles the scene needs to be observed, but also which camera moves are beneficial for transforming the already acquired information. We think that this capability of active vision would be especially useful in complex scenes where the scene has to be observed from very specific viewpoints to capture every detail.

5.2. From Multi-Task Learning to Higher Sample Efficiency for New Tasks

As a consequence of our multi-stream approach, the scene embedding vector can not only contain information for single tasks, but instead must represent a higher-level scene description. By having such a vector that already contains all aggregated information, we think that far fewer labeled samples are needed to train additional tasks, since the aggregation process is already present and does not need to be learned when acquiring capabilities for a new stream (at least as long as this stream can reuse the information aggregated). To verify this hypothesis, we conduct a final experiment with only 200 annotated training scenes. We use the *CNN + RNN* model described section 3.2 and train a very simple additional stream which needs to determine whether the scene contains any red objects. As we train only a single stream, we do not propagate errors back towards the convolutional and recurrent layers, but only optimize the newly added stream. By doing so, we attempt to keep the number of parameters that require an update as low as possible, both to prevent over-fitting and to simplify the training process. Training a single stream only could damage the aggregated scene embedding vector when going back using other streams. For the evaluation we use another 100 scenes that have never been used for training. Of these generated scenes, 58% contain a red object. We reach an impressive accuracy of 98% for the task. The training process for the stream required a total of 3,000 gradient updates.

While detecting red objects in a scene might not represent the level of skills a real world task would require, it still shows the capabilities of drastically reducing the need for a large number of training samples. We account this to different factors. First, the input dimensionality has been greatly reduced from $\#\text{frames} \times 4 \times 128 \times 128 = \#\text{frames} \times 65,536$ to only $2,048$. Second, by sampling non-uniform trajectories from the available viewpoints, the aggregated scene vector contains more variance, than it would if the trajectory always stayed the same. This acts as a regularization mechanism to the learning process because the number of actually used input (scene embedding) vectors is much higher, than just the number of scenes.

5.3. Towards Robotic Grasp Generation

Since this work was initially inspired by robotic interaction, we would see it as the next step to combine our vision-focused system with grasping approaches. An interesting direction would be to work towards solving benchmarks for robotic interaction such as RLBench [59].

When approaching such tasks, we mainly see two possible paths to take: One relatively simple way would be to keep the perception and the grasping system separate and only use the output of the presented approach as input for a grasp generation algorithm, such as Dex-Net [60]. This would mean that the perception part would identify the target object and then forward its position to the grasp planning mechanism which would plan the grasp and pass it to the robotic arm for execution. As second option it would be possible that the condensed scene representation produced by our system would be beneficial for grasp-generation. We think that a promising approach would be to train two streams for grasping. The first stream could create a large number of possible grasps, while the other one would rate them with likelihood of success. Of course, the system would first need to learn to include the required information in the scene representation, which might be a large leap compared to the information currently present within the trained system. However, with the human mind closely coupling perception and action as part of the dorsal pathway it seems that such a joint approach could work for robotic grasping too.

5.4. Where to Go From Here?

While the discussed approach successfully solves the tasks set, the system is still far from being a real replacement for existing computer vision algorithms in use. On the way to the application of a system like ours to solve tasks in the real world, it would be necessary to solve at least some of the following challenges:

Real world data: In order to use an approach like the one presented in a real-world use-case, it would above all be necessary to transfer the approach to real data or at least more realistic synthetic data. With the goal to jointly improve both scene understanding and active vision, we would inspire future research to use real-time data gathering with simulated environments as for example Isaac Sim [61]. This would allow non-discrete view-positions and thus a potentially better understanding. One aspect that still might not be solvable by using synthetic data is the noise of the depth-channel of the RGB-D data, which is quite prominent for most consumer class RGB-D-camera.

More different object classes/shapes: The presented solution uses only 5 primitive shapes with 7 colors, which most likely does not reflect the conditions of a real world use case. A possible solution could be the use of large-scale 3D object data sets as used for Dex-Net [60]. Closely related to more diverse objects would be the capability to deal with duplicate objects. This could possibly be solved by adding multi-object output to all streams (as demonstrated with the *Enumerating* stream).

Motion: In this work, we did not address the topic of motion, which does play a big roll in the real world. Environments like conveyor belts would be a domain where a scene understanding system like the one presented could prove highly useful. However, an application in such an environment would require previous research with dynamic scenes.

Rotation axis detection: With only primitive objects such as cube, cylinders, etc., we decided not to include rotation information for our streams. However, for a large number of use cases it could be very useful to obtain the rotation of an object, so we would encourage future work to extend our approach to include rotation information.

Grasping/bounding-box detection: For an actual application of robotic grasping, it would be necessary to generate possible grasping positions. We did not address this topic in the course of this work but leave the extension of the presented approach with grasping to further research. For more details see section 5.3.

More degrees of freedom for the camera: We showed in section 4.1 that dynamic camera trajectories can be beneficial for the perception of the scene. For simplicity reason, we only used the frames generated from predetermined positions where the camera can move in a 1-dimensional way (left or right). For further work, we think it would be worthwhile to research whether the approach could be extended to cameras with more degrees of freedom. While a robotic arm like the one used at ZHAW (see section 1.1) might perform such a circular trajectory like the one used, many agents of interest (such as drones) would be more agile and would therefore require more flexible capabilities.

6. Directories

6.1. Bibliography

[1] Giovanni Toffetti and Thomas Michael Bohnert. *Cloud Robotics with ROS*, pages 119–146. Springer International Publishing, Cham, 2020. ISBN 978-3-030-20190-6. doi: 10.1007/978-3-030-20190-6_5. URL https://doi.org/10.1007/978-3-030-20190-6_5.

[2] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel RealSense Stereoscopic Depth Cameras. *arXiv e-prints*, art. arXiv:1705.05548, May 2017.

[3] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017.

[4] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3-d mapping with an rgb-d camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2014.

[5] Marvin M. Chun. Scene perception and memory. In *Cognitive Vision*, volume 42 of *Psychology of Learning and Motivation*, pages 79 – 108. Academic Press, 2003. doi: https://doi.org/10.1016/S0079-7421(03)01003-X. URL http://www.sciencedirect.com/science/article/pii/S007974210301003X.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, art. arXiv:1512.03385, December 2015.

[7] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-prints*, art. arXiv:1409.1556, September 2014.

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *arXiv e-prints*, art. arXiv:1506.02640, June 2015.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[10] Marco Gori. *What's Wrong with Computer Vision?: 8th IAPR TC3 Workshop, ANNPR 2018, Siena, Italy, September 19–21, 2018, Proceedings*, pages 3–16. 08 2018. ISBN 978-3-319-99977-7. doi: 10.1007/978-3-319-99978-4_1.

[11] Christian Merkel, Jens-Max Hopf, and Mircea Ariel Schoenfeld. *How to Perceive Object Permanence in Our Visual Environment: The Multiple Object Tracking Paradigm*, pages 157–176. Springer US, New York, NY, 2020. ISBN 978-1-4939-9948-4. doi: 10.1007/7657_2019_28. URL https://doi.org/10.1007/7657_2019_28.

[12] Renée Baillargeon, Elizabeth S. Spelke, and Stanley Wasserman. Object permanence in five-month-old infants. *Cognition*, 20(3):191 – 208, 1985. ISSN 0010-0277. doi: https://doi.org/10.1016/0010-0277(85)90008-3. URL http://www.sciencedirect.com/science/article/pii/0010027785900083.

[13] N V Kartheek Medathati, Heiko Neumann, Guillaume Masson, and Pierre Kornprobst. Bio-inspired computer vision: Towards a synergistic approach of artificial and biological vision. *Computer Vision and Image Understanding*, 150, 04 2016. doi: 10.1016/j.cviu.2016.04.009.

[14] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26): eaau4984, 2019.

[15] Ming-xin Jiang, Chao Deng, Zhi-geng Pan, Lan-fang Wang, and Xing Sun. Multiobject tracking in videos based on lstm and deep reinforcement learning. *Complexity*, 2018:1–12, 11 2018. doi: 10.1155/2018/4695890.

[16] Arturo Marban, Vignesh Srinivasan, Wojciech Samek, Josep Fernandez, and Alicia Casals. Estimating position & velocity in 3d space from monocular video sequences using a deep neural network. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[17] Rui Hou, Chen Chen, Rahul Sukthankar, and Mubarak Shah. An Efficient 3D CNN for Action/Object Segmentation in Video. *arXiv e-prints*, art. arXiv:1907.08895, July 2019.

[18] Ali Diba, Mohsen Fayyaz, Vivek Sharma, Amir Hossein Karami, Mohammad Mahdi Arzani, Rahman Yousefzadeh, and Luc Van Gool. Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification. *arXiv e-prints*, art. arXiv:1711.08200, November 2017.

[19] Wikimedia Commons/Hugh Guiney. Diagram of human brain, April 2008. URL `https://commons.wikimedia.org/wiki/File:Human-brain.SVG`. [Accessed: June 2nd, 2020].

[20] Melvyn A. Goodale and A.David Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15(1):20 – 25, 1992. ISSN 0166-2236. doi: https://doi.org/10.1016/0166-2236(92)90344-8. URL `http://www.sciencedirect.com/science/article/pii/0166223692903448`.

[21] Eshed Margalit, Manan P. Shah, Bosco S. Tjan, Irving Biederman, Brenton Keller, and Rorry Brenner. The Lateral Occipital Complex shows no net response to object familiarity. *Journal of Vision*, 16(11):3–3, 09 2016. ISSN 1534-7362. doi: 10.1167/16.11.3. URL `https://doi.org/10.1167/16.11.3`.

[22] Werner X. Schneider. Visual-spatial working memory, attention, and scene representation: A neuro-cognitive theory. *Psychological Research*, 62(2):220–236, Jul 1999. ISSN 1430-2772. doi: 10.1007/s004260050052. URL `https://doi.org/10.1007/s004260050052`.

[23] James C.R. Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in Cognitive Sciences*, 23(3):235 – 250, 2019. ISSN 1364-6613. doi: https://doi.org/10.1016/j.tics.2018.12.005. URL `http://www.sciencedirect.com/science/article/pii/S1364661319300129`.

[24] Radoslaw Cichy, Aditya Khosla, Dimitrios Pantazis, Antonio Torralba, and Aude Oliva. Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence. *Scientific Reports*, 6:27755, 06 2016. doi: 10.1038/srep27755.

[25] Gerald E. Schneider. Two visual systems. *Science*, 163(3870):895–902, 1969. ISSN 0036-8075. doi: 10.1126/science.163.3870.895. URL `https://science.sciencemag.org/content/163/3870/895`.

[26] Thomas Schenk and Robert D. McIntosh. Do we have independent visual streams for perception and action? *Cognitive Neuroscience*, 1(1):52–62, 2010. doi: 10.1080/17588920903388950. URL `https://doi.org/10.1080/17588920903388950`. PMID: 24168245.

[27] Steven A. Jax and David A. Rosenbaum. Hand path priming in manual obstacle avoidance: Rapid decay of dorsal stream information. *Neuropsychologia*, 47(6):1573 – 1577, 2009. ISSN 0028-3932. doi: https://doi.org/10.1016/j.neuropsychologia.2008.05.019. URL `http://www.sciencedirect.com/science/article/pii/S0028393208002492`. Perception and Action.

[28] Zoe Kourtzi and Nancy Kanwisher. Representation of perceived object shape by the human lateral occipital complex. *Science*, 293(5534):1506–1509, 2001. ISSN 0036-8075. doi: 10.1126/science.1061133. URL `https://science.sciencemag.org/content/293/5534/1506`.

[29] Eric Jang, Sudheendra Vijayanarasimhan, Peter Pastor, Julian Ibarz, and Sergey Levine. End-to-End Learning of Semantic Grasping. *arXiv e-prints*, art. arXiv:1707.01932, July 2017.

[30] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *arXiv e-prints*, art. arXiv:1603.02199, March 2016.

[31] M. K. Ebrahimpour, J. Li, Y. Yu, J. Reesee, A. Moghtaderi, M. Yang, and D. C. Noelle. Ventral-dorsal neural networks: Object detection via selective attention. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 986–994, 2019.

[32] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, koray kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3225–3233. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6230-attend-infer-repeat-fast-scene-understanding-with-generative-models.pdf`.

[33] Adam R. Kosiorek, Hyunjik Kim, Ingmar Posner, and Yee Whye Teh. Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects. *arXiv e-prints*, art. arXiv:1806.01794, June 2018.

[34] S. M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. ISSN 0036-8075. doi: 10.1126/science.aar6170. URL `https://science.sciencemag.org/content/360/6394/1204`.

[35] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf`.

[36] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1(4):333–356, Jan 1988. ISSN 1573-1405. doi: 10.1007/BF00133571. URL `https://doi.org/10.1007/BF00133571`.

[37] Ricson Cheng, Arpit Agarwal, and Katerina Fragkiadaki. Reinforcement Learning of Active Vision for Manipulating Objects under Occlusions. *arXiv e-prints*, art. arXiv:1811.08067, November 2018.

[38] Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 1008–1014. MIT Press, 2000. URL `http://papers.nips.cc/paper/1786-actor-critic-algorithms.pdf`.

[39] Sabine Kastner and Mark Pinsk. Visual attention as a multilevel selection process. *Cognitive, affective and behavioral neuroscience*, 4:483–500, 01 2005. doi: 10.3758/CABN.4.4.483.

[40] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *arXiv e-prints*, art. arXiv:1502.03044, February 2015.

[41] Volodymyr Mnih, Nicolas Heess, Alex Graves, and koray kavukcuoglu. Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2204–2212. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention.pdf`.

[42] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[43] Martin Rünz and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20, 2018.

[44] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[45] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *arXiv e-prints*, art. arXiv:1506.04214, June 2015.

[46] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi: 10.1002/nav.3800020109. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

[47] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2020. URL http://www.blender.org.

[48] Yunzhi Lin, Chao Tang, Fu-Jen Chu, and Patricio A. Vela. Using Synthetic Data and Deep Networks to Recognize Primitive Shapes for Object Grasping. *arXiv e-prints*, art. arXiv:1909.08508, September 2019.

[49] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4243–4250, 2018.

[50] E. Johns, S. Leutenegger, and A. J. Davison. Deep learning a grasp function for grasping under gripper pose uncertainty. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4461–4468, 2016.

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[52] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, March 2008. ISSN 1542-7730. doi: 10.1145/1365490.1365500. URL https://doi.org/10.1145/1365490.1365500.

[53] JEFFREY SPONSLER, MARK JOHNSTON, GLENN MILLER, ANTHONY KRUEGER, MICHAEL LUCKS, and MARK GIULIANO. *An AI scheduling environment for the Hubble Space Telescope*. doi: 10.2514/6.1991-3703. URL https://arc.aiaa.org/doi/abs/10.2514/6.1991-3703.

[54] Robert Hoehndorf and Núria Queralt-Rosinach. Data science and symbolic ai: Synergies, challenges and opportunities. *Data Science*, 1:27–38, 2017. ISSN 2451-8492. doi: 10.3233/DS-170004. URL https://doi.org/10.3233/DS-170004. 1-2.

[55] Pedro Zuidberg Dos Martires, Nitesh Kumar, Andreas Persson, Amy Loutfi, and Luc De Raedt. Symbolic Learning and Reasoning with Noisy Data for Probabilistic Anchoring. *arXiv e-prints*, art. arXiv:2002.10373, February 2020.

[56] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

[57] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1146–1155. JMLR.org, 2017.

[58] Alex White, John Palmer, and Geoffrey Boynton. Evidence of serial processing in visual word recognition. *Psychological Science*, 29:095679761775189, 05 2018. doi: 10.1177/0956797617751898.

[59] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The Robot Learning Benchmark & Learning Environment. *arXiv e-prints*, art. arXiv:1909.12271, September 2019.

[60] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Kenneth Y. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *ArXiv*, abs/1703.09312, 2017.

[61] NVIDIA Corporation. NVIDIA Isaac Sim, Feb 2020. URL `https://developer.nvidia.com/isaac-sim`. [Accessed: June 2nd, 2020].

[62] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Winter 1989.

[63] Marc D. Binder, Nobutaka Hirokawa, and Uwe Windhorst, editors. *Fovea (Fovea Centralis)*, pages 1626–1627. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-29678-2. doi: 10.1007/978-3-540-29678-2_1835. URL `https://doi.org/10.1007/978-3-540-29678-2_1835`.

[64] Trevor Huff, Navid Mahabadi, and Prasanna Tadi. *Neuroanatomy, Visual Cortex*. StatPearls Publishing, Treasure Island (FL), Oct 2019. URL `https://pubmed.ncbi.nlm.nih.gov/29494110`. 29494110[pmid].

6.2. List of Figures

6.3. List of Tables

6.4. Glossary

Hint: For abbreviations, see 6.5 Abbreviations.

Symbols

3D occlusion Visual obstruction in three-dimensional space. 2

B

Blender Open source 3D creation suite for modelling, simulation and rendering [47]. 18, 19

C

CNN Convolutional neural network is a deep neural network architecture which uses learnable filters (convolutions) to extract features from input matrices (usually images) [62]. CNNs that use backpropagation for training have been proposed by Yann LeCun. 38

ConvLSTM A LSTM which uses convolution operations inside the LSTM cell. Often used for processing of videos or other sequential data which requires feature extraction [45]. 1, 2, 38

CUDA Programming technique from Nvidia to move calculations to the GPU [52]. 19

F

foveal region The area of the human retina which allows for full acuity [63]. 12, 25

G

GQN System by DeepMind which learns to create an internal scene representation and to render this scene from novel viewpoints [34]. 38

J

JSON JavaScript Object Notation, a compact data format for data exchange between applications. 19

L

LOC Lateral occipital complex is a part of the human brain which produces a higher-level object representation from lower-level visual perceptions. This representation is, however, limited to shape representation, semantic information is not present in this area [28]. 38

LSTM Long Short-Term Memory is a recurrent neural network architecture which is able to remember information over short and long periods of time [44]. 38

M

Mask R-CNN Approach for object instance segmentation, which efficiently detects objects and simultaneously generates segmentation masks for every instance [9]. 8

MNIST A database of handwritten digits which is widely used for testing of machine learning algorithms [42]. 12

Monte-Carlo A learning method in RL which learns from completed episodes and estimates the value function (or q-function) [56]. 23

P

PointNet++ State of the art approach for semantic point clouds segmentation [3], presented by Qi et al.. 5

PyTorch Open source machine learning framework for Python [51]. 19

R

Reinforcement learning Area of machine learning, where an agent learns through interaction with an environment [56]. 38

ResNet Residual neural networks are a special kind of neural network which have the ability to skip connections or jump over certain layers. These networks are extraordinarily successful in image classification [6]. 8, 26

RNN Recurrent neural networks are a subcategory of neural networks which allow to memorize certain information from previous timesteps. Prominent representatives of recurrent neural networks are for example LSTMs [44]. 38

S

streams Different sub-networks described in section 3.3. In this work, different streams should be able to answer specific questions regarding the scene embedding vector. I, 15–18, 20, 21, 23, 26–28, 34, 35

Summit XL Steel mobile robot Robotic platform for indoor use made by Robotnik Automation (see Figure 1.1). 5, 7, 34

V

V1 The part of the brain which is responsible for processing visual information. It operates on input from the eyes, which is passed to the primary visual cortex via the lateral geniculate nucleus (LGN). The primary visual cortex is specialized in pattern recognition and admits information to the ventral and the dorsal pathway [64]. 38

VDNN Neural network architecture for object detection via selective attention, which is inspired by the ventral and dorsal stream of the human brain [31]. 38

VGG VGG nets consist out of stacked convolutional layers and perform very well in image localization and classification tasks [7]. 8

Y

YOLO You Only Look Once, an object detection approach which uses a single pass to classify objects and detect bounding boxes around objects [8]. 8

6.5. Abbreviations

Symbols

3D CNN 3D convolutional neural networks. 9

C

CNN convolutional neural network (see CNN in Glossary). II–IV, 6, 8–10, 16, 17, 19, 20, 22, 27, 34

ConvLSTM Convolutional LSTM (see ConvLSTM in Glossary). III, 17, 19, 20, 34

F

fMRI functional magnetic resonance imaging. 10

G

GPU Graphics Processing Unit. 19, 36

GQN Generative Query Network (see GQN in Glossary). 11, 13

I

i.i.d. independent and identically distributed. 6

ICCLab InIT Cloud Computing Lab. 5

L

LGN lateral geniculate nucleus. 10, 15

LOC Lateral occipital complex (see LOC in Glossary). 10

LSTM Long Short-Term Memory (see LSTM in Glossary). III, IV, VIII, 16–18, 34, 36

M

MEG magnetoencephalography. 10

MT middle temporal visual area. 10

R

RGB-D Red-Green-Blue-Depth. 5, 15, 22, 28

RL Reinforcement learning (see Reinforcement learning in Glossary). 8, 11, 12, 23–26, 36

RNN Recurrent neural network (see RNN in Glossary). II–IV, 9, 11, 12, 16, 17, 19, 20, 22, 27, 34

V

V1 Primary visual cortex (see V1 in Glossary). 10, 12, 15

VDNN Ventral dorsal neural network (see VDNN in Glossary). 10

Z

ZHAW Zurich University of Applied Sciences. 5, 7, 28

A. Appendix

A.1. Contact Information and Code Access

If you have any questions regarding this work, you can contact us via dano.roost@gmail.com or ralphlmeier@gmail.com. All source code should be available at:
- `https://github.com/Danoishere/ba-brain-net` (scene understanding network)
- `https://github.com/Danoishere/ba-simple-scene-gen` (training data generation)

A.2. Remaining Stream Evaluations

Remaining Stream Evaluation - No occlusion

Stream	**Screen Space Coordinates to Class**	**Has Neighboring Object**
Metric	Accuracy	Accuracy
Baseline (CNN)	0.8	0.95
CNN + RNN	0.92	0.99
ConvLSTM	0.79	0.97

Table A.1.: Remaining evaluation of streams not listed in section 3.6 for **non-occluded** scenes.

Remaining Stream Evaluation - With occlusion

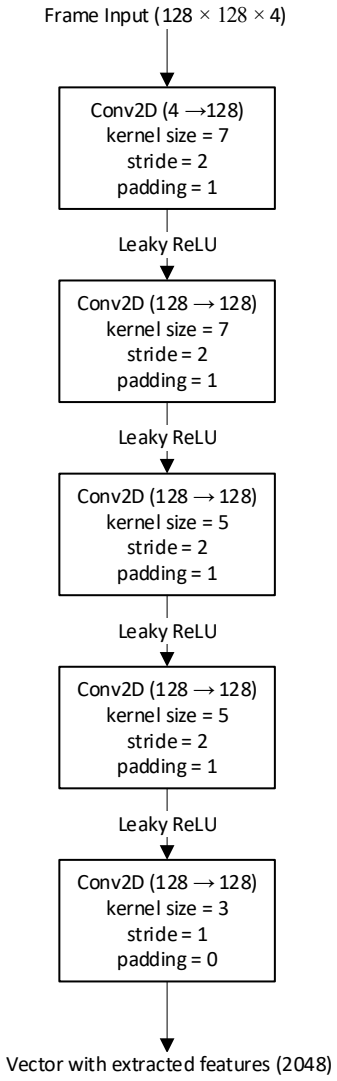
Stream	**Screen Space Coordinates to Class**	**Has Neighboring Object**
Metric	Accuracy	Accuracy
Baseline (CNN)	0.54	0.73
CNN + RNN	0.89	0.95
ConvLSTM	0.67	0.93

Table A.2.: Remaining evaluation of streams not listed in section 3.6 for **occluded** scenes.

A.3. All Used Neural Network Architectures

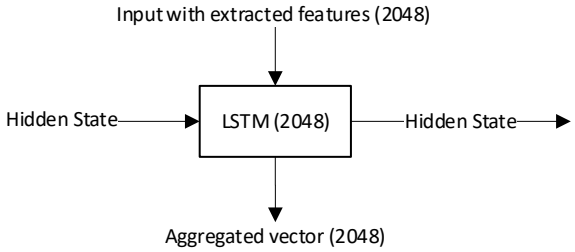
Feature Extraction Network (Baseline and CNN + RNN)

Feature extraction net used for both the baseline and the CNN + RNN version.



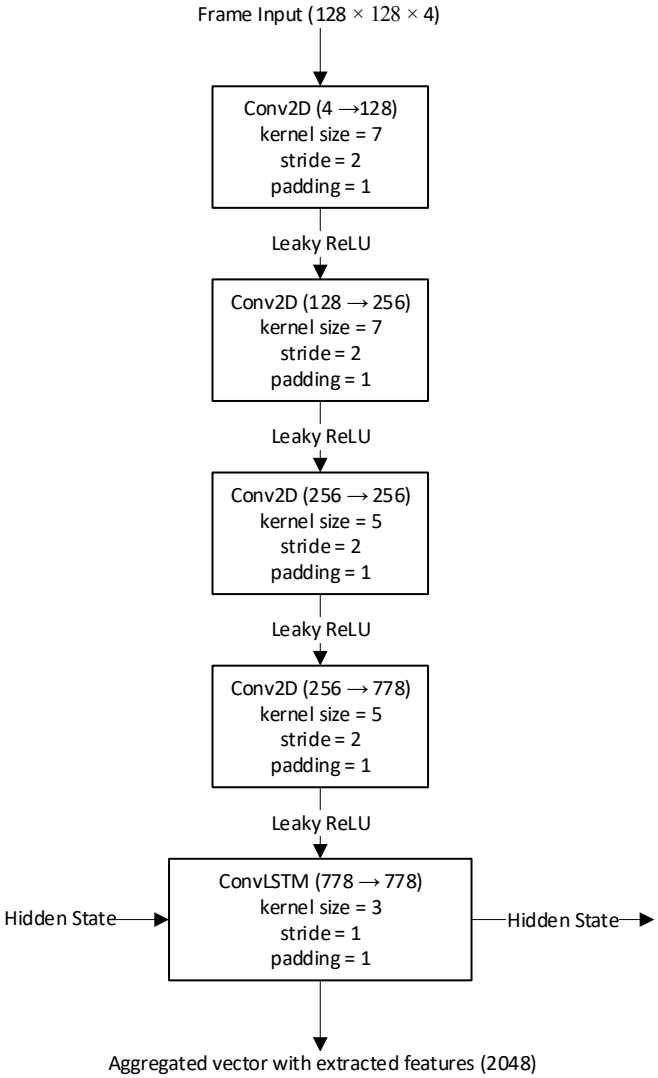
Recurrent Part of CNN + RNN

LSTM layer, placed after the feature extraction for the CNN + RNN version.



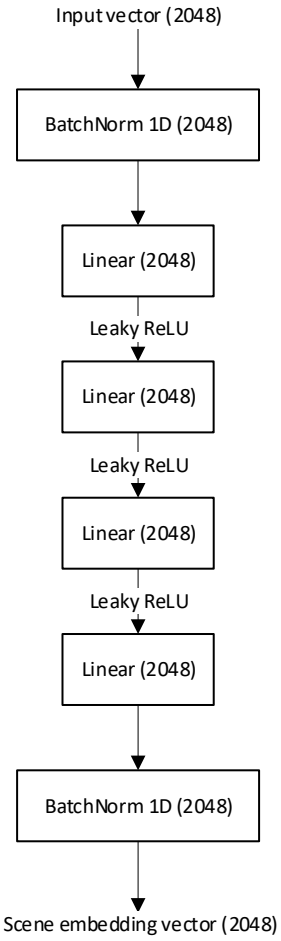
ConvLSTM

Architecture of the evaluated ConvLSTM version, up to the scene embedding output.

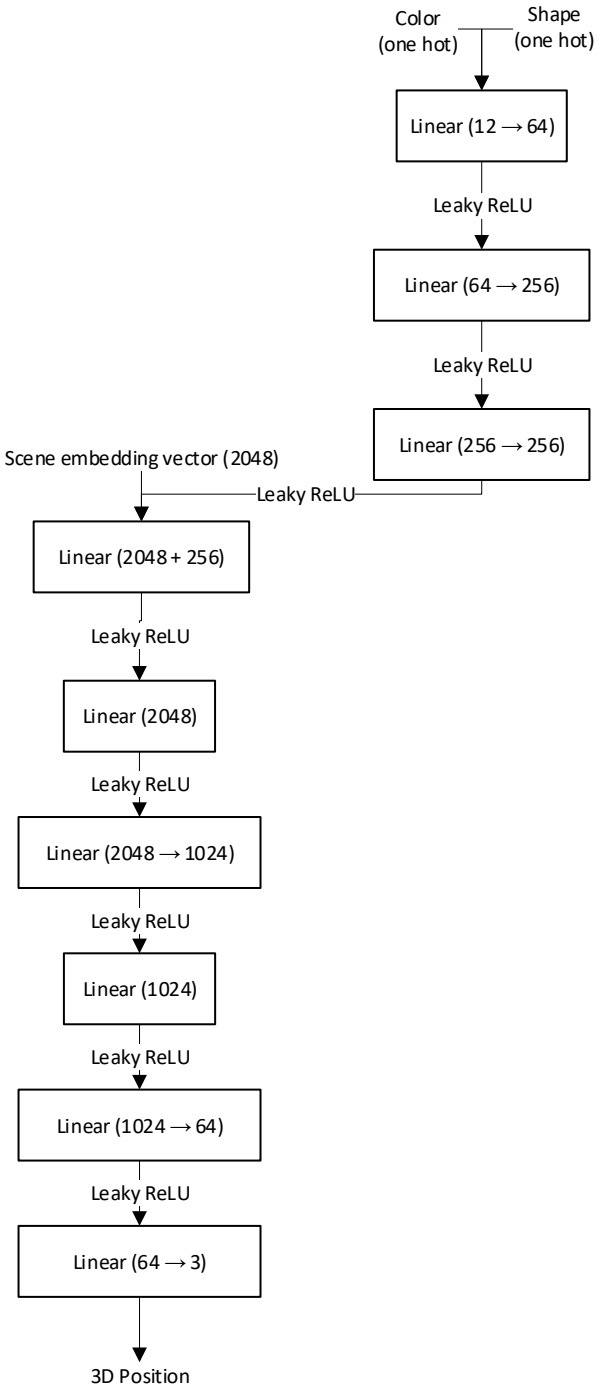


Fully Connected to Scene Embedding

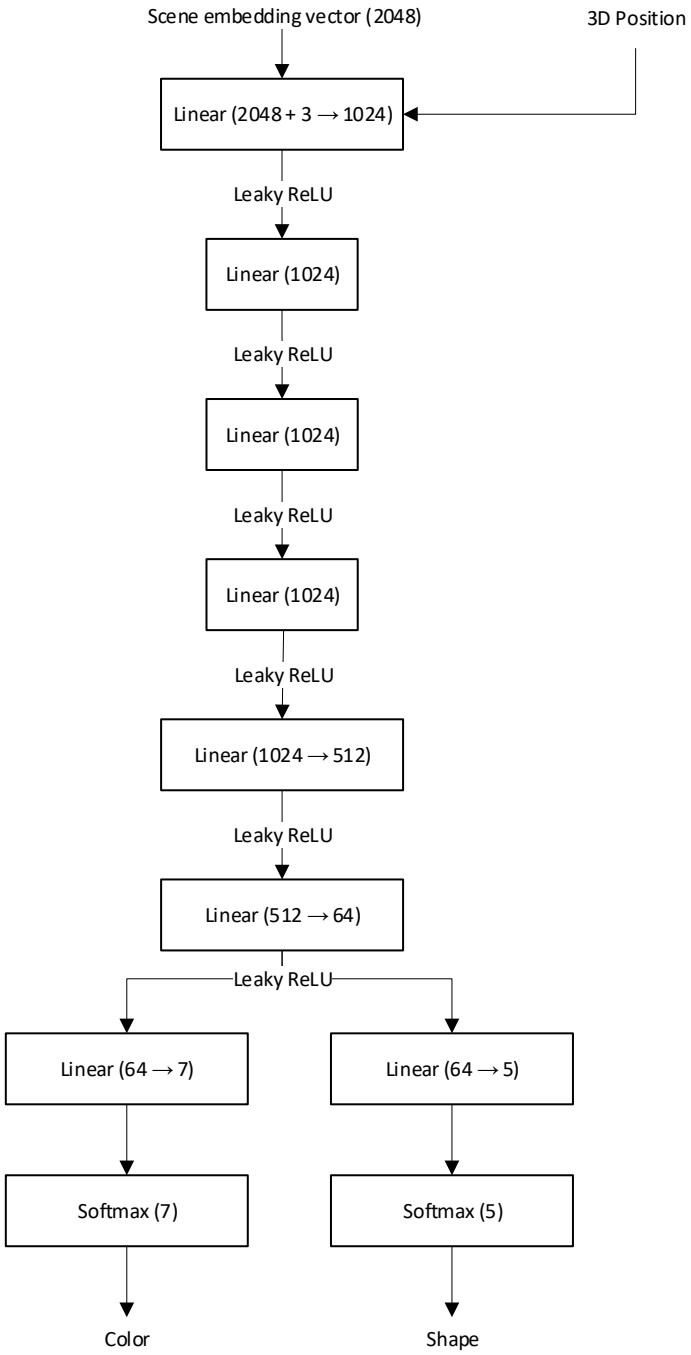
These fully connected layers are used in all architectures for creating the scene embedding vector after feature extraction (and for the CNN + RNN approach after the LSTM layer).



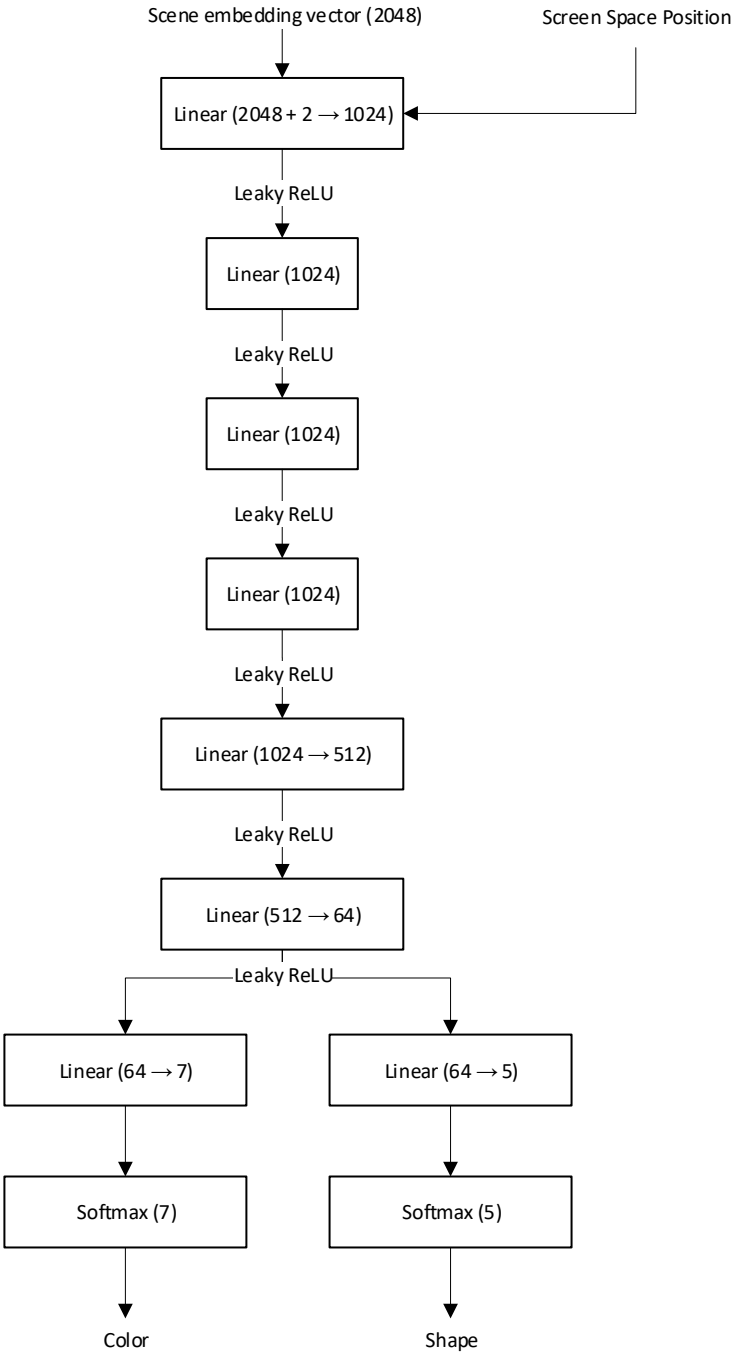
Stream - Class to Position



Stream - Position to Class

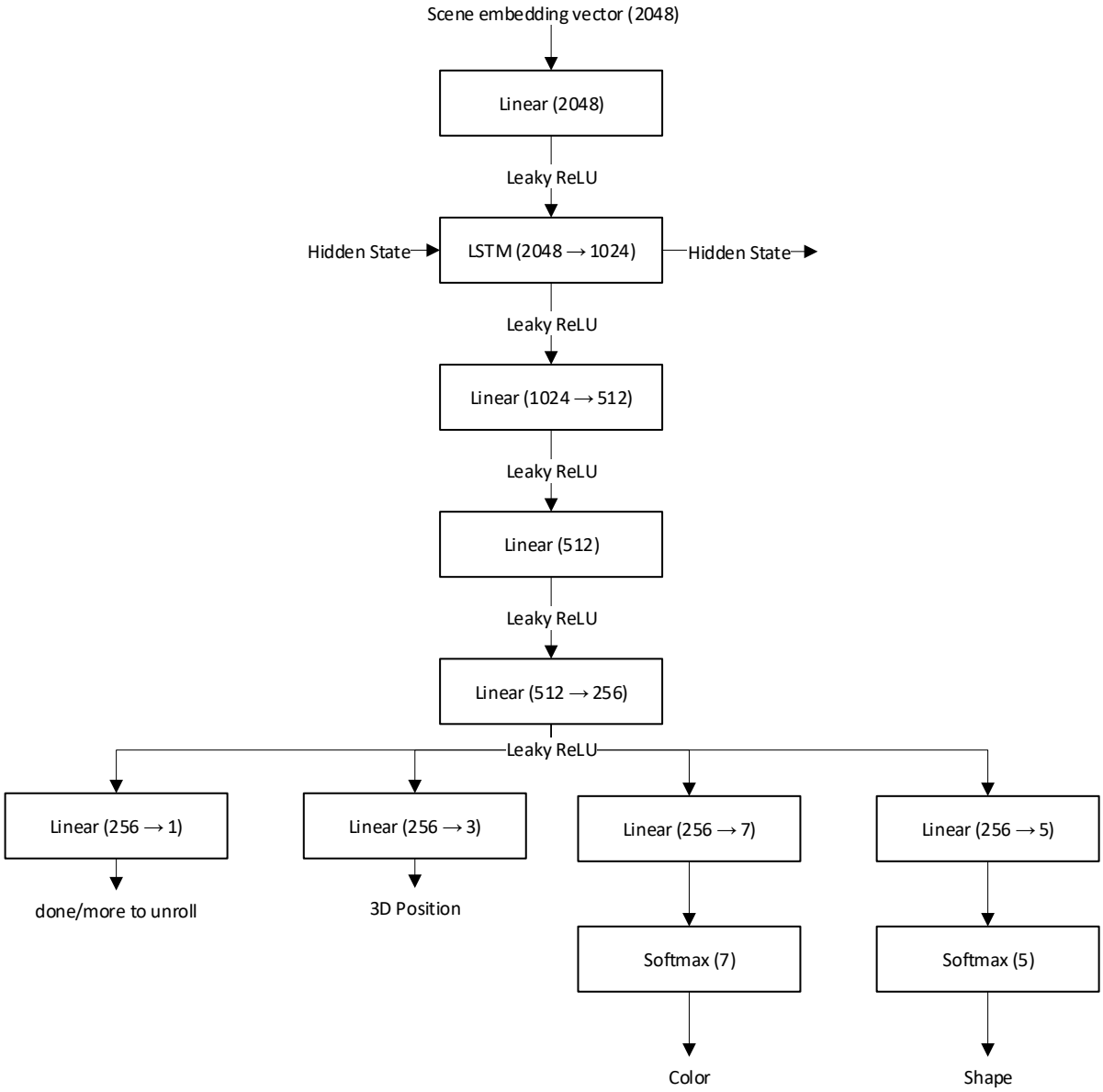


Stream - Screen Space Coordinates to Class

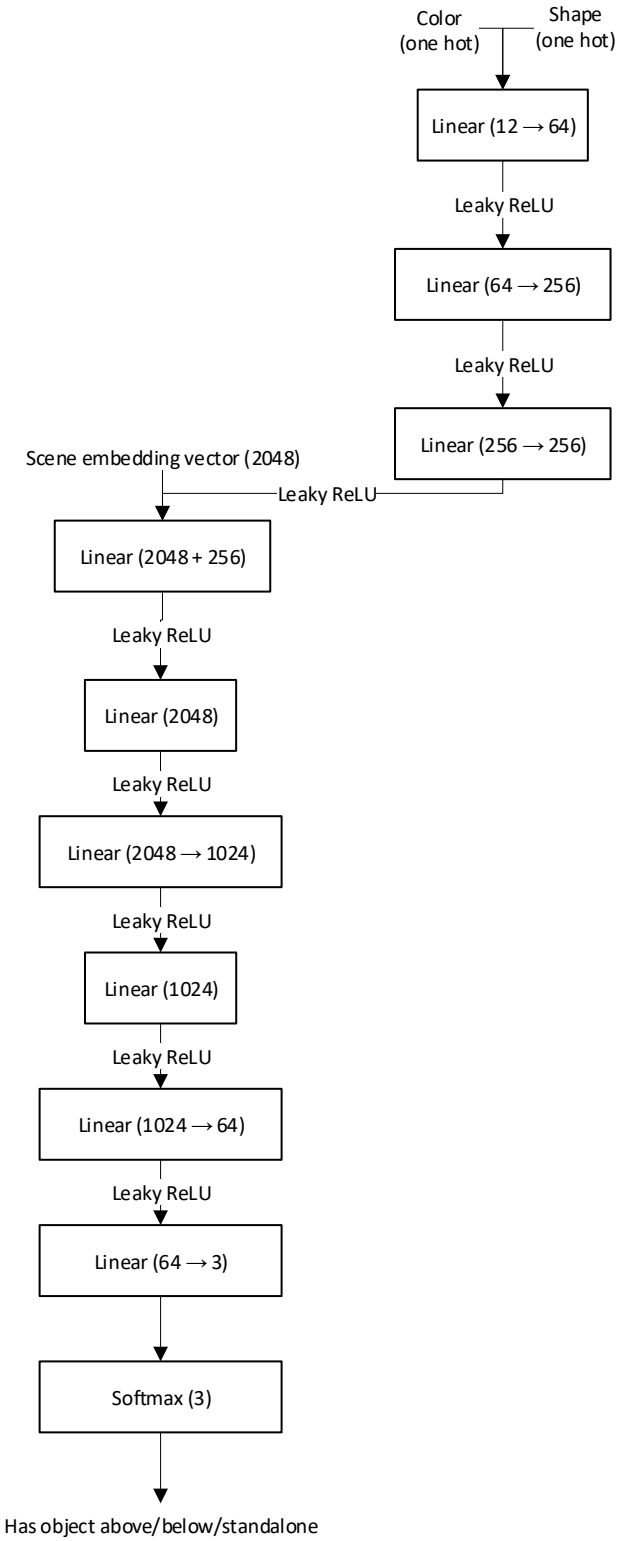


Stream - Enumerating

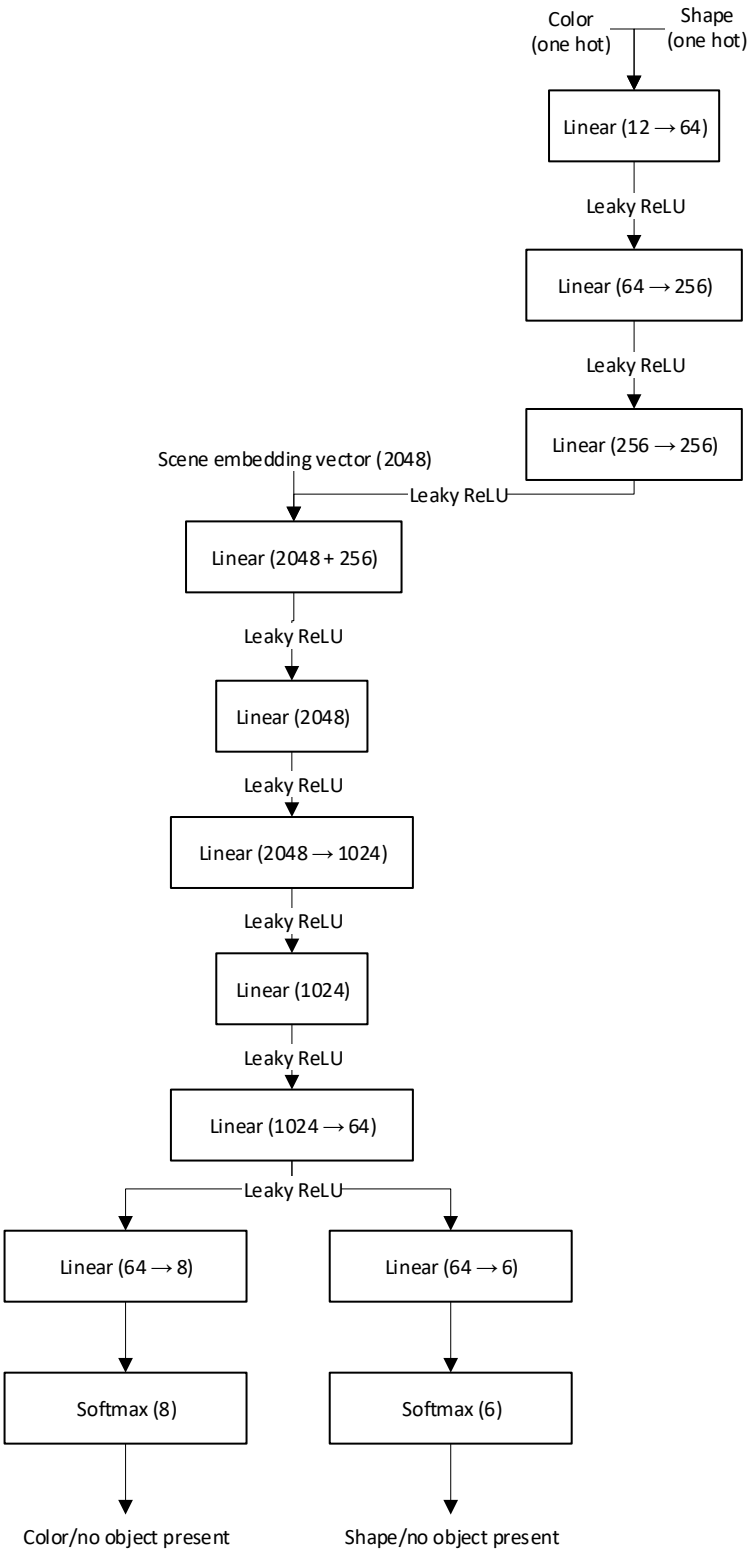
LSTM layer, followed by different heads that output object specific information



Stream - Has Object Below Or Above

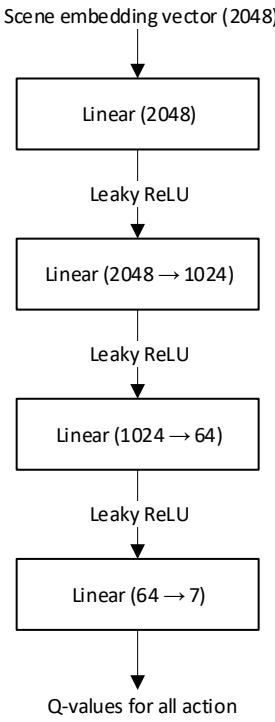


Stream - Class Below Or Above



Stream - QNet

Stream for active vision. Trained using reinforcement learning (see chapter 4).



A.4. Offical Assignment

Zürcher Hochschule für Angewandte Wissenschaften School of Engineering

[Online Administration Praktische Arbeiten]

[Dept. T Admin Tools]

zurück Logout

Bachelorarbeit 2020 - FS: BA20_toff_01

Allgemeines:

Titel:	Active Scene Understanding From Image Sequences for Next-Generation Computer Vision
Anzahl Studierende:	2
Durchführung in Englisch möglich:	Ja, die Arbeit kann vollständig in Englisch durchgeführt werden und ist auch für Incomings geeignet.

Betreuer:

HauptbetreuerIn: Giovanni Toffetti Carughi, toff
NebenbetreuerIn: Thilo Stadelmann, stdm

Zugeteilte Studenten:

Diese Arbeit ist zugeteilt an:
- Ralph Meier, meierr18 (IT)
- Dano Roost, roostda1 (IT)

Fachgebiet:

SOW Software

Studiengänge:

IT Informatik

Zuordnung der Arbeit :

InIT Institut für angewandte Informationstechnologie

Infrastruktur:

benötigt keinen zugeteilten Arbeitsplatz an der ZHAW

Interne Partner :

Es wurde kein interner Partner definiert!

Industriepartner:

Es wurden keine Industriepartner definiert!

Beschreibung:

In order to pick up an object a robot needs to interpret the scene around it and figure out where the object to be picked is and what is the best way to grasp it (e.g., approach, gripper opening, gripper position). In order to do this, robots use depth cameras to provide a 3D representation of a scene in the form of a "point cloud". Point clouds, just like 2D images, need to be interpreted (possibly semantically) to identify which 3D point belongs to which object. The operation of assigning each point to an object is called segmentation. The goal of this work is to use AI to produce a semantic-based segmentation algorithm leveraging the stream of data coming from an arm-mounted depth camera.

zurück Logout