Karlsruhe Institute of Technology

Carnegie Mellon University

Using Scene-Aware Voice Dialogs in Human-Drone Interaction

Master's Thesis of

Tino Fuhrmann

At the Department of Informatics
Institute for Anthropomatics and Robotics
Interactive Systems Lab
Karlsruhe Institute of Technology
Karlsruhe, Germany

School of Computer Science
Robotics Institute
Air Lab
Carnegie Mellon University
Pittsburgh, United States

Reviewer:	Prof. Dr. A. Waibel
Second Reviewer:	Prof. Dr. S. Scherer

Duration: June 2019 – January 2020

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE

. . . . . .

(Tino Fuhrmann)

Abstract

Progress in robotics research enables drones to be deployed in more and more use cases. However, the user interfaces in particular for high-level mission control (e.g. search and report persons in specific area) remain complex and unintuitive for the average user. To solve these issues, we propose an interactive dialog system based on the Transformer that is scene-aware and enables intuitive high-level mission control. In particular, an approach with which this system can generate goal-oriented questions that simplify interactive mission clarification based on scene-understanding is proposed. We demonstrate these capabilities by deploying this system on an autonomous aerial vehicle for a person following task. Experiments using a user simulator confirm that our system is scene-aware. An online user study and end-to-end systems further show that users are able to intuitively interact with the system with a high degree of success without formal training. This lays the groundwork for significantly reduced barriers to the deployment of drones in new applications by users without extensive formal training.

Zusammenfassung

Fortschritte in der Robotikforschung erlauben den Einsatz von Drohnen in immer mehr Bereichen. Die Mensch-Maschine-Schnittstelle für abstrakte Missionen (z. B. Absuchen eines Areals und Meldung von gefundenen Personen) ist jedoch weiterhin für den durchschnittlichen Nutzer unintuitiv und komplex. Um diese Schnittstellen zu verbessern, präsentieren wir in dieser Arbeit ein interaktives System basierend auf dem Transformer-Modell, dass Szenen-Verständnis lernt und die intuitive Steuerung von Missionen ermöglicht. Weiterhin wird ein Ansatz zur Generierung von zielgerichteten Fragen vorgestellt, der das Erreichen eines gemeinsamen Verständnisses von Missionszielen zwischen Nutzer und Drohne vereinfacht. Um diese Fähigkeiten zu demonstrieren, wird dieses System auf einer Drohne für das Folgen von Personen eingesetzt. Experimente mit einem Nutzersimulator zeigen, dass dieses System tatsächlich Szeneninformationen zur Lösung der Aufgabe verwendet. Eine Online-Nutzerstudie, sowie ein Ende-zu-Ende Systemtest zeigen weiterhin, dass Nutzer intuitiv mit dem System interagieren und es erfolgreich einsetzen können. Dies legt den Grundstein für signifikant reduzierte Barrieren für den Einsatz von Drohnen in neuen Anwendungsfällen durch Nutzer ohne extensives, formales Training.

Contents

Abstract i

Zusammenfassung iii

1. Introduction 1

2. Related Works 3
2.1. Interaction Strategies . . . 3
2.2. Dialog Systems . . . 5
2.2.1. Mono-Modal Dialog Systems . . . 5
2.2.2. Integrating Scene Information into Dialog Systems . . . 8

3. Scene-Aware Dialog Systems 11
3.1. System Overview . . . 11
3.2. The Transformer Model . . . 14
3.2.1. Attention . . . 15
3.2.2. Functional Principles of the Encoder and Decoder . . . 16
3.3. Integration of Scene-Awareness into the Transformer . . . 16
3.3.1. Modifications to Encoder . . . 17
3.3.2. Modifications to Decoder . . . 18
3.3.3. Target Selection . . . 19
3.4. Training . . . 20
3.5. Limitations . . . 21

4. Drone System 23
4.1. Hardware . . . 23
4.2. System Overview . . . 24
4.2.1. Flight Control . . . 25
4.3. Mission Control . . . 27
4.3.1. Person Detection & Tracking . . . 28
4.3.2. Human-Robot Communication . . . 31
4.3.3. Dialog System . . . 31
4.3.4. Global Planner . . . 32

5. Experimental Evaluation 35
5.1. Data Sets . . . 35
5.2. Data Collection & Augmentation . . . 35
5.3. Synthetic Dialog Generation . . . 36

5.4. Experiments 37
5.4.1. Experiment 1: User Simulator 37
5.4.2. Experiment 2: Online User Study 41
5.4.3. Experiment 3: System Evaluation 43

6. Conclusion & Future Work **47**

Bibliography **49**

A. Appendix **55**
A.1. Dialog System 55
A.1.1. System Outputs and Responses 55
A.1.2. Image Feature Extractor 56
A.2. Drone System 57
A.2.1. Hardware 57

List of Figures

2.1. **Dialog System Components** 5

3.1. **System Overview** 12
3.2. **The Transformer Model** 15
3.3. **Modified Encoder** 17
3.4. **Modified Decoder Layer** 18
3.5. **Feature Combination Types** 19
3.6. **Target Selection Network** 19

4.1. **Drone Platform** 24
4.2. **System Overview** 25
4.3. **Grid Map and Trajectory Library** 26
4.4. **Deproject-Transform-Project Workflow for Alignment of Depth Image with Color Image** 30
4.5. **Global Planner State Machine** 32

5.1. **Three Data Sets** 35
5.2. **Action Confusion Matrix for Model Trained with „*Summation*“ Feature Combination Type** 40
5.3. **Interface of the User Study** 41
5.4. **Histogram of Dialog Length** 42
5.5. **Testing Area with two test persons** 43

A.1. **CNN Used for Feature Extraction from Person Images** 56

List of Tables

2.1. **Example: Intent Classification and Slot Filling** 6

3.1. **System Inputs and Outputs** 12
3.2. **Example: Action Types Used in the Dialog System** 14

5.1. **Overview of Images in Each Data Set** 36
5.2. **Comparison of Feature Combination Types** 38
5.3. **Results: Generalization of Model to Other Data Sets** 40
5.4. **User Study: Average Dialog Length and Target Accuracy at the End of Dialog** 42
5.5. **Survey** 43
5.6. **End-to-End System Test Results** 44

A.1. **List of Actions** 55
A.2. **List of Questions** 56

1. Introduction

In the past few years, the capabilities of autonomous unmanned aerial vehicles (UAVs), and in particular drones, have grown enormously due to advances in the various areas of robotics research. Additionally, commercially available, light-weight computers have considerably increased the computational power available on these platforms. Combined, this enables these UAVs to be deployed in new use cases that benefit from improved autonomous behavior like search and rescue (SAR), industrial inspections, and surveying.

While these applications greatly benefit from the use of autonomous aerial, their complex user interfaces make the deployment of these robots in the real world, without significant amounts of training and experience, difficult. Additionally, these interfaces often require the full attention of the operator hindering them from performing other tasks. With more natural and interactive user interfaces, such as the natural spoken dialogue systems used in this thesis, coupled with a highly autonomous robot, these issues could be alleviated, and the efficiency and user-friendliness of using drones significantly increased, so that even persons without any formal training can operate these vehicles.

For example, this would allow SAR personnel to treat drones similar to a human operator with whom activities are coordinated, and information is shared. Thus, the required personnel could be reduced or focus on other tasks while the drone autonomously completes its mission. Similar arguments hold for other use cases like indoor drone assistants, which aim to complete tasks like finding a person or checking room availability upon a request by a user, or delivery drones which could interactively negotiate package drop-off locations with users.

All of these use cases have in common that user requests have to be understood in the context of the surroundings of the drone. Traditionally, this context was integrated into dialog systems by storing high-level concepts, that were detected in the scene, in a system-queryable knowledge base. In addition to being prone to misdetections, the design of appropriate detectors for these concepts also requires significant engineering effort. The few recent works [25, 31, 65] that research the integration of visual features into state-of-the-art dialog systems solve this issue by directly integrating visual information into their system model. However, these models focus on question-answer dialogs and are not designed to interactively work with the user to clarify user requests and complete tasks.

In this thesis, we address this limitation and present a dialog system that integrates scene-understanding based on visual information into the Transformer [57]. We propose to use multi-head attention to capture multi-modal features in the input. Furthermore, a mechanism that selects appropriate questions to clarify user requests is presented. This clarification is not limited to merely requesting the user to change their request but includes asking appropriate, goal-oriented questions based on the current scene. Furthermore, our

system can be easily ported to new domains because it learns relevant high-level visual features directly from dialogs with corresponding images.

To demonstrate these capabilities, we further present an autonomous drone system that is able to autonomously follow a person that was selected by spoken dialog. We call this task, that was inspired by a potential mission of an indoor drone assistant, the person following task. In particular, we will discuss the optimizations that allow us to run all components of the system, with the exception of speech recognition, fully on-board the drone.

This thesis is divided into five chapters. In chapter 2, the use of speech as the input modality of our system is motivated. Additionally, an overview of state-of-the-art approaches to dialog systems, including the integration of visual features, is given. The third chapter discusses how visual information and target selection are integrated into a state-of-the-art dialog system. Following this, the system design of the autonomous quadrotor for the person following task is described in chapter 4. Afterward, in chapter 5, the dialog system and the drone system are evaluated. Finally, the conclusion and future work are presented in chapter 6.

2. Related Works

In this chapter, we review various areas of research closely related to human-robot interaction and scene-aware dialog systems. In particular, we first present several human-robot interaction strategies and discuss their applicability to the person following use case introduced in chapter 1. After that, a short introduction to dialog systems and an overview of state-of-the-art techniques is given. To conclude this chapter, current research on the integration of additional modalities, e.g., an image of the scene as in our use case, into dialog systems is examined.

2.1. Interaction Strategies

Human-robot interaction has an immense impact on the usability and the user's perception of the fitness of a robot to complete a particular task. While many different interaction strategies are described in the literature, there is no one size fits all approach suitable for every task and environment. This is in part due to the varying requirements of each strategy and task, such as the needed level of instruction details or availability of sensors.

In our scenario, the interface should be intuitive and hands-free to keep the operator's workload caused by controlling the robot at a minimum. Furthermore, to allow seamless switching between users, only commonly available sensors, such as cameras or microphones, should be required to interact with the robot. With these limitations in mind, we give an overview of various interaction strategies, their advantages and disadvantages, as well as their applicability to our problem in the following sections.

The literature distinguishes two kinds of user interfaces [17]:

- **Graphical User Interfaces (GUI)** use graphics, images, buttons, and/or a window manager to accept control inputs and convey information to the user
- **Natural User Interfaces (NUI)** use a more direct approach and allow instructing the robot directly through speech, gestures or other means

GUIs, such as WIMP (Windows, Icons, Menus, Pointer) interfaces, are commonly used on various input devices such as personal computers and mobile phones [17]. This is contrary to the requirement of hands-free use and makes GUIs not suitable for our scenario. Furthermore, GUIs are often used for complex systems, and much effort is being expended on making these interfaces more intuitive to use through modern technologies like touch screens [6] or augmented and virtual reality [20]. Nonetheless, GUIs are well suited for systems or tasks that require high precision, the configuration of many parameters, or an overview of the internal and external state.

In contrast to GUIs, NUIs remove the intermediate graphical layer to provide a more direct interaction modeled after human-human communication forms such as spoken language, body language, and markers [17]. However, not all NUIs are equally suitable for our use case.

Markers, a method that uses physical signs to issue commands, in particular, are not appropriate for our case. While marker-based systems can achieve high accuracy and responsiveness [17], the use of physical signs (e.g., a sign with a printed, machine-readable code [17, 14]) to issue robot commands, makes them inherently not hands-free. Furthermore, with increasing complexity of commands, the use of markers becomes unintuitive, which significantly hampers the speed of communication compared to more natural modalities like gestures [14].

Gestures, while better suited than markers because of their expressiveness and intuitiveness, have their own set of challenges. Specifically, gesture recognition without additional sensing capabilities such as wearable sensors [40, 3, 6] or depth cameras [17] remains challenging. Nonetheless, gesture recognition has been successfully applied to various tasks [40, 59, 28, 39], including SAR [6].

All communications forms presented above, have one property in common: They are only used in human-human interaction to support or, if necessary, substitute spoken language. Speech satisfies almost all requirements, in particular, intuitive and hands-free use, and is thus the best suited of the communication forms presented. This especially applies to its intuitiveness, which enables nearly everyone, including persons with little technical understanding, to interact with robots. This intuitiveness is also supported by advances in dialog system technology that enable the translation of dynamic interactions to actionable robot commands [42, 50, 49, 30, 32, 18]. Furthermore, the ubiquity of microphones in today's world makes speech recognition possible in a wide variety of contexts.

Nonetheless, there are also some limitations to the application of speech. Noise or very loud environments can make speech recognition difficult or even impossible. Additionally, while speech can be precise, it is difficult to achieve precision comparable to other means like pointers in WIMP interfaces. However, this precision is not required for the high-level mission descriptions that the use cases proposed in the introduction require. Furthermore, for these use cases, the operator will generally be far enough away from the drone such that its noise will not significantly interfere with speech recognition.

The flexi-modal approach [38] combines multiple interaction forms to enable the use of the best-suited or best-suited combination of modalities in every scenario. This is a very intriguing and well-suited interaction mode for our use case because it enhances the intuitiveness and ease of use. However, this integration comes with unique challenges. Contradictions, ambiguities, and references between the input sources must be resolved, possibly with the involvement of the user and an understanding of the scene, increasing system complexity significantly [51].

Furthermore, as only little work has been done on integrating scene-awareness into the communication forms presented in this section, building a scene-aware flexi-modal system may present many additional research challenges in various areas. Thus, as a first step, this work focuses on integrating scene-awareness into a speech-based dialog system.

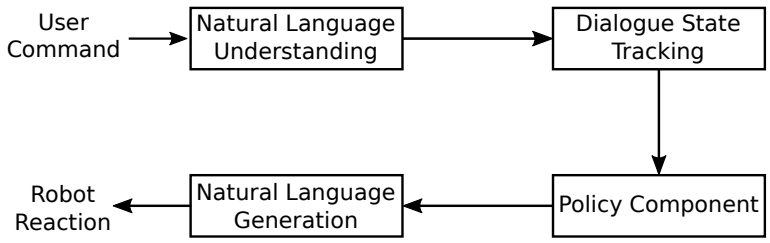


Figure 2.1.: **Dialog System Components**
Dialog systems generally consist of four components responsible for understanding the user query, the context of the dialog and determining the correct course of action. Depending on the techniques used to implement the system, two or more of these components may be merged together, thereby reducing the need for the coordination of their interactions. (Source: [7], Fig. 1)

2.2. Dialog Systems

Research on dialog systems has made tremendous progress over the past years. Advances in machine learning, such as the encoder-decoder model [53] or the Transformer [57], have led to significantly improved performance on various dialog tasks. However, even though this is a very active area of research, only few works on integrating multiple modalities into these systems have been published.

To better understand the trade-offs involved in choosing certain models over others, we first give an overview of the state of the art of traditional mono-modal dialog systems. Thereafter, we present works on integrating additional modalities and analyze approaches from other research areas like for example, visual question answering (V-QA), with respect to their applicability to our use case.

2.2.1. Mono-Modal Dialog Systems

Traditionally, dialog systems can be grouped into two categories: Task-oriented and non-task oriented systems [7]. Non-task oriented dialog systems try to have a natural conversation on open domains with the user and thus require external knowledge to have meaningful dialogs [45]. In contrast, task-oriented dialog systems try to aid the user in solving a particular task, e.g., making travel arrangements in the case of the Airline Travel Information System (ATIS) [22] or selecting a person to approach and follow in our scenario. Hence, we concentrate on task-oriented systems in this section.

Task-oriented dialog systems traditionally use a pipeline approach with four stages, as shown in Fig. 2.1 [7].

First, the natural language understanding (NLU) component creates an internal representation of the user's request. This representation consists of two components: The intent of the user, describing the user's goal, and the parameters, called slots, of the intent with their associated values. For example, in Table 2.1 the intent of the user's command „*Fly to the person wearing the green shirt*“ is *FlyToPerson*. In the context of *FlyToPerson*, only one slot value was not marked as *ignored (IGNR)* - the *COLOR* parameter with value *green*. A good model for these two tasks must be robust to significant variations in sentence structure and word choice for the same intent, and very similar expressions for different classes [12].

Command	Fly	to	the	person	wearing	the	green	shirt
Slots	IGNR	IGNR	IGNR	IGNR	IGNR	IGNR	COLOR	IGNR
Intent	FlyToPerson							

Table 2.1.: **Example: Intent Classification and Slot Filling**

After this representation is obtained from the NLU component, it is passed to the dialog state tracking (DST) subsystem, which is responsible for estimating the current dialog state. This includes estimating the current intents and slot values with respect to the entire dialog history. The policy component then uses this state to select an appropriate action. This can range from simple responses like asking the user about specific parameter values to issuing API calls. Finally, this action is converted to a human-understandable format by the natural language generation (NLG) component.

Naturally, there are multiple approaches to implementing this pipeline: On one end of the extreme, each system unit can correspond to one component of Fig. 2.1, all tightly integrated and dependent on each other's inputs and outputs. On the other end of the spectrum, the entire pipeline can consist of one end-to-end trainable component. This greatly simplifies system design and improves the transferability to new tasks and domains [68, 5]. Furthermore, the design could be anywhere in-between these two extremes featuring various levels of component integration. Today, most state-of-the-art systems integrate some or all of these components into one model trained either in a supervised fashion or with reinforcement learning (RL).

For our model, we employ supervised learning due to the greater flexibility afforded by models trained in this manner compared to RL approaches. This is in part because RL dialog system approaches are based on a discrete action-state representation which requires either separate classes for actions and parameters or one class with all action-parameter combinations (e.g., *QuestionColor COLOR_green*). However, both of these representations are not well suited for our scenario. The first does not appropriately model the relationships in between parameters and between actions and parameters, while the latter would lead to a very large action-state space and, thus, exacerbate the slow convergence to a good policy reported by [68, 64, 63, 13]. While the convergence issue was solved by pre-training using supervised learning [68, 64, 63, 13], our approach would not additionally benefit much from optimizing for additional metrics like dialog length because most dialogs in our scenario are shorter than three steps.

These two facts highlight some of the significant challenges of applying RL to dialog systems. Consequently, to be able to focus on integrating scene-awareness, we chose a supervised approach. Future work may, however, explore how the model presented in this work can be trained with RL to allow for additional optimizations from which other tasks might benefit, and to enable online learning from weak dialog success signals [64].

In the following section, we provide an overview of state-of-the-art techniques for supervised, end-to-end dialog systems and relate them to our approach. Thereafter, natural language generation is briefly discussed.

Supervised Models

A key difference between end-to-end models and their modularly implemented counter-

parts is that end-to-end models must encode the dialog state, including past steps, within the model.

One mechanism capable of this are memory networks (MemN2N) [62, 52]. MemN2N provides an internal memory with a learnable access mechanism that the network can use to store and read data (e.g., encoded dialog steps [5, 61]) before making a prediction. While this model was successfully applied to dialog systems [5, 61, 36] and showed promising results compared to non-neural models, it is not able to represent the dynamically generated action-parameter sequences required for our scenario without additional modifications for sequence prediction [43].

This issue is avoided by the attention-based [4] encoder-decoder RNN [54] approach to dialog systems [15, 9] on which our model is based. [15, 9] interpret dialogs as a sequence-to-sequence task and directly predict the dialog state or policy response from the concatenated dialog history. Thus, this model can generate the dynamic action-parameter sequences needed for our scenario. Additionally, this model, unlike MemN2N, does not encode each step in one fixed-length vector and is consequently able to retain all information contained in the dialog history at the expense of longer input sequences.

Our solution deviates from [15, 9] in two ways. One, we replace the encoder-decoder RNN with the Transformer[1] [57] to take advantage of its superior parallelizability and thus training speed. Moreover, because in our scenario multiple responses can be correct for a given scene and dialog context, we do not directly predict the next response. Instead, we predict an intermediary action-parameter form that is mapped to a set of valid questions. From this set, the next question is chosen at random. This is comparable to how [9] generates different human-understandable texts for the same action.

Other systems use sequence-to-sequence models in a more modular fashion. In [60], an end-to-end model, consisting of a separate network for each dialog system component, is designed. While this leads to a more flexible approach and allows the integration of additional components, it requires significantly more engineering effort than our method. Our method instead separates the modules of our model by feature type (e.g., image, text) and thus must only ensure correct output dimensions to integrate the subnetworks.

Natural Language Generation

Up until now, we have not discussed how the action is converted to a human-understandable form and communicated to the user. In this section, we briefly summarize the two methods that all the approaches presented above use.

[5, 8] use the first method and employ a generative model to create a human-understandable response directly. By contrast, all other methods use a template-like representation [16, 35, 60, 63, 64] from which the final text is built. This template-like representation is either manually defined [63, 64] or generated online with a neural model [16, 35, 60]. Generating responses with neural models has the advantage that suitable outputs can be directly learned from training data, thus saving engineering effort on designing valid output sentences. However, these models may generate grammatically incorrect responses and their variability in sentence structure and word choice is heavily dependant on the

[1] This was first brought up by Stefan Constantin when he introduced me to his research while we prepared a conference paper

training data. As only little training data is available for our use case, our model uses manually defined templates. This guarantees a certain degree of variability, grammatical correctness and naturalness of system responses.

2.2.2. Integrating Scene Information into Dialog Systems

Finally, to make the dialog system scene-aware, relevant scene features have to be extracted and integrated into the dialog system model. This requires extracting and combining features from both the dialog and the image so that an informed decision about the intent and slot values can be made. Research on this topic has only been gaining traction in the last year due to the publication of the Dialog State Technology Challenge 7 (DSTC7) [65] that provided a question-answer dialog data set for audio-visual scene-aware dialog (AVSD). In the following paragraphs, we first discuss works on fusing features from multiple modalities. Afterward, we review recent approaches to integrating additional modalities into dialog systems, including approaches developed for DSTC7.

Feature Fusion Methods
Feature fusion is an essential part of multi-modal tasks, in particular for visual question-answering (V-QA) [1] and audio-visual scene-aware dialog [65]. Most state-of-the-art techniques use several encoders to extract features from each modality, e.g., convolutional neural networks (CNNs) for image features or recurrent neural networks (RNNs) for text before they are combined. Feature combination is then performed using one of four general techniques [67]: Simple vector operations, attention, pooling, or neural networks.

In this work, we evaluate the first two methods and the neural-network-based approach for the combination of image, user command, and previous decoder output features. In the following section, we briefly introduce each technique and provide a brief discussion on how they relate to our model.

Attention is a popular method that allows the model to focus on specific parts of one modality by calculating a weighted sum whose weights are influenced by this and a second modality. For example, [37] obtain two sets of attention weights by marginalizing a combined matrix over each modality. The transformer also uses attention extensively, among others, to attend encoder text features based on prior decoder outputs. Our model uses the same mechanism to select the most relevant image features. However, in contrast to other methods [37, 19] we do not split the image into multiple regions and instead follow [34] that uses the features of detected objects as image features.

Note, that attention does not directly combine both modalities but only allows extracting important parts of one based on the other. Thus, many models use other means like multi-modal attention [26] or vector operations to combine attended features.

For example, multi-modal attention calculates a weighted sum of features of different modalities based on weights derived from another modality. We follow this approach and use it as one of the three feature fusion modes of our model.

A more basic approach are simple vector operations like concatenation [37] or element-wise sum [57]. This forms the second fusion mode that is supported by our model and extends the fusion module of the Transformer by adding attended image features as an additional input.

The third technique is bilinear pooling [56, 19], a method that allows all elements of two modalities to interact multiplicatively. While we do not use this in our model, due to its high computational cost and sensitivity to hyper-parameters [67], we use another mechanism in which all elements can interact non-linearly as the third and last fusion method.

This method follows the simple idea of embedding all modalities in the same feature space and passing them directly to a neural network, e.g., an LSTM in the case of [44]. In our model, this is implemented by concatenating all modalities and then using a multi-layer perceptron with non-linear activation to reduce the concatenated vector to the desired embedding dimensionality.

While the three fusion methods supported by our model provide a way to combine different features, they do not yet provide an obvious way of integration into a Transformer. Furthermore, other methods that combine multiple modalities for audio-visual dialog systems exist. These techniques are reviewed in the next section.

Scene-Aware Dialog Systems

Only a few approaches to integrating multi-modal features into dialog systems have been proposed in the literature. In this section, we first present two simple approaches for this purpose, before reviewing some of the models created for the audio-visual scene-aware dialog (ASVD) track of DSTC7.

A straightforward approach to integrating scene-awareness into a dialog system is the use of an online generated knowledge base with features like detected objects, their locations, and relations (e.g., person wearing green shirt) stored in a knowledge base (KB) [60, 16]. This, however, requires ground truth labels for objects and relations to train detectors for these features. Thus, this approach is not well suited for our goal of reducing the amount of labeling required.

The integration of a CNN feature extractor as an additional input to a standard neural model, e.g., encoder-decoder RNN [10, 66], avoids this problem. However, it may be difficult for such a model to learn the relevant relations between objects if only little training data is available. Furthermore, in our use case, replacing the CNN with a pre-trained model does not necessarily work due to the greater focus on colors than commonly used tasks like object detection or image classification.

Various works with significantly more sophisticated models have been submitted for the DSTC7 challenge. In the following, we present some of these submissions and highlight how they relate to our work.

One of these works, [25], applied the multi-modal attention model introduced in the previous section, to the AVSD track. Similarly, we apply this approach as one of our feature combination modes. However, in contrast to [25], our multi-modal attention is not guided by the last dialog step but guided by all modalities. Furthermore, we treat the dialog history as one continuous sequence instead of using a stacked LSTM to first encode each step and then combine the dialog history.

In [31], a model for audio-visual dialog, very similar to ours, and that also modifies the Transformer, is presented. Comparable to our approach, additional encoders for all newly added features (images, videos, audio, captions in the case of [31]) are integrated. However, we represent image features using the object-based approach of [34] instead

of CNN features. Furthermore, [31] use the last dialog step to attend to audio and visual input. In contrast, we use the entire dialog history for this purpose. Finally, modifications of similar nature are made to the decoder of the Transformer. Both approaches apply the same attention mechanism, used to combine decoder with encoder features in the original Transformer model, to the additional modalities. The only significant difference is that [31] uses these attentions in series instead of in parallel like in our model. The parallel structure separates the decoder layer, essentially in two parts. The first part attends to the input features based on the decoder features of the previous layer, while the second part focuses on fusing these features. In contrast, the serial structure promotes the use of the attended features as memory. Even though we did not individually examine the impact of this particular architectural difference, we hypothesize that it would not have a significant impact due to similar information being available to both models in subsequent layers.

To summarize, we present a novel approach to integrating scene-awareness into an end-to-end learnable dialog system based on the Transformer model. In addition, we propose an approach that can be used to ask goal-oriented questions that are selected using scene descriptions generated by our model. Finally, we also show how this model can be used to determine the target object of a dialog.

3. Scene-Aware Dialog Systems

In this chapter, we present the design of a scene-aware dialog system based on the Transformer [57] and explain which modifications enable the use of additional modalities in this model.

To get a better grasp on the requirements the system has to fulfill, we first review the goals set forth in chapter 1. We reformulate them as follows:

1. The system must be able to determine which person to approach
2. The system must be able to ask clarifying questions if the user command is ambiguous or the system is unsure about its task
3. The system should not require hand-crafted features for person characteristics (like t-shirt color) as input. It should learn appropriate features on its own.

Requirements *1 & 2* imply that a complete dialog system encompassing all four components (NLU, DST, Policy, and NLG) described in chapter 2 must be designed. Requirement *3* forces an implicit encoding of input features directly retrieved from sensors with minimal preprocessing.

In the following sections, we first describe how these criteria are incorporated into the system design from a high-level perspective before presenting the core machine learning model that processes all user inputs and predicts the user's intent. Lastly, the limitations of this model is discussed in-depth. Throughout this section, the person following scenario introduced in chapter 1 is used. However, the general design choices presented herein are equally applicable to other use cases aiming to include scene-awareness in a dialog system.

3.1. System Overview

An overview of the system is shown in Fig. 3.1. The system uses an image and the textual transcription of a user's command as input and emits a target object, an action representing the user's intent, and a response to the user's command. The action and the response are encoded in an action-parameter form as shown in Table 3.1 and translated to text before they are communicated to the user.

The interaction of the user with the system generally begins when a command is sent to the system. This instruction is then processed by the dialog context component as described in [9]: It is concatenated with the n most recent user commands, system-generated actions and responses. Between each of these context steps, a special marker symbol, in our case

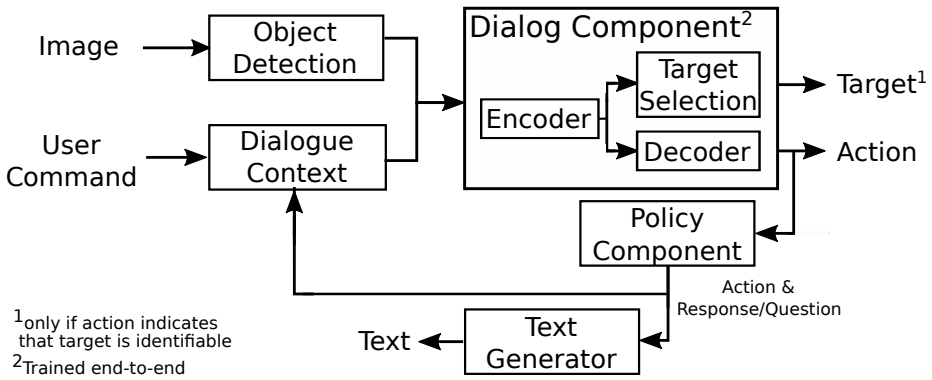


Figure 3.1.: **System Overview**

The dialog system uses an image of the scene and the most recent dialog steps to predict both an action and an intended target. If the targte was not identifiable given the current context, the action is used to ask a question about the scene. Lastly, both the internal question and action representations are translated to text and communicated to the user.

Type	**Encoding**	**Example**	**Meaning**
User Command	Text	Fly to the person in the green shirt	
Action	Intent-Parameter	RejectUnknownColor COLOR_green | COLOR_blue DIRECTION_MIDDLE	No person in a green shirt exists, only one in the center in blue clothes
Target	Object Identifier	2	The third object emitted by the object detection is the target **if** the action indicates a target was identified
Response	Intent-Parameter	QuestionColor COLOR_blue	The system asks the user if they want to fly to the person in blue clothes
Text	Text	I don't see anyone in green. Did you mean the person in blue clothes?	

Table 3.1.: **System Inputs and Outputs**

„#“, is inserted. Extending the example from Table 3.1, the input for the next step may look as follows:

„*Fly to the person in the green shirt # RejectColor COLOR_green | COLOR_blue DIRECTION_MIDDLE # QuestionColor COLOR_blue # yes*“

This design allows the downstream dialog component to use previous dialog steps to make an accurate prediction of the current dialog state.

Finally, byte pair-encoding (BPE) [48] is applied to allow the model to better handle rare and unknown words. To achieve this, BPE, which was originally proposed for natural machine translation (NMT), splits words into frequently appearing subword units, e.g., „*sweetish*“ into „*sweet*“ and „*ish*“. This allows the model to learn separate meanings for these units and consequently generalize beyond the commonly used word-level encoding.

In parallel to the processing of the textual command, the object detector extracts object image crops and their locations from the input image. This extraction step will later facilitate understanding object relations. Additionally, this significantly reduces the required amount of training images because the feature extractor of the dialog component can directly operate on the objects instead of the entire image.

Both, the cropped object images and the textual command, are then used by the dialog system to predict an appropriate action and select one of the input persons as the target. The generated action uses an intent-parameters encoding as shown in Table 3.2. This encoding was inspired by [9, 18] and is well suited for the dialog task because it offers high information density with little noise. Consequently, the process of learning relevant relations between user commands and actions is greatly simplified and a direct mapping to API calls can be implemented.

The policy component distinguishes three intent types to which the system responds differently. These types are:

- **Ignore**: The user command was not directed at the system. Accordingly, the system ignores commands of this type.
- **Goal**: The user specified a clear goal and all required information to determine the target was included in the command. As a response, the system generates a textual confirmation of its mission understanding, determines the targeted object and starts the mission.
- **Reject**: The user tried to specify a goal but their command was either ambiguous, missing information or incorrect. Thus, the policy component randomly selects a response from a set of predefined question types about the scene. The scene description included with every reject action (separated by the „|“ symbol from the other action parameters) is used to dynamically determine parameters for all question types and to filter questions invalid in the current scene. This level of indirection was necessary because our training data lacks the subtle information used by humans to choose the next question in a goal-oriented manner. Thus, the next question could not be learned directly.

Type	Command	Action
Ignore	He's helping Christoph	Ignore
Goal	Fly to the person in the green shirt	FlyToColor COLOR_green
Reject	Follow the person on the left	RejectDirection DIRECTION_LEFT | COLOR_blue DIRECTION_RIGHT

Table 3.2.: **Example: Action Types Used in the Dialog System**

Finally, both the system action and the question are converted into human-readable text by the text generator. From hereon, the user can issue the next instruction and the above process repeats itself.

So far, we have not discussed how the dialog component works. This component is trained end-to-end and integrates language understanding, state tracking and partially policy learning. Similarly to [9, 25], we model the dialog task as a sequence-to-sequence mapping task. The following formalization of this task was adapted to our problem from [25].

Given the input dialog context X, represented as a sequence of words, and the scene context S, encoded as a set of image crops and location features, we use a sequence-to-sequence model based on the Transformer [57] to predict the posterior $P(Y|X,S)$ for the output action-parameter sequence Y. The most likely hypothesis $\hat{Y}$ is then calculated as:

$$\hat{Y} = \arg\max_{Y \in V^*} P(Y|X,S) \tag{3.1}$$

$$= \arg\max_{Y \in V^*} \prod_{m=1}^{|Y|} P(y_m|y1, \cdots, y_{m-1}, X, S) \tag{3.2}$$

Here, V^* represents a set of sequences of words corresponding to system actions and their parameters. Note that during inference, beam search is used to limit the number of sequences which have to be evaluated.

Furthermore, the system in Fig. 3.1 also predicts the most likely target for a command conditioned on the input X, S. As we will describe in more detail in the next section, the target selection model predicts the probability of each person being the target separately, i.e. $P(p_i|X,S,P)$ for all persons $p_i \in P$.

3.2. The Transformer Model

The prediction of both of these probabilities is done by a modified Transformer [57] model[1]. As our changes reuse a lot of the design choices made in the Transformer architecture, we first review the general structure of the Transformer before presenting our modifications.

The Transformer architecture shown in Fig. 3.2 is an encoder-decoder model, i.e. it first encodes the entire input sequence before decoding the output sequence word by word. In

[1]Our implementation is based on https://github.com/quanpn90/NMTGMinor by Ngoc Quan Pham (Interactive Systems Lab, KIT)

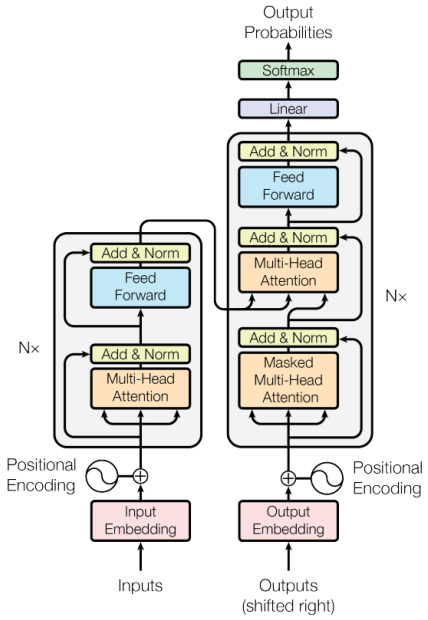


Figure 3.2.: **The Transformer Model**

The Transformer first encodes all elements of the input sequence before fusing this encoding with prior decoder outputs to predict the next word to emit. Note this models use of self-attention which is used to replace recurrent connections of the standard encoder-decoder model. **Source:** [57], Fig. 1

contrast to other encoder-decoder models, the Transformer replaces recurrent connections commonly seen in other encoder-decoder models [53, 4] with self-attention.

Due to the importance of attention to this model, this mechanism is discussed first in the next section. Subsequently, the architecture of the encoder and decoder is presented.

3.2.1. Attention

Attention is a learnable mechanism first introduced by [4] that allows a neural model to focus on specific parts of a sequence. Formally, the attention mechanism for a query sequence $Q = \{q_1, \cdots, q_m\}$, a key sequence $K = \{k_1, \cdots, k_n\}$ and values $V = \{v_1, \cdots, v_n\}$ with output $O = \{o_1, \cdots, o_m\}$ can be described as follows:

$$\alpha_i = \text{softmax}(\text{attention}(W^Q q_i, W^K k_1), \ldots, \text{attention}(W^Q q_i, W^K k_n)) \in \mathcal{R}^n \tag{3.3}$$

$$o_i = \sum_{j=1}^{n} \alpha_{i,j} \cdot W^V v_j \tag{3.4}$$

Here, W^Q, W^K, W^V are trainable weight matrices, *attention* is a function mapping projected queries and keys to a scalar and softmax is the softmax function over all $k_j \in K$. For example, [57] uses the dot-product scaled by the dimension of the keys d_k:

$$\text{attention}(q, k) = 1/\sqrt{d_k} \cdot q_k^T) \tag{3.5}$$

The attention weight $\alpha_{i,j} \in [0, 1]$ controls how much attention is paid to v_j in output o_i.

The Transformer uses two extensions of this mechanism. Self-attention, i.e. $Q = K = V$, allows learning dependencies between elements of one sequence, comparable to the effect of recurrent connections in RNN encoder models. Additionally, self-attention facilitates learning long-range dependencies [57], which is particularly valuable due to the long length of the dialog context encoding of our model. Furthermore, self-attention offers greater parallelizability than recurrent models and thus significantly lower training times [57].

Secondly, multi-head attention allows the model to focus on different features or multiple parts of the same input sequence by calculating multiple attentions, as described above, in parallel.

3.2.2. Functional Principles of the Encoder and Decoder

To better understand how this mechanism is used in the Transformer, we now review the functional principles of the encoder, followed by those of the decoder.

The encoder first embeds the input sequence word by word in a learned vector space before adding a positional encoding to each embedding. This is necessary because the self-attention layers would otherwise not have access to position information. Following the embedding step, the encoder layers extract increasingly complex features from the input sequence using self-attention followed by a position-wise feedforward layer.

Similarly to the encoder, the decoder embeds decoder outputs from previous decoding steps. The decoder differs from the encoder only in one key way. In each layer, the decoder combines the encoder features with decoder features using attention. The decoder steps are used as the query and the encoder features as key and value of this attention mechanism. This allows the model to focus on the encoder features that are most important to predicting the next token of the decoder sequence.

Finally, a simple classification layer with each class representing one word of the output vocabulary is used to predict the next token. After the next token is predicted, the decoder is called again until a special end-of-sentence symbol is emitted.

Note that this model uses a constant inner dimension to facilitate the residual connections around the inner layers of both the encoder and decoder. In contrast to [57], we chose $d_{\text{model}} = 32$ (in the following d_{w}) as the model size and 64 as the inner size of the position-wise feedforward network. Additionally, we used only $N = 2$ layers in both the encoder and decoder. For further details of this architecture, the reader may refer to [57].

3.3. Integration of Scene-Awareness into the Transformer

In the following sections we discuss the integration of scene-awareness into this model. First, we present how the encoder uses a convolutional neural network (CNN) and attention to encode scene features. Subsequently, we describe how these features are integrated into the decoder and the target selection subnetwork. Finally, the training and inference procedures for the whole network are presented.

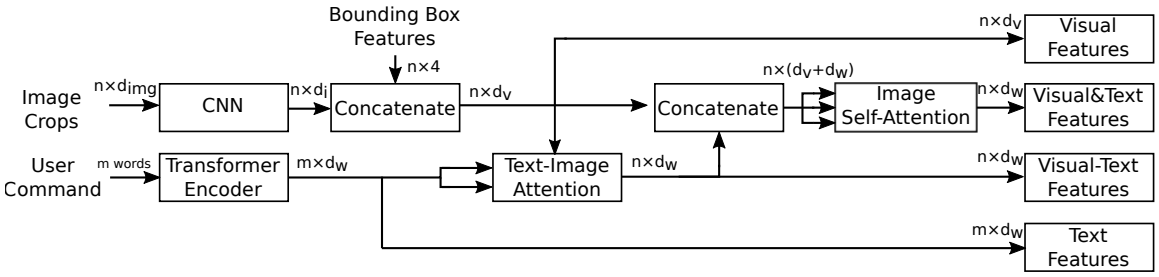


Figure 3.3.: **Modified Encoder**

The modified encoder combines visual and textual features obtained from a CNN and the standard Transformer encoder into four feature types: Pure visual features, multi-modal visual & text features, text feature attention guided by visual features and pure text features

3.3.1. Modifications to Encoder

The modifications to the encoder aim to extract features from the added image modality and relate them to the existing text features. The modified encoder shown in Fig. 3.3 achieves this by calculating several combined features, and replaces its counterpart in the Transformer.

To achieve the first goal, extracting features from the input images, a feature extractor is applied to each image crop separately. Naturally, a CNN was chosen for this task. While pretrained CNNs were considered, empirical tests showed that these networks did not perform well on our data set. We suspect that a difference in relevancy of different feature types makes the transfer of knowledge to our task difficult. In particular, color features are significantly more relevant in our scenario than they are in commonly used image tasks like object classification or detection that require more intricate edge detection features.

Instead, we train a CNN from scratch. Due to the limited amount of image training data available to us and to avoid overfitting, a relatively shallow network based on a three layer ResNet [21] architecture[2] was designed. The full details of this layer are provided in section A.1.2 of the appendix. To facilitate multi-modal feature combination, the final features after the last pooling layer are projected to a $d_{\text{image}} = 64$ dimensional vector. Afterwards, visual features are concatenated with bounding box features to obtain what we call image features. In our model, bounding boxes are encoded as a 4-tuple $(x_{\text{center}}, y_{\text{center}}, \mathit{width}, \mathit{height}) \in [0, 1]^4$. Each dimension of the bounding box features is scaled independently to the interval $[0, 1]$ so that the encoding is independent of the image resolution. Before these features are concatenated with the visual features, the bounding box vector is repeated $d_{\text{image}}/4$ times to balance the relative weight of both feature types, especially at the beginning of training.

After encoding both the image features and the textual features using the visual decoder and the standard Transformer encoder respectively, multiple combined features are calculated. To allow the model to relate visual information to the textual command, an attention, that we call text-image attention in Fig. 3.3, between the text and each image is calculated. As this attention sums over textual features, the model can use this component to select text features important to each of the image crops.

[2]We use torchvision's ResNet implementation

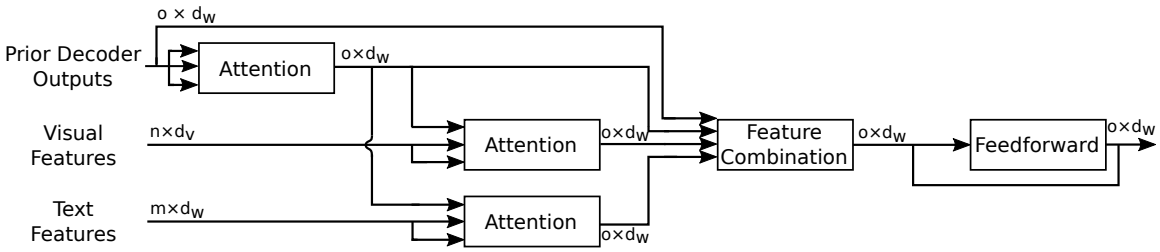


Figure 3.4.: **Modified Decoder Layer**

The modified decoder layer incorporates visual and textual information into its prediction of the next word of the output sequence using multiple attentions. Even though no multi-modal features are added to this part of the model, subsequent decoder layers allow the model to relate both the textual and visual features. Unless specifically mentioned, you can assume that we retained all details (e.g., post-processing) of the original Transformer [57] as implemented in https://github.com/quanpn90/NMTGMinor.

The third component, image self-attention, uses both the image-attended textual features and visual features to calculate relation features between objects. An example for a command where these features are particularly important is „*Fly to the person in the green shirt next to the person wearing a blue suit*“. Note, that this self-attention layer is equivalent to the fully-connected relation-aware graph attention proposed by [34] with the only difference being the use of a different attention function.

All four of these features are then used in the decoder and target selection subnetworks to predict the output action-parameter sequence and the target of the command.

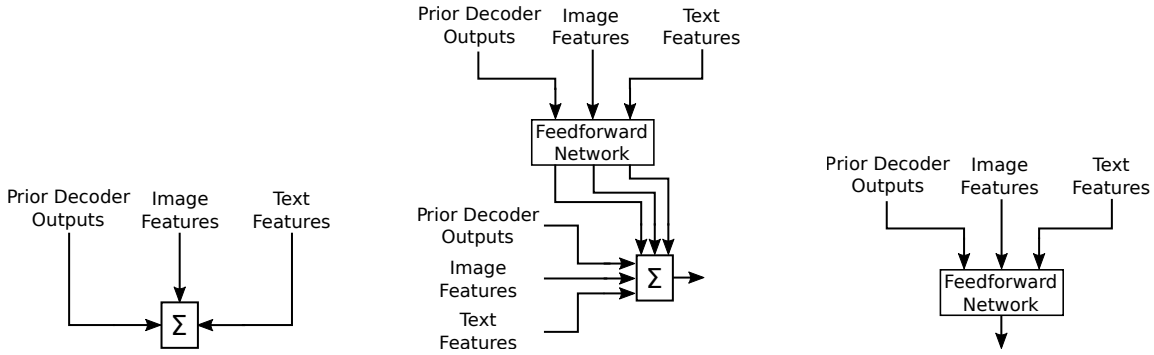
3.3.2. Modifications to Decoder

The modifications of the decoder are mainly concerned with integrating the additional encoder features and combining them with prior decoder steps. For this purpose, we modify the decoder layer as presented in Fig. 3.4.

As shown in this figure, only two of the four features produced by the encoder have been integrated. This decision was motivated by empirical tests that showed no improvement and in some cases even degraded performance when additional features like the *Visual-Text features* from Fig. 3.3 were added. Note that the model is nonetheless able to calculate relations between both visual and textual modalities in subsequent layers. Thus, adding these features to the decoder explicitly is not necessary.

Compared to the standard Transformer decoder, two major architectural changes have been made. One, an additional attention over the image features guided by the prior decoder outputs was added similar to the combination of textual features from the encoder. Secondly, a feature combination layer has been introduced.

The three different approaches to implementing this layer shown in Fig. 3.5, have been considered. The first approach 3.5a follows the convention used in the standard Transformer model and calculates a sum of the three features. Generalizing this approach, 3.5b calculates a weighted sum of features similar to attention. However, instead of calculating similarities as in [25], a feedforward network with all features as inputs is used to predict one attention weight per feature. This allows the model to focus on one or more of the modalities. Lastly, in 3.5c, the three features are simply concatenated and projected



(a) Combination: Sum (b) Combination: Multi-Modal Attention (c) Combination: Projection

Figure 3.5.: **Feature Combination Types**

Three different feature combination types are proposed . (a) follows the approach of [57] uses summation to combine multiple features. (b) generalizes this approach and uses multi-modal attention [26]. (c) allows the model to calculate an arbitrary projection from the features to the Transformer's model size.

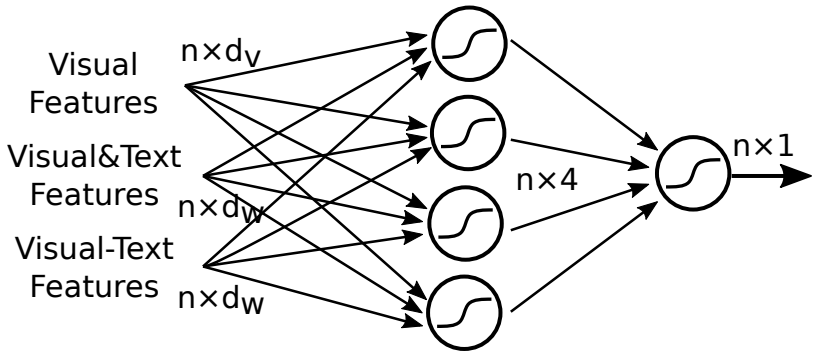


Figure 3.6.: **Target Selection Network**

The target selection network is a position-wise feedforward neural network [57] which uses a sigmoidal output unit to predict the likelihood of each object being the target. The final selected target is then chosen by finding the maximum of all objects. Note that target selection is only performed if the dialog component emitted a *goal* action.

to a d_{model}-dimensional vector space. An evaluation of these three methods with respect to overall system performance is provided in chapter 5.

3.3.3. Target Selection

Finally, the model needs a way of identifying the intended target of a command. This task can be understood in two different ways with implications for the design of the output layer of this component. On one hand, one can understand this task as a selection task which tries to choose the target object out of a set of input objects. On the other hand, this can also be viewed as predicting the likelihood of one specific object being the target. Additionally, we have to take into account that this component may receive a varying number of objects.

This in-fact makes the latter of the two interpretations better suited because the model should decide if a person is the target of the current dialog step. For example, if there is only one person, the solution to the first task definition would incorrectly be that that person is the target. In the second interpretation, the policy learning component is able to reject targets with low confidence and ask the user for clarification. Thus, the robustness of the model against false positives is improved.

To enable the model to deal with varying numbers of objects and improve robustness, we extend the idea of position-wise feedforward networks [57] as shown in Fig. 3.6. In this architecture, each object is passed through the same network independently. Thus, if the task of the network requires knowledge about relations between these objects, features encoding these relations have to be added. In our case, the visual & text features produced by the encoder play this role. Additionally, the target selection model can use visual features and text features of the current object for its prediction.

The two layer feedforward network used for this prediction consists of 4 neurons in the hidden layer, a ReLu non-linearity in between layers and a sigmoidal output unit. Thus, the network predicts the likelihood of the input object being the intended target in the current dialog step, matching the definition in the introduction of this section. For comparison, a Softmax activation over all objects as the output layer would be equivalent to the first interpretation of the target selection task. We also tested models with larger hidden layer sizes but did not observe any increase in prediction performance.

The final selected target is then chosen as the maximum over the output of the network for all objects. Note, that this selection is only used if a goal action, that has a target, was predicted by the decoder.

3.4. Training

To train the model presented in the previous paragraphs, two loss functions, one for the decoder and one for the target selection head, have to be combined. In this section, we review a simple approach to balancing these two loss functions for efficient joint training.

As in [57], we use the standard cross entropy criterion as a loss function for the decoder in each decoding step:

$$E_{\text{dialog}}(x) = -\sum_{i=1}^{n}\sum_{c\in V} y_{c,i} \cdot \log\left(p_{c,i}\right) \tag{3.6}$$

Here, n is the number of words in the output sentence x, V is the output vocabulary, $y_{c,i}$ the binary ground truth indicator for the i^{th} word of x being the word c, and $p_{i,c}$ the predicted likelihood of c. From hereon, $E_{\text{dialog}}(x)$ will be called dialog loss.

Similarly, we minimize the binary cross entropy loss for the target selection network:

$$E_{\text{target}}(x, P) = \begin{cases} -\sum\limits_{i\in P} \left[y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)\right] & \text{if } x \text{ has a target} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

Here, P is the set of objects, y_i is the ground truth target indicator and p_i the likelihood predicted by the target selection network for person i. If no target is defined for a specific dialog context x, the target error is set to 0.

To combine both of these losses, we have to pay close attention to their scale and their relative importance to our task. $E_{\text{dialog}}(x)$ is generally far larger than $E_{\text{target}}(x, P)$ because the dialog loss consists of one cross entropy criterion per output word. Thus, to guarantee that the model considers both losses equally, we balance the sum of the losses for one batch of dialogs X and corresponding objects T as follows:

$$E_{\text{total}}(X, T) = (m + n) \cdot (\frac{\alpha}{m} \sum_{i=1}^{|X|} E_{\text{dialog}}(X_i) + \frac{\beta}{n} \sum_{i}^{|X|} E_{\text{target}}(X_i, T_i)) \tag{3.8}$$

where

$$m = \sum_{i=1}^{|X|} |X_i| \tag{3.9}$$

$$n = \sum_{i=1}^{|T|} |T_i| \tag{3.10}$$

Here m and n are the number of words and targets in all dialogs respectively. α and β can be used to further tune the importance of each of the two losses. Note that we did not observe any significant performance impact for various values of α and β and thus decided to use $\alpha = 1$ and $\beta = 2$ to place a slightly higher focus on target selection.

Furthermore, to reduce overconfidence and improve generalization, we applied label smoothing [55] with a value of $\epsilon_{ls} = 0.1$ to both losses during training. We used Adam [29] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$ as the optimizer and updated the learning rate over the course of the training according to Equation (3) of [57] with warmup steps set to 200 and the initial learning rate equal to 0.05 or 0.01.

3.5. Limitations

The design presented in this chapter imposes certain limitations on the kinds of dialogs that can be learned and the transferability to new domains. This section discusses these limitations in more detail than some of the brief remarks given throughout this chapter provided.

A first limitation compared to other models [68, 11] is that the question selection process is not integrated end-to-end. Hence, it is not possible to optimize for the length of dialogs or other criteria such as the quality of selected questions. Accordingly, we expect the selected questions to be less goal-oriented than those of approaches that optimize this process. Furthermore, this also leads to comparably longer dialogs on average. This limitation, however, can be eliminated by replacing the question selection subsystem with a learnable component trained with reinforcement learning or supervised targets.

Furthermore, the model requires a way of handling commands that do not require the presence of objects. Without objects in the scene, no image features can be calculated

and it becomes very difficult for the model to make accurate predictions. We tackled this problem by introducing a special, completely black image crop that is used whenever no objects are present in the scene. While we expect this to not have a significant impact on performance, it would be desirable to avoid this workaround because it requires the model to be trained with modified training data.

The deployment of this model in new use cases and domains also presents unique challenges. data set availability, in particular in the area of visual dialog systems, is problematic, especially if labeled data is needed. It is thus necessary to find solutions to generating the large amounts of data required to train supervised systems like the one presented in this thesis. In this work, this limitation was overcome by expending a significant amount of engineering effort on generating dialogs for many potential scenarios and ensuring their correctness. This is of paramount importance because any error in the generation process immediately results in numerous incorrect dialogs. Note, that while data set engineering works great for well-defined tasks, it may not result in a good representation of the real world. Thus, real-world performance may be lower than indicated by experiments on such a data set.

Similarly, even if data set engineering is used, the transferability of this system to new domains is difficult. For every new domain, the action and parameter space has to be manually redefined, which, depending on the complexity of the dialog system, may require many hours of careful action design. This, however, presents only a minor caveat, as task-oriented dialog systems must eventually map their output to API calls.

4. Drone System

To demonstrate the capabilities of the dialog system presented in the previous chapter, a custom quadrotor with appropriate sensors and control software for the person following scenario introduced in chapter 1 was designed. This system is also used for the field test in chapter 5 to measure overall system performance.

In this chapter, the system design and control software are presented with a focus on the high-level components specifically developed for this thesis. To make this thesis self-contained, we briefly introduce the low-level controllers. Note that most of these controllers were developed at Carnegie Mellon University's (CMU) AirLab by various authors, and only minor modifications have been made. Furthermore, most of the hardware design was also developed at CMU.

4.1. Hardware

In this first section, we review the hardware components[1] and sensing capabilities of the quadcopter. To give a better impression of the system and the location of all sensors, it is shown from multiple angles in Fig. 4.1.

The drone hardware is based on Lumenier's Student Drone Kit[2]. In addition to including almost all physical components required to build a fully functional quadrotor, the frame has built-in propeller guards, making it significantly better suited for indoor use than other comparable frames without this feature. The frame also features a wide variety of different mounting holes that facilitate the installation of additional components.

For low level control, a Pixracer running the Open Source PX4 Autopilot[3] firmware is used. This component is responsible for tracking the desired *Yawrate*, *Pitch angle*, *Roll angle* and *Thrust* which it receives from the onboard autonomy controller.

The autonomy controller runs on a Nvidia Jetson TX2 with 6 CPU cores and an integrated Nvidia GPU. Additionally, the autonomy controller has access to two onboard sensors: a tracking camera, responsible for localizing the robot relative to its starting position, and a depth camera used for obstacle avoidance and person detection.

For optimal operation, the tracking camera must achieve high tracking precision and robustness to temporary occlusions. The Intel Realsense T265 was chosen for this purpose because it fulfills these requirements and in addition performs all processing for visual simultaneous localization and mapping (V-SLAM) on-board the camera. Thus, valuable resources on the TX2 are freed up for other tasks. Note that the camera was mounted on a vibration dampened platform facing downward at a 45° angle to significantly reduce

[1] A full list can be found in section A.2.1 of the appendix

[2] https://www.getfpv.com/student-competition-5-bundle.html

[3] https://px4.io/

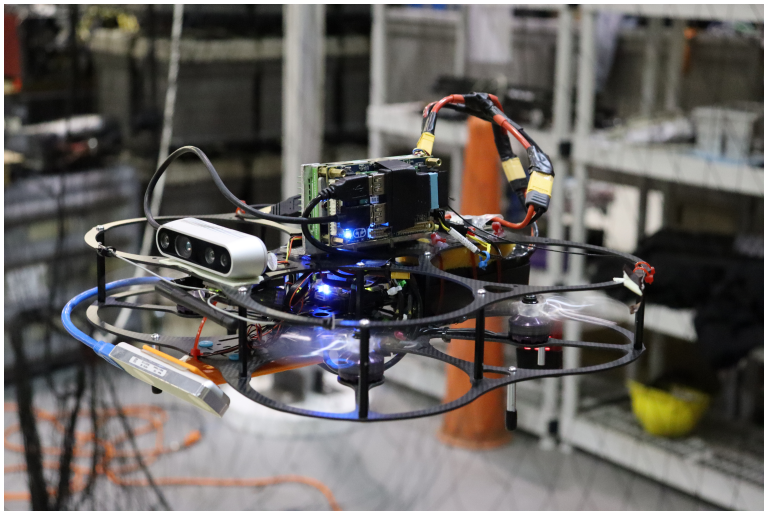


Figure 4.1.: **Drone Platform**
The quadrotor used for the person following task provides various sensors including a tracking camera, depth camera and IMU. Additionally, an on-board computer provides sufficient processing power for sensor data processing and flight control.

drift over time. Furthermore, a separate USB Y-cable powered by the on-board battery was connected to the T265 because the TX2 carrier board is not able to supply sufficient power for both cameras.

For similar reasons, the Intel Realsense D435 was selected as the depth camera. It provides accurate, although in parts noisy, depth images suitable for obstacle avoidance. Furthermore, it records RGB and Infrared images that can be used to detect and track persons in the field of view of the drone. The Intel RealSense driver additionally takes care of most low level processing such as extracting a pointcloud from the depth image, and aligning the rectified depth image to the color image.

Combined, these sensors and the Pixracer flight controller form the foundation on top of which all other components build.

4.2. System Overview

These components can be grouped into two categories, Flight Control and Mission Control, as shown in Fig. 4.2.

Components in the first category are responsible for all low-level control loops, including velocity control, position control, and trajectory tracking, and were in large parts developed at CMU's Airlab. For the most part, these components use standard approaches for robot control. Thus only a brief description of their interdependencies and implications for the system design and performance are provided in the first subsection.

The second category contains components concerned with high-level functions such as human-robot interaction, target detection and following. Most of these components were specifically developed for the person following task. Consequently, in the second part of this chapter, we highlight design decisions relating to these components in detail and also

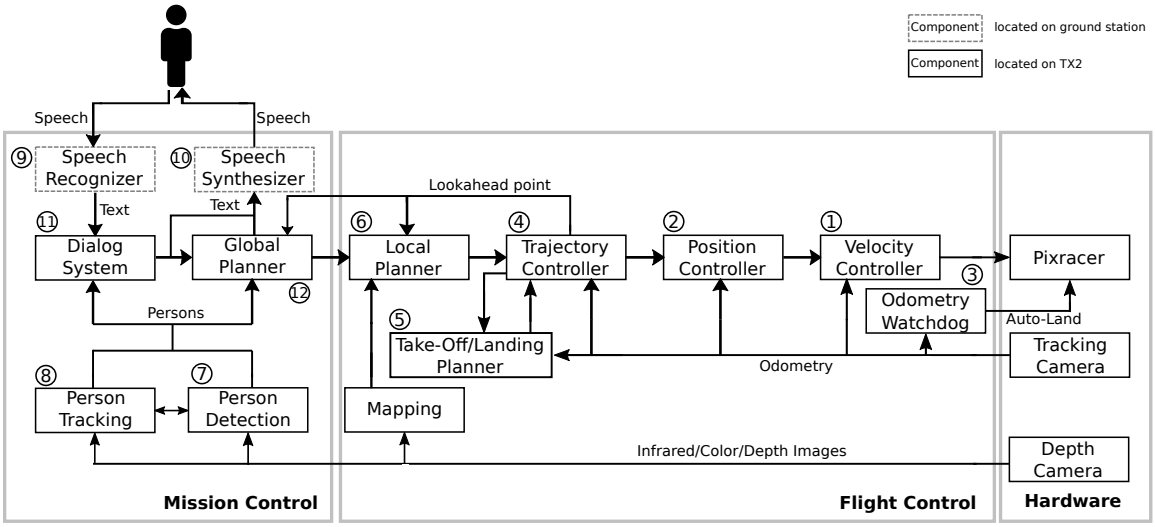


Figure 4.2.: **System Overview**

The system uses a series of successive control loops which extend the capabilities of the system step by step to enable the drone to autonomously complete person following missions. Note, that some components are executed on the accompanying ground station to reduce transmission overhead.

provide insights into integrating the dialog system presented in the previous chapter into a real robot.

4.2.1. Flight Control

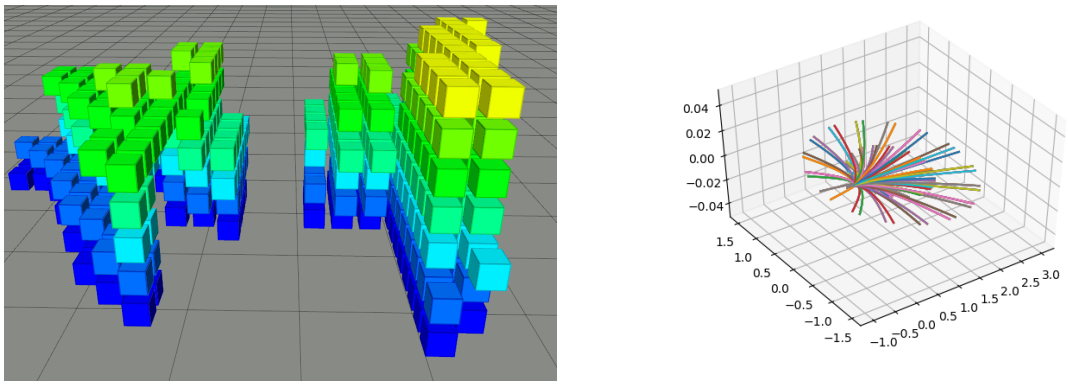
The flight control system uses a series of closed-loop controllers to enable the drone to follow trajectories. In the following section, we present these control loops going from low-level control towards more sophisticated controllers.

On the lowest level, two closed-loop controllers, one for velocity (1) and one for position (2) are used. They receive the odometry estimate from the tracking camera as feedback and aim to achieve a desired velocity and position, respectively. In the case of the position controller, the desired position (x, y, z, yaw) is converted to appropriate velocity commands. Similarly, these commands are then used by the velocity controller to send desired yaw rate, thrust, roll and pitch angles to the Pixracer.

Both of these components are implemented as PID controllers. PID controllers are a standard technique that determines a control signal based on the deviation of the actual state from the desired state. To calculate the control signal, three terms with respect to the signed state error E are combined:

- **P**roportional Control: $c_p = \kappa_p \cdot E$
- **I**ntegral Control: $c_i = \kappa_i \cdot \int_t E dt$
- **D**erivative Control: $c_d = \kappa_d \cdot \frac{dE}{dt}$
- Constant offset: c_o (manually tuned)

Here, t is the time and $\kappa_p, \kappa_i, \kappa_d$ are manually tuned constants. The control signal c is then calculated as $c = c_p + c_i + c_d + c_o$. For our drone, it was sufficient to use only



(a) Map (b) Trajectory Library

Figure 4.3.: **Grid Map and Trajectory Library**
The local planner uses the grid map (a) and the trajectory library (b) to plan a trajectory which follows the global plan. Note that the trajectory library currently uses 498 trajectories in the x-y plane only but can be easily extendeded to 3D trajectories.

proportional control for all tracked values except for velocity in z-direction. The z-velocity controller used an additional integral term to compensate for declining effective thrust at the same thrust setting as a result of battery discharge, and a constant offset set to a value slightly below hover thrust.

On the same hierarchy level, a watchdog (3) observes odometry messages from the tracking camera. If it receives no odometry for a certain amount of time, it instructs the PX4 to land the drone immediately. This was necessary to ensure the safety of the drone and persons in its vicinity due to infrequent crashes of the tracking camera with no discernible pattern.

The next control loop, the trajectory controller (4), uses the position controller to follow a trajectory with respect to position and velocity. The trajectory controller uses two points called tracking and lookahead point to follow the desired trajectory. The tracking point represents the location the drone should have at the current time t according to the input trajectory. Similarly, the lookahead point is a point in the future at time $t + \Delta$ on the input trajectory. From this point, all higher-level components can replan the trajectory as needed, i.e., the trajectory between tracking and lookahead point is fixed and currently being executed. In our system Δ is set to 0.1 seconds.

The local trajectories are planned by one of two local planners. First, the take-off and landing planner (5) plans a very simple trajectory from the current tracking point to a certain height with a fixed rate of ascent or descent. No further checks for obstacles are performed as both take-off and landing are triggered manually. This is necessary because the drone cannot detect obstacles directly above or below due to a lack of upwards and downwards pointing sensors.

In contrast, the second planner, which is called local planner (6), integrates obstacle avoidance into the trajectory selection process. It uses a grid map that is generated from the point cloud of the depth camera, to find obstacle-free paths in a set of predefined trajectories. A visualization of this map and the trajectory library are shown in Fig. 4.3.

Note that in addition to the probabilistic filtering used to generate the grid map in [58]'s approach, we also use a simple $O(n)$ approach to marking grid cells as occupied based on the number of observations in each cell.

To be able to check the large number of potential trajectories in this library in real-time, an efficient, two-phase algorithm for collision checking based on bitwise operations was devised by [58] at CMU's Airlab. In the offline phase, a bit is set in each cell a trajectory passes through. This bit signals that a cell must be unoccupied for this trajectory to be obstacle-free. In the online phase, two checks are performed. First, the bitset is used to find all collision-free trajectories using bitwise operations. Afterward, a cost function is used to determine which of the valid trajectories should be chosen. Finally, a velocity profile for the selected trajectory is determined such that the velocity at the end of the trajectory is 0 and a maximum velocity is not exceeded at any time.

Here, we deviate from [58]'s cost function implementation. In [58], trajectories are selected based on the closeness of the last trajectory point to the next goal location. However, this selection process is biased towards flying at an angle to the direct path to the goal. This behavior is caused by the end of the straight trajectory being closer to the goal than all curved trajectories that follow the global plan more closely. Thus, the flown path is often longer than necessary.

To alleviate this issue, we combine two measures in the cost function. First, to favor trajectories with a long planning horizon over shorter ones, we calculate the closest distance of the trajectory to the goal location. Intuitively, this assumes that the drone can stop at any point on the trajectory. While this assumption does not hold for fast-flying UAVs, it is valid for slow-moving drones like ours. Second, to punish deviations from the global plan, we measure the average distance of the trajectory from the global plan. Finally, the cost function is calculated as the sum of these measures weighted by a manually tuned constant that balances the relative importance of progress towards the goal and closeness to the global plan.

Combined, these components allow the drone to autonomously follow a global plan defined by waypoints while avoiding collisions with obstacles.

4.3. Mission Control

Based on this foundation, the capabilities of the UAV are extended by implementing additional components for the execution of missions. A major challenge during the implementation of these components was the limited computational power provided by the TX2 that forced us to invest significant engineering effort into optimizing the CPU usage of mission control components. In the following sections, we give an overview of these modules and their optimization. Additionally, we describe how the drone and the user interact. We first present supporting functionality, including person detection & tracking, and the communication interface. Lastly, the two core components, the dialog system and global planner, which are responsible for mission-level drone control, are discussed.

4.3.1. Person Detection & Tracking

Person following requires the quadrotor to be able to detect, track, localize and reidentify persons. In this section, we present an approach which combines these steps into two intertwined components.

For detection and tracking two different approaches exist: Detection-by-Tracking and Tracking-by-Detection [2]. In Tracking-by-Detection, the object is detected in the image in every frame. By contrast, Detection-by-Tracking uses a model of the tracked object (e.g., visual features and a motion model) to follow the object in consecutive frames. Both approaches have different trade-offs. While Detection-by-Tracking is generally speaking faster than running a full detection every frame, it deals poorly with occlusions and often suffers from drift over time. Detection-by-Tracking avoids these problems. However, false-negative detections can lead to constantly appearing and disappearing detections. Thus, both techniques are combined in this work to create a tracker that is able to benefit from the advantages of both methods while avoiding their shortcomings.

Detection
For person detection (7), a pretrained MobileNet [27] for pedestrian recognition[4] is used to predict 2D bounding boxes around persons in the color image. Even though this network is optimized for embedded platforms, each forward pass takes about 800ms on the TX2's graphics card. This presents a significant challenge to tracking persons in real-time.

Tracking
Thus, a person tracking component (8) with a kernelized correlation filter[5] (KCF) [23] in combination with a simple motion model is used. The KCF tracker updates an object's position every time a new infrared frame arrives until new evidence in the form of a detection is received. Here, we use the infrared image because it is aligned to the depth image by design - this will later simplify calculating the 3D location of tracked persons. In addition, using the infrared image instead of the color image did not seem to have any adverse effect on tracking performance. Furthermore, if the person leaves the frame, a simple motion model using only the 3D position and velocity of the tracked object is used to update the object's location for a couple of seconds. Note that the acceleration of tracked objects was not used due to the low frame rate of depth images ($\approx$ 6 Hz) and high noise in the 3D locations, which makes accurate estimation of acceleration practically impossible.

Realistically, the motion model gives only a very rough estimate of the person's whereabouts. Nonetheless, it improves the person following performance of the drone because the drone is able to continue following the target even if it leaves the frame temporarily. This happens relatively frequently, if a person walks around a corner or if the drone is not able to rotate as fast as needed to keep the tracked object in the frame. This is comparable to how humans can extrapolate person locations for a short period of time.

Localization
For the motion model described previously, and to be able to localize the target, the 3D

[4] https://github.com/cftang0827/pedestrian_recognition
[5] We use openCV's KCF implementation

position of each person has to be determined. Although the color and infrared image on their own do not help solving this, the depth image of the D435 depth camera can be used to reconstruct the 3D position.

Given the 2D bounding box in the depth image, we first sample multiple points in a grid centered inside the object's bounding box. Outliers are then removed from these samples by removing the lower and upper 10% percentile. Finally, the average depth d is used to reconstruct 3D points from the top left (x_t, y_t) and bottom right (x_b, y_b) of the 2D bounding box by calculating the inverse (Eq. 4.2) of the camera projection equations (Eq. 4.1):

$$project((X, Y, d), (f_x, f_y, c_x, c_y)) = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \end{pmatrix} \begin{pmatrix} X/d \\ Y/d \\ 1 \end{pmatrix} \tag{4.1}$$

$$deproject((u, v), d, (f_x, f_y, c_x, c_y)) = \begin{pmatrix} X \\ Y \\ d \end{pmatrix} = d \cdot \begin{pmatrix} \frac{1}{f_x} & 0 & 0 \\ 0 & \frac{1}{f_y} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{4.2}$$

Here (c_x, c_y) is the principal point and (f_x, f_y) are the focal lengths of the rectified depth camera. Also note that the depth reported by the D435 is the distance from the camera plane, i.e., equivalent to the z coordinate in the 3D camera frame. Thus, the 3D location corresponding to a pixel (x, y) can be directly calculated by inverting the camera projection as shown in Eq. 4.2. The location data is then further processed to estimate the velocity of all tracked objects and is passed to the global planner.

The above calculations are correct as long as the image in which the 2D bounding box is detected and the depth image share the same frame of reference. For the D435, this is only true for the depth and infrared image. For all other images, a mapping of pixels into the depth image frame is required. The standard way of obtaining this mapping provided by the depth camera driver maps all pixels from the depth image to the color image as shown in Fig. 4.4. However, we only require the depth for a small subset of pixels. Thus, a lot of processing power (~15% across all 6 CPUs of the TX2) would be wasted by calculating this mapping. Instead, we opt to follow a similar approach with a simplification that significantly reduces CPU utilization in exchange for slightly lower accuracy.

We achieve this by reversing the deproject-transform-project workflow of the camera driver as shown in Fig. 4.4. This direct mapping is only possible because we assume that the extrinsic transform T_{d2c} is the identity matrix. Without this assumption, this process could not be reversed because d_c would have to be known. Note that this assumption does not introduce significant inaccuracies because the extrinsic matrix for the D435 from color to depth is essentially a translation in x-direction by only 1.5cm.

Given a pixel $p_c = (u_c, v_c)$ in the color image, one can then calculate the corresponding pixel p_d in the depth image as follows:

$$p_d = project(T_{c2d} \cdot deproject(p_c, d_c, I_c), I_d) \tag{4.3}$$

$$\Leftrightarrow p_d = project(deproject(p_c, d_c, I_c), I_d) \tag{4.4}$$

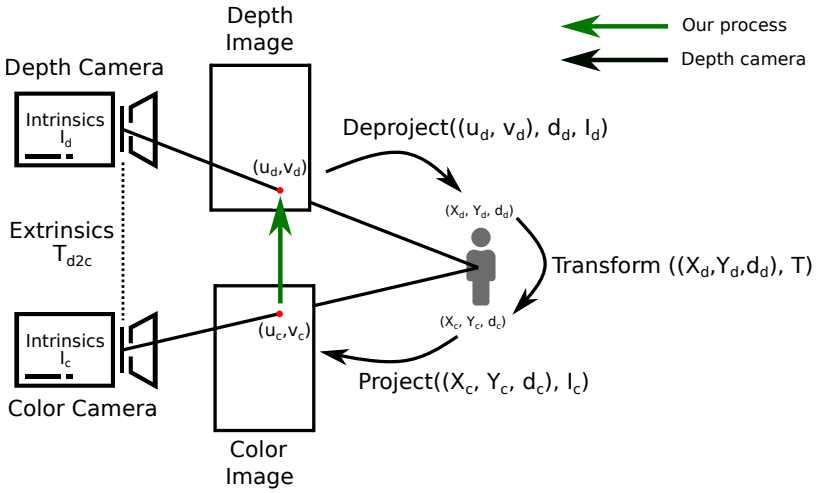


Figure 4.4.: **Deproject-Transform-Project Workflow for Alignment of Depth Image with Color Image**

Substituting the camera equations 4.1 and 4.2 into Eq. 4.4, with intrinsics marked with superscript c and d for color and depth camera respectively, admits:

$$p_d = \begin{pmatrix} f_x^d \cdot (\frac{u - c_x^c}{f_x^c}) + c_x^d \\ f_y^d \cdot (\frac{v - c_y^c}{f_y^c}) + c_y^d \end{pmatrix} \tag{4.5}$$

This allows us to map the 2D bounding boxes detected in the color image to the depth image without the knowledge of the world z-coordinate. Consequently, we can calculate the location of the person in the world space with equation 4.2.

Person Reidentification

Lastly, the task of person reidentification remains. This is an important part of person following because occlusions or other persons crossing between the drone and the tracked target may cause tracking to be lost. For this purpose, a MobileNet-based CNN, which acts as an embedding extractor for persons, is used. This embedding vector belongs to a vector space in which euclidean distance can be used to measure the similarity between persons. The model learns this relation through a special loss function called Triplet Loss [47, 24] that minimizes the distance between an anchor and examples of the same class while simultaneously maximizing the distance between the anchor and negative examples. Thus, person similarity can be determined using the learned embedding and used as a measure for reidentification. In addition, we use the motion model and 3D location data to find missing predictions and reidentify persons not picked up by this first method. This additional check simply provides a sanity check for the identities provided by the network and helps correct some rare issues where the network has difficulties differentiating between physically distant persons.

4.3.2. Human-Robot Communication

The human-robot communication interfaces consist of an automatic speech recognizer (ASR) (9) and a speech synthesizer (10). Both of these components run on the accompanying laptop, but could also be deployed on any other device due to the relatively low computational resources consumed by the speech recognition system.

The user uses their smartphone through a simple app or the laptop's microphone to send verbal commands to the system. The speech signal is then processed by the Janus Speech Recognition Toolkit [33, 41]. We adapted this model to improve the speech recognition accuracy on our task by merging the existing language model (LM) with one trained on the training data described in chapter 5. In essence, this improves recognition performance on our task by giving Janus access to prior information about the likelihood of encountering certain word sequences. No further modifications were necessary to achieve good recognition rates. Note that perfect recognition is not necessary for our approach because the dialog system can to some degree deal with misrecognitions.

The second component of the communication interface is the speech synthesizer which translates text emitted by the mission control system to speech. This way, the user is informed about all major system decisions and requests for assistance. As our system emits grammatically correct English sentences, no special modifications for the person following scenario are needed. Hence, an external, publicly available component is used.

Together with the user, the speech recognizer and synthesizer form a control loop which we dub „human-in-the-loop". This setup allows the user to keep up to date with all actions the drone is performing while being able to intervene whenever the drone requests assistance or the human operator determines that a system decision needs to be corrected.

4.3.3. Dialog System

The dialog system (11) is the core component used to understand the user's requests in the context of the scene and past dialog. As input, it receives the last user utterance from the ASR and the set of persons visible to the drone.

Based on this input, it then uses the dialog system described in the previous chapter to determine the next system response. This response consists of a textual response and possibly a mission description. The textual response, which can include a question, is then relayed in verbal form to the user via the speech synthesizer.

If the user command contains a request for the execution of a mission (i.e., a goal action was emitted by the dialog system), the action including its parameters is forwarded to the global planner for mission planning and execution.

This design allows the dialog system to rely on the user to clarify commands until it is certain that it understood the mission's objective correctly. It can also be easily extended to allow for predictive execution of missions. In this case, the dialog component would relay information about the mission target to the global planner while it is still in the process of confirming the final user intent. This is particularly useful if partial transcriptions of the user's utterance can be sent to the system. Unfortunately, the android app available for Janus does not support this use case and thus, this extension is left for future work.

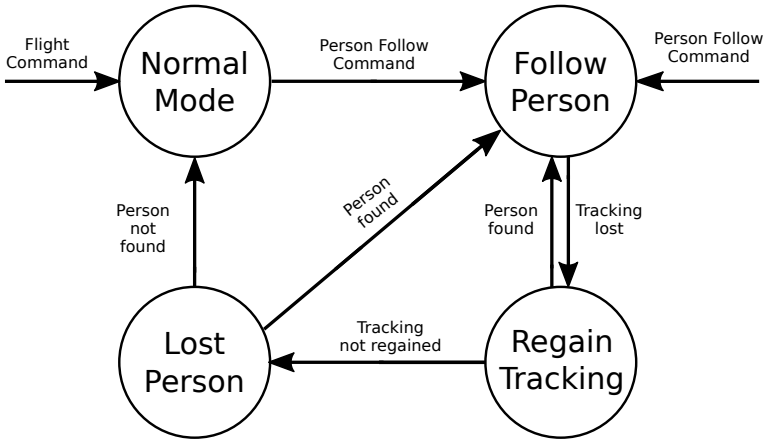


Figure 4.5.: **Global Planner State Machine**

The global planner uses a simple state machine to execute user commands. In particular, the person following command possess a failure cycle in the state graph. Throughout this cycle, different strategies are employed to reidentify targets after tracking is lost. Note that this state machine is reset every time a *Flight Command* or *Person Follow Command* is received as indicated by the start state markers.

4.3.4. Global Planner

The global planner is responsible for the planning and execution of received missions. It also contains strategies for recovery from failures.

Mission planning uses a straightforward approach. If a flight command (e.g., „Fly a bit forward“) is received, the global planner simply creates a global plan with one waypoint containing the target of the command. A similarly simple global plan is used for person following. The global planner receives the target person's location and then calculates a straight line path from the lookahead point to the target. To make sure that the drone stops at an appropriate distance, the vector between drone position and the target location is shortened by 0.5 meters. Under the assumption that a line of sight exists between the drone and the target, this simple procedure is sufficient to approach the targeted person. As the drone must be able to detect a person in the field of view of its camera for it to be chosen as the target, this assumption holds. In a more complex scenario, a global planner that uses the grid map and for example, the A^* algorithm with an iterative replanning strategy would have to be designed.

During the execution of a person following mission, it is possible that target position updates fail. For example, the detection and tracking could suffer from a false negative, or the drone loses sight of the target. In this case the state machine shown in Fig 4.5 tries to recover tracking using a human-inspired approach.

To understand this approach, assume that a human observer lost track of an observed target. In this case, the observer would first look towards the last known position, exploiting the fact that the set of possible locations grows circular from the last known position over time. If this is not successful, the observer would start to expand the search area and look further to the left or right of the last location and potentially also check the area behind them.

The state machine implements a very similar behavior for the recovery of the target location. First, after no location updates have been received for several seconds, the drone cancels the global plan, informs the user and reorients towards the last known target location, mimicking the human observer's reaction. This helps regain tracking when the drone had to turn away from the target to avoid an obstacle or when detection failed for several frames, e.g., due to motion blur. If tracking is regained, the drone updates its global plan and continues normally.

However, if tracking is not reestablished after a set amount of time (five seconds in our system), the state machine enters the lost person mode. In this mode, the drone expands the search area. It first rotates 90° in the direction of the last velocity vector of the target. Afterward, it rotates 180° to the other side to check the opposite side. If the target is still not found, a last recovery attempt is made and the drone checks if the person is behind it. This is particularly important indoors, where the followed target can turn around and pass by the drone. Finally, if all of these steps were unsuccessful, the user is informed about the loss of target tracking and asked to issue their next order. This again uses the human-in-the-loop concept introduced at the beginning of this section.

The drawback of this approach is that it heavily relies on a line of sight towards the target. As long as one is close to the target and in a straight, spacious hallway, like we assume in our experiments, this should not present a significant challenge. However, if the building has a lot of turns and corners, a better approach may be to extract a local floor plan from the mapping component. This plan could then be used to build an improved model of human motion in a building to allow the drone to understand how it should continue. Regardless, this approach would require significant research efforts into improved environment understanding as well as autonomous decision making.

5. Experimental Evaluation

In this chapter, we present the experimental evaluation of the dialog and drone system. First, we describe how we acquired image and dialog pairs for the training of the dialog system. Afterward, we present three experiments. The first experiment uses a user simulator to evaluate how the system performs on synthetic test data. The second experiment reports on an online user study in which users interacted with the dialog system with little instruction. Lastly, a full end-to-end system test of the combined dialog and drone system is presented.

5.1. Data Sets

In order to train the dialog system introduced in chapter 3, a training data set, i.e., images and corresponding dialogs, need to be selected. However, only a few of the publicly available image data sets are suitable for this purpose, and none of them contain dialogs for our scenario. Thus, the three data sets shown in Fig. 5.1 were manually created using a two-step approach consisting of the collection and annotation of image data, followed by synthetic dialog generation. These two steps are described briefly in the following two sections.

5.2. Data Collection & Augmentation

Two methods were employed to collect image data. One, videos from a publicly available data set [46] originally intended for human interaction recognition, were used to form the

(a) SDHA (b) KIT (c) CMU

Figure 5.1.: **Three Data Sets**

Three different data sets were to train the dialog system. SDHA contains manually selected images from [46], while KIT and CMU were collected specifically for this work using Intel RealSense D435 cameras mounted on the drone. All three data sets were annotated manually with 2D person bounding boxes and the dominant color of each person's top (e.g., t-shirt, sweater, …).

Data Set	Training	Validation	Test
SDHA	110	27	34
KIT	163	41	51
CMU	154	38	48

Table 5.1.: **Overview of Images in Each Data Set**

SDHA data set (Fig. 5.1a). In contrast to the elevated perspective of this data set, the other two data sets, KIT (Fig. 5.1b) and CMU (Fig. 5.1c) were recorded from the perspective of an office drone assistant.

After this first collection step, frames were extracted from all videos at about 0.1 Hz and further filtered manually to remove too similar looking shots. In the next step, all selected images were annotated with person bounding boxes and the main color of each person's top (e.g., t-shirt, sweater, ...).

Combined, this process yielded a total of 666 images separated into the train, test and validation data sets using a 60%:15%:25% split as shown in Table 5.1.

Lastly, to increase the amount of training data, images were augmented using a random combination of the techniques described in the following[1]. First, all images were flipped and saved as additional data points. Afterward, the images were either scaled by up to 10% or rotated by up to 5 degrees. In essence, this models the typical changes of the perspective of the drone, e.g., a person coming closer and the roll angle of the drone. Furthermore, to mimic the noise from vibrations and movement of the drone, Gaussian noise, Gaussian blur, and motion blur were added. Lastly, the variability of colors was increased by applying one of three more extreme adjustments: Either the color of each pixel, the brightness, or the contrast of the image were modified in reasonable bounds.

In total, for each input image, eighteen augmented images were generated using a random combination of the augmentations described above. From hereon, appropriate dialogs are generated for each image.

5.3. Synthetic Dialog Generation

With no real dialogs being available for our data sets, a different approach has to be taken to obtain the needed training data. In our case, the solution to this problem is synthetic dialog generation according to a well-structured process to guarantee that as many as possible scenarios are covered.

We identified five different scenarios, that can reasonably be learned from our data set:

- Ignore: Command not directed at system
- Simple: Color/Direction of a person
- Combined: Color and Direction of one person
- Relative: Color/Direction of the targeted person relative to another person

[1]We used imgaug (https://github.com/aleju/imgaug)

- Flight: Commands directing the drone to fly in a certain direction

Note that while some users proposed other features (e.g., person holding a phone) during some of our experiments, these detailed features cannot be learned from our data set due to their rare occurrence. Thus, only color and direction features are used.

For each scenario, both success and failure cases (e.g., command with color that is not present in the current scene) were generated. Each failure case was further extended by a question and a success case containing a reference to that question (e.g., *I don't see anyone in blue. Are you talking about the person in black clothes? # yes*). To make the different cases distinguishable, each is represented as an action, e.g., FlyToColor for a simple color command or RejectRelativeDirectionWrongColor for a relative command in a scene in that no one with the specified color stood next to a person in the specified direction. A full list of actions, and in extension scenarios, is given in section A.1.1 of the appendix.

In all cases, templates were used to generate user utterances and system responses for all of these cases. Furthermore, a combination of synonym lists and varying sentence structures was employed to improve the realism of the generated dialogs. However, not all possible dialogs can be modeled due to the large variety of speech. Consequently, we rely on the dialog system to generalize from the training examples to unseen user utterances.

Additionally, to better model continuous dialog and allow the system to understand target changes, dialogs containing multiple scenarios consecutively were added to the data set.

While the essential idea sounds rather simple in theory, the actual implementation requires significant engineering efforts to ensure that only logically consistent dialogs are generated. Additionally, variable numbers of persons in each scene further complicate this process. These difficulties presented a serious challenge during the design of our data set even though only a limited number of features (color and direction), and number of persons (up to 4) were considered. For larger systems with more complicated scenarios, this approach would not be suitable, and other avenues such as gathering dialogs from real users (e.g., Amazon Mechanical Turk) would have to be explored.

5.4. Experiments

To evaluate the performance of the drone and dialog system, three experiments were conducted. The first two used a user simulator and an online user study, respectively, to evaluate the dialog system in isolation. Lastly, the performance of the dialog and drone system was tested end-to-end. In the following sections, the results of each experiment are analyzed in detail.

5.4.1. Experiment 1: User Simulator

The metrics collected during training assume that dialog steps can be evaluated in isolation, i.e., prior outputs do not influence the current dialog step. However, this is not comparable to the actual use of the system in which the current step depends on previous system outputs. Consequently, we examine system performance with a user simulator that emulates the step-by-step interaction a user would have with the dialog system.

Combination Type	Correct Action	Incorrect Action		
		Invalid	Action	Parameters
Text (Baseline)	83.43%	**0.00%**	14.04%	2.53%
Multimodal Att. (ReLu)	90.45%	0.04%	8.33%	1.17%
Multimodal Att. (Sigmoid)	91.02%	0.01%	7.77%	1.20%
Projected (ReLu)	57.13%	4.23%	25.78%	12.86%
Projected (Sigmoid)	50.25%	0.07%	30.62%	19.06%
Summation	**91.63%**	0.01%	**7.23%**	**1.13%**
Summation (w/o scene desc.)	90.89%	0.01%	7.89%	1.21%

(a) **Accuracy of Predicted Actions across Feature Combination Types**

Combination Type	∅ Steps	Success	Failure		
			Action	Target	Both
Multimodal Att. (ReLu)	1.3	95.19%	0.01%	4.67%	**0.12%**
Multimodal Att. (Sigmoid)	1.3	94.15%	0.06%	5.56%	0.23%
Projected (ReLu)	1.1	79.58%	0.17%	19.74%	0.51%
Projected (Sigmoid)	1.0	73.79%	**0.00%**	25.99%	0.22%
Summation	1.3	**96.04%**	0.01%	**3.81%**	0.15%
Summation (w/o scene desc.)	1.3	95.75%	0.01%	4.04%	0.20%

(b) **Accuracy of Predicted Target Across Different Feature Combination Types**

Combination Type	Person Count	Color (Correct)	Direction (Correct)
Text (Baseline)	47.12%	19.66%	59.89%
Multimodal Att. (ReLu)	99.61%	**92.66%**	**73.69%**
Projected (ReLu)	64.13%	39.33%	45.79%
Summation	99.67%	90.67%	71.17%
Summation (w/o scene desc.)	99.68%	91.63%	70.48%

(c) **Scene Description: Share of Correctly Predicted Parameters**

Table 5.2.: **Comparison of Feature Combination Types**

Tables 5.2a to 5.2c show an overview of the three feature combination types multi-modal attention, summation and projection with different activation functions trained on all three data sets combined. Additionally, a baseline based on training only on the dialog text and desired system responses is provided. The postfix *w/o scene description* indicates that the scene description was pruned from the system response before concatenation with the dialog history.

The simulator first picks a target in an image and then generates user utterances based on one of the scenarios (including failure cases) outlined in the preceding section. The dialog continues until a maximum number of steps is reached or a goal action is emitted, i.e., the system indicates that it can identify the target unambiguously. Finally, various metrics from the correctness of generated actions and parameters to the identification of the correct target are collected.

Note that this experiment only evaluates how the system performs under lab conditions as real users may react differently to the system's responses, or use different sentence structures and word choices not modeled by our templates. However, this experiment provides valuable insights into how differences in system architecture affect model performance.

In the following, we first briefly discuss the impact of different feature combination types on our system, including a comparison to a baseline text-only system. Following this, a comparison between models trained on one data set and evaluated on the others is presented.

Scene-Awareness and Feature Combination Types

The impact of different feature combination types on the accuracy of target identification,

predicted actions and scene descriptions is shown in Tables 5.2a, 5.2b and 5.2c, respectively. Table 5.2a additionally shows results for a model that was trained using only user utterances and desired system responses without image input.

In all tables of 5.2, projected feature combination performs significantly worse than other models and even the text baseline. We suspect that the large number of parameters in the projection feedforward layer made it difficult for the training algorithm to find suitable parameters with the available training data in the allotted 20 epochs.

For similar reasons, we hypothesize that the Summation model outperforms multi-modal attention slightly. The smaller difference between these models and projection can be explained by the fact that multi-modal attention is a generalization of summation.

Furthermore, we observe that the Summation model improves the accuracy of predicted actions (Table 5.2a) by about 50% compared to the baseline. Combined with the high target prediction accuracy of 96% of this model (Table 5.2b), this suggests that the model is able to leverage scene information to determine the correct action, i.e. it is indeed scene-aware. This is further corroborated by the significant increase of correctly predicted colors to 90% compared to 20% for the baseline shown in Table 5.2c.

One downside, as seen in the same table, is the low accuracy of 71% of direction prediction. In particular, a closer inspection of the incorrectly identified directions shows that most errors are caused by the confusion of *left* with *right* and *middle*. We suspect that the training data set inconsistently used directions (e.g., as action parameters) and thus, the model was not able to infer a generalized meaning of these words. This may be addressed by representing the different meanings of direction (e.g., absolute vs relative) by different output words.

Furthermore, examining the action confusion matrix for the Summation model, we observe that actions using relative reference (e.g., *fly to the person in green next to the person in blue*) seems to be particularly difficult to learn for the model. While the prediction accuracy for many of these actions (e.g., *RejectRelativeDirectionWrongColor (19)* or *RejectRelativeSameDirectionDirection (20)*) is high, the model has problems distinguishing other actions that have similar parameters. For example, the model often confuses *RejectWrong1ColorColor*/*RejectWrong2ColorColor* with *FlyToRelativeColorColor*. We hypothesize that it is significantly more difficult to learn these relative references because of their infrequent occurrence in the data set and the need to correlate multiple images with the input text while also relating them to each other. As learning these relations is mostly the task of the image self-attention component introduced in 3.3.1, an improved design of this part could help improve the prediction of these rarely occurring actions.

Nonetheless, these results show that only summation or its generalization multi-modal attention are well suited as a feature combination type. Furthermore, this evaluation indicates that our model is able to take advantage of the provided scene information, in addition to understanding relative descriptions between objects in the same scene.

Generalization to other data sets

In this second experiment, we examined the generalization of our model by training on one data set and evaluating on the remaining data sets. The results of these evaluations are presented in Table 5.3a and 5.3b.

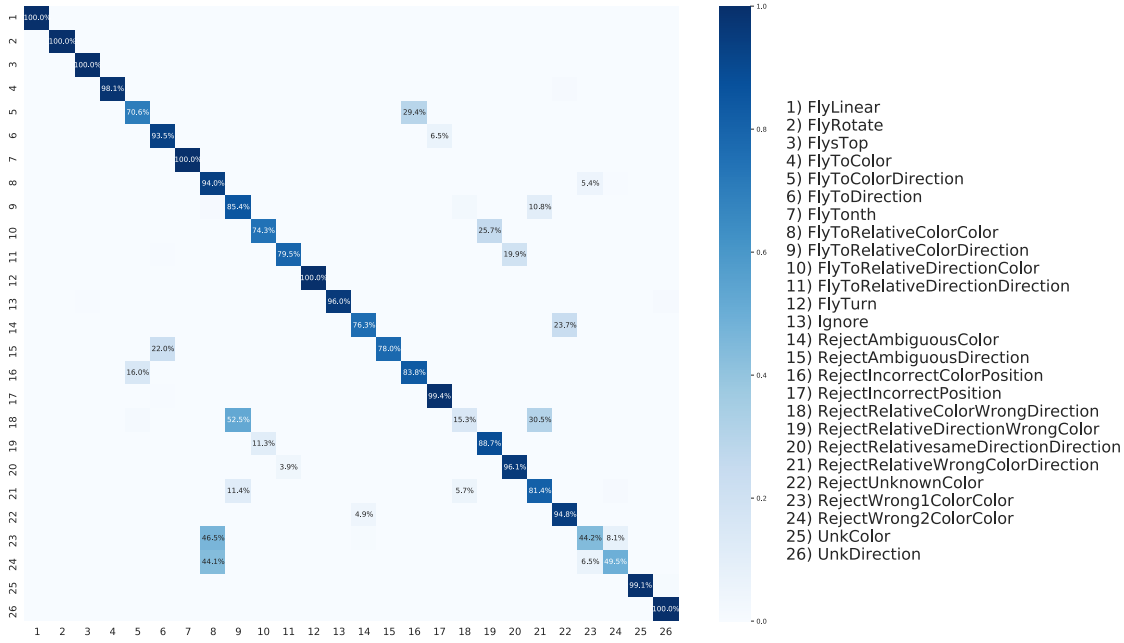


Figure 5.2.: **Action Confusion Matrix for Model Trained with „*Summation*“ Feature Combination Type**

The matrix contains row-wise the ground-truth action and column-wise the predicted action. Note that for reasons of clarity and comprehensibility, the percentage is only shown for entries with more than 3% and that the color coding has been retained for all entries. As expected based on Table 5.2a, the model predicts most actions correctly. However, actions with relative references (i.e. all actions containing any combination of *direction* and *color* in their name) are most often confused, e. g. *RejectWrong1ColorColor (23)*/*RejectWrong2ColorColor (24)* and *FlyToRelativeColorColor (8)*. These actions are complicated to learn for the model because it needs to not only relate text to one image crop but also relate multiple crops to each other. We hypothesize that a modification of the image self attention mechanism could lead to improved recognition of this type.

Training Set \ Test Set	All	CMU	KIT	SDHA
All	**90.68%**	**92.37%**	92.16%	86.76%
CMU	80.49%	91.38%	80.00%	68.06%
KIT	82.05%	78.66%	**92.96%**	71.06%
SDHA	81.62%	79.19%	78.50%	**90.34%**

(a) **Accuracy of Predicted Action Including Parameters Across Different Training and Test Sets**

Training Set \ Test Set	All	CMU	KIT	SDHA
All	**95.23%**	**97.41%**	**98.50%**	88.11%
CMU	85.44%	95.44%	91.92%	65.25%
KIT	86.20%	89.27%	96.08%	69.84%
SDHA	87.48%	81.73%	91.31%	**88.73%**

(b) **Accuracy of Predicted Target Across Different Training and Test Sets**

Table 5.3.: **Results: Generalization of Model to Other Data Sets**

Tables 5.3b and 5.3a show an overview of the target and action accuracy when when training on one of the three data sets and evaluating on the others. All models used multi-modal attention with ReLu activation and were trained for the same number of epochs.

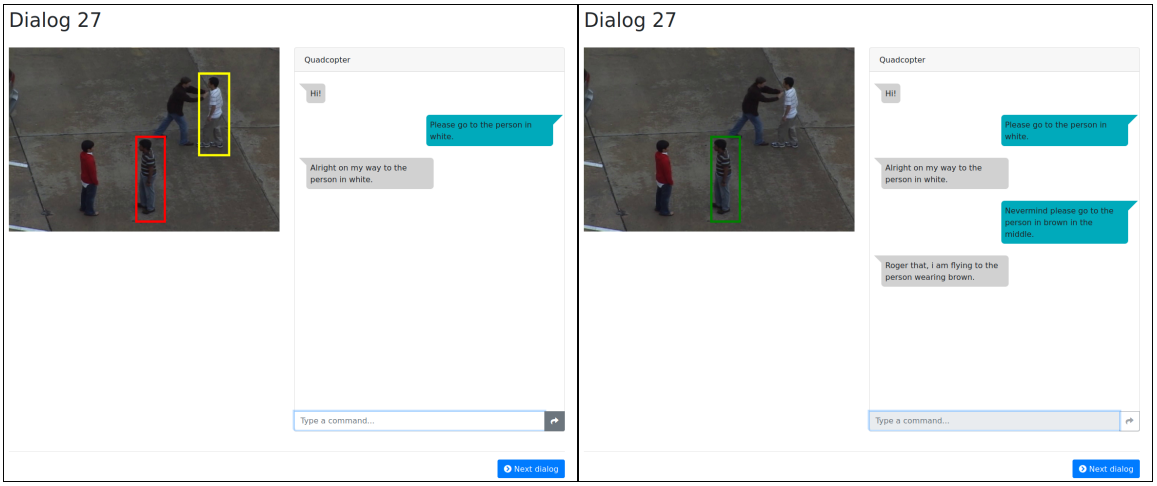


Figure 5.3.: **Interface of the User Study**

Users were instructed to try to order a „virtual“ drone to the person marked with a red rectangle. They could then enter commands in a chat-like interface to command the drone to the right target, including possibly correcting themselves after issuing a wrong command.

In Table 5.3b, we can observe that for the most part, training on all data sets combined leads to the best performance. This model even slightly surpasses the recognition rate of models only trained and evaluated on KIT (96.08% vs. 98.50%) and CMU (95.40% vs. 97.41%). We suspect that two factors mainly contribute to this improvement. One, the combined data set contains more dialogs, potentially leading to a more optimized classification head. Nonetheless, note that the model saw each image and dialog combination the same number of times (once per epoch) in all data sets. This directly leads to the second factor. As both data sets share a certain subset of colors, the model may learn a better association between a color word in the input command and different shades of that color in the input image.

A similar observation can be made for the action and parameter prediction shown in Table 5.3a. The model trained on the combined data set achieves results very close to the performance of training and evaluating on the same data set. An exception is the SDHA data set, which contains far more very similar colors (e.g., gray, dark blue and black). Thus, a model trained only on this data set may be more sensitive to distinguishing these colors and as a result achieve a higher accuracy on the metric reported in Table 5.3a.

From this discussion, it becomes evident that a factor that significantly limits the generalization of our models is that our data set is rather small and does not contain a large variety of colors. However, as observed in Table 5.3b, this variety may help improve the model's understanding of user commands. Consequently, this should be addressed in future research.

5.4.2. Experiment 2: Online User Study

To further evaluate the system under more realistic conditions, we conducted an online user study. During this study, users were instructed to imagine they are controlling a drone and are trying to send it with a verbal command to the person marked with a red

rectangle. This was visualized using a chat-like interfaces as shown in Fig 5.3. To control the „virtual“ drone, users entered text commands in the text input field at the bottom before receiving a response from the drone. They were instructed to continue chatting with the drone until the right target was identified or they gave up. Every seven dialogs, a short three-question survey was conducted to judge the user experience. Furthermore, note that we incentivized users to complete more dialogs by providing a gift card give-away with increased chance of winning proportional to the number of completed dialogs.

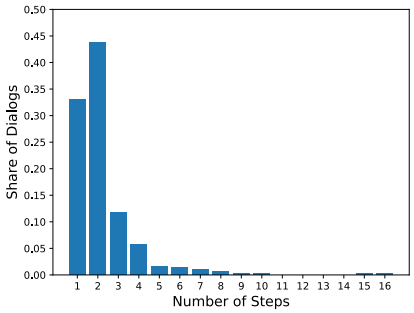


Figure 5.4.: **Histogram of Dialog Length**

∅ Steps	Success	Failure	
		Target	Action
2.2	86.8%	3.1%	10.1%

Table 5.4.: **User Study: Average Dialog Length and Target Accuracy at the End of Dialog**

In total, during the week-long study, 19 users with mostly technical background contributed 424 dialogs. A histogram of the average length of these dialogs is shown in Fig. 5.4. Additionally, Table 5.4 provides an overview of the success and failure rates and the average length of dialogs.

A first observation concerns the average length of dialogs. In contrast to the user simulator, which only needed 1.3 steps on average, users required 2.2. This is further confirmed by the histogram shown in 5.4 which shows that about 45% of dialogs required two steps. Closer inspection of the dialogs reveals that the system often responded by questioning the first command although it was unambiguous. In the second step, the system then correctly recognized the target.

We suspect that this is in part caused by users choice of words and sentence structure which deviated from the training data quite significantly in some cases. This issue could be addressed by collecting more data from real users as opposed to using hand-designed templates in addition to optimizing for dialog length.

Additionally, one can observe in Table 5.4 that most failures were not caused by the system identifying the wrong target, but by misrecognitions of actions, e.g. the system asking a question instead of identifying a target or confusing a goal action with one that instructs the drone to fly to a specific position. Noticeably, this happens mostly when the dialog becomes longer than the dialogs used in training, and users continue to refer to the start of the conversation. Nonetheless, the system achieves a high success rate of 86.8%.

Furthermore, users were asked to complete the survey shown in Table 5.5. In summary, users reported average naturalness of dialogs and difficulty of selecting the right target. These results imply that further work on improving the dialog (e.g., not questioning the first command if it is unambiguous) needs to be done. In addition, users did report that they would find voice control for drones only moderately useful. An additional survey has

Question	Average Response
On a scale of 1-10, how natural did the dialogs feel to you?	4.9
On a scale of 1-10, how difficult was it to send the quadcopter to the correct target?	5.5
On a scale of 1-10, how useful would you find controlling a quadcopter with a mature dialog system (if available, e.g., for quadcopter filming, bridge inspections, ...) instead of manual flight control?	6.0

Table 5.5.: **Survey**

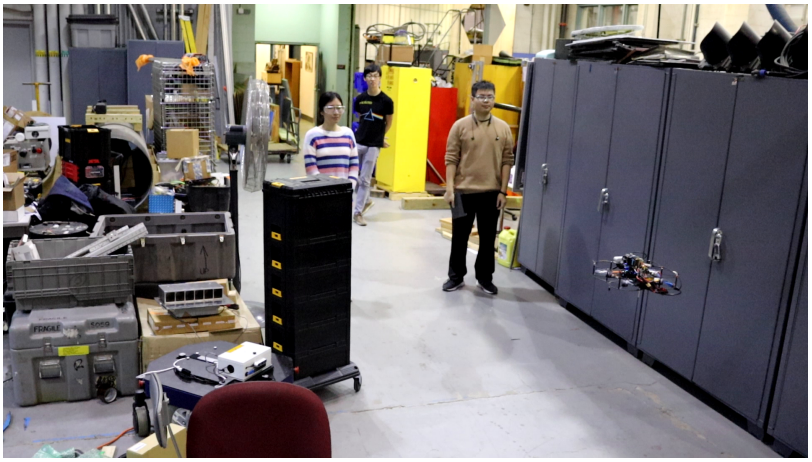


Figure 5.5.: **Testing Area with two test persons**

The location all experiments were conducted in. Users were instructed to wait until the operator selected a person and then move back independently. To increase the safety of this experiment, test subjects were asked to walk backwards, and if the drone comes to close, move out of its way. However, the latter issue only happened once during the experiments.

to be conducted to evaluate if this perception changes when users actually have to use a drone or this control mode is applied to other, more practical uses case like SAR.

5.4.3. Experiment 3: System Evaluation

Lastly, we evaluated the performance of the entire system end-to-end. Table 5.6 shows the results of 10 experiments that were conducted with 3 different persons in the location shonw in Fig. 5.5. All users received a brief introduction explaining which features (color, direction, flight commands) the system understands and were not provided any additional guidance on which sentences to use. However, note that some of our subjects may have heard about the training data prior to the experiment. To ensure the safety of all participants, the drone was limited to flying at 0.5m/s or slower.

As Table 5.6 shows, the dialog system worked as intended for the majority of cases. However, speech recognition did not work very reliably. Independet tests without the drone flying confirmed that these failures were mostly caused by the noise of the drone. When operators moved further away from the drone, speech recognition performance improved significantly. This system could be improved by using a directional microphone instead of the omnidirectional microphone built into the tablet used for the experiments.

Experiment No.	Speaker	Speech	Dialog	Global Planner / Obstacle Avoidance	Hardware
1	1	Didn't recognize speech on first try due to noise and accent	✓	✓	✓
2	1	Didn't recognize speech on first try due to noise and accent	Dialog selected wrong person	✓	✓
3	1	Didn't recognize speech on first try due to noise and accent	✓	✓	✓
4	2	✓	✓	Drone got stuck due to late recognition of obstacle	✓
5	2	✓	✓	✓	✓
6	2	✓	✓	✓	✓
7	3	Recognized wrong word due to noise	✓	✓	✓
8	3	✓	✓	✓	✓
9	3	✓	✓	✓	✓
10	3	✓	✓	Drone got stuck and too close to person	✓
Summary		6/10	9/10	8/10	10/10

Table 5.6.: **End-to-End System Test Results**

Alternatively, additional noise filtering could also lead to significantly improved performance.

Ignoring this issue, the system was able to determine and follow the targeted person in 7 of 10 cases. In particular, Experiment 8 nicely demonstrated
how the user learned to use the system and was able to correct the drone after a misdetection of a person walking around in the background sent the drone into the wrong direction. Additionally, note that the filtering added by us to the mapping component did only partially solve the appearance of spurious points in the pointcloud. From data gathered during the experiments, we hypothesize that the depth camera miscalculated the depth for very bright spots (e.g., lights) and incorrectly believed that an obstacle was right above it. This caused the drone to either get stuck or turn seemingly randomly as can be observed in the video accompanying this thesis.

To summarize, users were able to interactively control and correct the drone and quickly learned to use the system. While speech recognition could be improved, the drone was able to successfully track and follow the targeted person.

6. Conclusion & Future Work

In this thesis, we addressed the problem of integrating scene-awareness into an interactive dialog system. We also investigated how this system can be used on a UAV for the person following task.

To enable dialog systems to take advantage of scene context, we presented a novel approach to integrating visual information into a transformer-based dialog system and evaluated different feature combination methods based on multi-modal attention and projection. Additionally, we implemented a question selection procedure which allows the dialog system to ask goal-oriented questions grounded in the system's scene understanding to clarify a user's task description. This system was then deployed and evaluated on an autonomous UAV.

The experimental evaluation showed promising results and suggested that this model indeed learns an understanding of the scene. Furthermore, our experiments on several data sets confirmed that the model's design based on object image crops can generalize to unseen scenes with similar camera perspectives. Furthermore, a user study and experiments with a UAV indicated that humans can use this system with little to no formal training with a high success rate of 86.8% and 70%, respectively.

These results combined suggest that the presented system's transferability to new uses cases and ease of use make it applicable to many practical use cases beyond the person following task. In particular, this applies to tasks, whose high-level mission descriptions can be accurately expressed with speech, like interactive package drop-off negotiation or drones as fully-integrated team members in SAR. Consequently, the barrier to deploying drones in these applications can be significantly lowered with the presented system and make the use of drones more efficient.

However, speech alone may not be the best modality for all use cases due to the limited precision of verbal instructions and external factors like noise making speech recognition difficult. This could be addressed in future research on flexi-modal human-robot interaction, which aims to combine multiple communication modes. This area of research also presents new challenges for conflict resolution and disambiguation across modalities.

Furthermore, making the dialog system, including the question selection component, end-to-end trainable, could be investigated further. This integration would have the advantage of allowing the optimization of additional criteria such as dialog length and more natural, more goal-oriented dialogs by tighter integration of system components.

During work on this thesis, it also became evident that only few data sets suitable for visual dialog exist today. Additionally, most existing data sets only contain limited dialog, such as the question-answer format used for DSTC7 [65]. While these data sets provide ways of evaluating audiovisual dialog systems, they do not model true interactivity, including the system asking questions as opposed to the human operator. This perspective, however, is crucial for natural human-robot interactions. Thus, to make works on similar

dialog systems comparable and support research on multi-modal dialogs, the creation of a suitable data set with multiple modalities and truly interactive dialog should be explored.

Following a similar train of thought, another more general research direction related to making the machine learning process more human-like arises from this work. Humans, in contrast to ML models, are capable of acquiring new skills with comparably few examples. This is in part based on imagination allowing humans to create new, fictional scenarios based on prior experiences. Transferring this ability to machine learning models presents a very intriguing research area with great potential for improved system performance and a huge overall impact on the field of machine learning.

Bibliography

[1] Stanislaw Antol et al. “Vqa: Visual question answering”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2425–2433.

[2] Michael Arens. *Einführung in die Bildfolgenauswertung - Objektverfolgung I*. May 2018. URL: https://ies.anthropomatik.kit.edu/lehre_bildfolgenauswertung.php (visited on 01/26/2020).

[3] Anand Asokan, Allan Joseph Pothen, and Raj Krishnan Vijayaraj. “ARMatron—A wearable gesture recognition glove: For control of robotic devices in disaster management and human rehabilitation”. In: *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*. IEEE. 2016, pp. 1–5.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[5] Antoine Bordes, Y-Lan Boureau, and Jason Weston. “Learning end-to-end goal-oriented dialog”. In: *arXiv preprint arXiv:1605.07683* (2016).

[6] Jonathan Cacace et al. “A control architecture for multiple drones operated via multimodal interaction in search & rescue mission”. In: *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2016, pp. 233–239.

[7] Hongshen Chen et al. “A survey on dialogue systems: Recent advances and new frontiers”. In: *Acm Sigkdd Explorations Newsletter* 19.2 (2017), pp. 25–35.

[8] Stefan Constantin, Jan Niehues, and Alex Waibel. “An end-to-end goal-oriented dialog system with a generative natural language response generation”. In: *9th International Workshop on Spoken Dialogue System Technology*. Springer. 2019, pp. 209–219.

[9] Stefan Constantin, Jan Niehues, and Alex Waibel. “Multi-task learning to improve natural language understanding”. In: *CoRR* abs/1812.06876 (2018). arXiv: 1812.06876. URL: http://arxiv.org/abs/1812.06876.

[10] Abhishek Das et al. “Visual dialog”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 326–335.

[11] Robin Deits et al. “Clarifying commands with information-theoretic human-robot dialog”. In: *Journal of Human-Robot Interaction* 2.2 (2013), pp. 58–79.

[12] Li Deng et al. “Use of kernel deep convex networks and end-to-end learning for spoken language understanding”. In: *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2012, pp. 210–215.

[13] Bhuwan Dhingra et al. “Towards end-to-end reinforcement learning of dialogue agents for information access”. In: *arXiv preprint arXiv:1609.00777* (2016).

[14] Gregory Dudek, Junaed Sattar, and Anqi Xu. “A visual language for robot control and programming: A human-interface study”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 2507–2513.

[15] Mihail Eric and Christopher D Manning. “A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue”. In: *arXiv preprint arXiv:1701.04024* (2017).

[16] Mihail Eric and Christopher D Manning. “Key-value retrieval networks for task-oriented dialogue”. In: *arXiv preprint arXiv:1705.05414* (2017).

[17] Ramon A Suarez Fernandez et al. “Natural user interfaces for human-drone multi-modal interaction”. In: *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2016, pp. 1013–1022.

[18] Tino Fuhrman et al. “An Interactive Indoor Drone Assistant”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2019.

[19] Akira Fukui et al. “Multimodal compact bilinear pooling for visual question answering and visual grounding”. In: *arXiv preprint arXiv:1606.01847* (2016).

[20] Scott A Green et al. “Collaborating with a mobile robot: An augmented reality multimodal Interface”. In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 15595–15600.

[21] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[22] Charles T Hemphill, John J Godfrey, and George R Doddington. “The ATIS spoken language systems pilot corpus”. In: *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*. 1990.

[23] João F Henriques et al. “High-speed tracking with kernelized correlation filters”. In: *IEEE transactions on pattern analysis and machine intelligence* 37.3 (2014), pp. 583–596.

[24] Alexander Hermans, Lucas Beyer, and Bastian Leibe. *In Defense of the Triplet Loss for Person Re-Identification*. 2017. arXiv: 1703.07737 [cs.CV].

[25] Chiori Hori et al. “End-to-end audio visual scene-aware dialog using multimodal attention-based video features”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 2352–2356.

[26] Chiori Hori et al. “Attention-based multimodal fusion for video description”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 4193–4202.

[27] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV].

[28] Pei Jia et al. “Head gesture recognition for hands-free control of an intelligent wheelchair”. In: *Industrial Robot: An International Journal* 34.1 (2007), pp. 60–68.

[29] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[30] Gakuto Kurata et al. "Leveraging sentence-level information with encoder lstm for semantic slot filling". In: *arXiv preprint arXiv:1601.01530* (2016).

[31] Hung Le et al. "Multimodal transformer networks for end-to-end video-grounded dialogue systems". In: *arXiv preprint arXiv:1907.01166* (2019).

[32] Oliver Lemon et al. "The WITAS multi-modal dialogue system I". In: *Seventh European Conference on Speech Communication and Technology*. 2001.

[33] Lori Levin et al. "The Janus-III translation system: speech-to-speech translation in multiple domains". In: *Machine translation* 15.1-2 (2000), pp. 3–25.

[34] Linjie Li et al. "Relation-aware Graph Attention Network for Visual Question Answering". In: *arXiv preprint arXiv:1903.12314* (2019).

[35] Xiujun Li et al. "End-to-end task-completion neural dialogue systems". In: *arXiv preprint arXiv:1703.01008* (2017).

[36] Fei Liu and Julien Perez. "Gated end-to-end memory networks". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. 2017, pp. 1–10.

[37] Jiasen Lu et al. "Hierarchical question-image co-attention for visual question answering". In: *Advances In Neural Information Processing Systems*. 2016, pp. 289–297.

[38] Brad Myers et al. "Flexi-modal and multi-machine user interfaces". In: *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*. IEEE. 2002, pp. 343–348.

[39] Jawad Nagi et al. "Human control of UAVs using face pose estimates and hand gestures". In: *2014 9th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE. 2014, pp. 1–2.

[40] Pedro Neto et al. "Gesture-based human-robot interaction for human assistance in manufacturing". In: *The International Journal of Advanced Manufacturing Technology* 101.1-4 (2019), pp. 119–135.

[41] Thai-Son Nguyen et al. "Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation". In: *arXiv preprint arXiv:1910.13296* (2019).

[42] Ryuichi Nisimura et al. "ASKA: receptionist robot with speech dialogue system". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 2. IEEE. 2002, pp. 1314–1319.

[43] Cesc Chunseong Park, Byeongchang Kim, and Gunhee Kim. *Attend to You: Personalized Image Captioning with Context Sequence Memory Networks*. 2017. arXiv: 1704.06485 [cs.CV].

[44] Mengye Ren, Ryan Kiros, and Richard Zemel. "Exploring models and data for image question answering". In: *Advances in neural information processing systems*. 2015, pp. 2953–2961.

[45] Alan Ritter, Colin Cherry, and William B Dolan. "Data-driven response generation in social media". In: *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics. 2011, pp. 583–593.

[46] M. S. Ryoo and J. K. Aggarwal. *UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA)*. http://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html. 2010.

[47] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015). DOI: 10.1109/cvpr.2015.7298682. URL: http://dx.doi.org/10.1109/CVPR.2015.7298682.

[48] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units". In: *arXiv preprint arXiv:1508.07909* (2015).

[49] Tao Shen et al. "Disan: Directional self-attention network for rnn/cnn-free language understanding". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[50] Dimitris Spiliotopoulos, Ion Androutsopoulos, and Constantine D Spyropoulos. "Human-robot interaction based on spoken natural language dialogue". In: *Proceedings of the European workshop on service and humanoid robots*. 2001, pp. 25–27.

[51] Rainer Stiefelhagen et al. "Enabling multimodal human–robot interaction for the karlsruhe humanoid robot". In: *IEEE Transactions on Robotics* 23.5 (2007), pp. 840–851.

[52] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. "End-to-end memory networks". In: *Advances in neural information processing systems*. 2015, pp. 2440–2448.

[53] I Sutskever, O Vinyals, and QV Le. "Sequence to sequence learning with neural networks". In: *Advances in NIPS* (2014).

[54] I Sutskever, O Vinyals, and QV Le. "Sequence to sequence learning with neural networks". In: *Advances in NIPS* (2014).

[55] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[56] Joshua B Tenenbaum and William T Freeman. "Separating style and content with bilinear models". In: *Neural computation* 12.6 (2000), pp. 1247–1283.

[57] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[58] Vaibhav Viswanathan. "Fast motion-planning for high-speed drone flight". unpublished. 2019.

[59] Stefan Waldherr, Roseli Romero, and Sebastian Thrun. "A gesture based interface for human-robot interaction". In: *Autonomous Robots* 9.2 (2000), pp. 151–173.

[60] Tsung-Hsien Wen et al. "A network-based end-to-end trainable task-oriented dialogue system". In: *arXiv preprint arXiv:1604.04562* (2016).

[61] Jason E Weston. “Dialog-based language learning”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 829–837.

[62] Jason Weston, Sumit Chopra, and Antoine Bordes. “Memory networks”. In: *arXiv preprint arXiv:1410.3916* (2014).

[63] Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. “Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning”. In: *arXiv preprint arXiv:1702.03274* (2017).

[64] Jason D Williams and Geoffrey Zweig. “End-to-end lstm-based dialog control optimized with supervised and reinforcement learning”. In: *arXiv preprint arXiv:1606.01269* (2016).

[65] Koichiro Yoshino et al. “Dialog System Technology Challenge 7”. In: *arXiv preprint arXiv:1901.03461* (2019).

[66] Licheng Yu et al. “Multi-target embodied question answering”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6309–6318.

[67] Dongxiang Zhang, Rui Cao, and Sai Wu. “Information fusion in visual question answering: A Survey”. In: *Information Fusion* 52 (2019), pp. 268–280.

[68] Tiancheng Zhao and Maxine Eskenazi. “Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning”. In: *arXiv preprint arXiv:1606.02560* (2016).

A. Appendix

A.1. Dialog System

A.1.1. System Outputs and Responses

Action	Parameters	Example
FlyLinear	distance, flight_direction	FlyLinear MUCH forward
FlyRotate	distance, turn_direction	FlyRotate NORMAL TURN_left
FlyStop		FlyStop
FlyToColor	color	FlyToColor COLOR_black
FlyToColorDirection	color, image_direction	FlyToColorDirection COLOR_black DIRECTION_LEFT
FlyToDirection	image_direction	FlyToDirection DIRECTION_MIDDLE
FlyToNth	nth_person, image_direction	FlyToNth 1 DIRECTION_LEFT
FlyToRelativeColorColor	color, image_direction, color, image_direction	FlyToRelativeColorColor COLOR_WHITE DIRECTION_LEFT / COLOR_BLACK DIRECTION_RIGHT
FlyToRelativeColorDirection	color, image_direction, color, image_direction	FlyToRelativeColorDirection COLOR_WHITE DIRECTION_LEFT / COLOR_BLACK DIRECTION_RIGHT
FlyToRelativeDirectionColor	color, image_direction, color, image_direction	FlyToRelativeDirectionColor COLOR_WHITE DIRECTION_LEFT / COLOR_BLACK DIRECTION_RIGHT
FlyToRelativeDirectionDirection	image_direction, image_direction	FlyToRelativeDirectionDirection DIRECTION_RIGHT DIRECTION_RIGHT
FlyTurn	turn_direction	FlyTurn AROUND
RejectAmbiguousColor	color_word	RejectAmbiguousColor brown
RejectAmbiguousDirection	image_direction	RejectAmbiguousDirection DIRECTION_MIDDLE
RejectIncorrectColorPosition	color_word, image_direction	RejectIncorrectColorPosition green DIRECTION_RIGHT
RejectIncorrectPosition	image_direction	RejectIncorrectPosition DIRECTION_LEFT
RejectRelativeColorWrongDirection	color_word, image_direction	RejectRelativeColorWrongDirection white DIRECTION_LEFT
RejectRelativeDirectionWrongColor	image_direction, color_word	RejectRelativeDirectionWrongColor DIRECTION_LEFT blue
RejectRelativeSameDirectionDirection	image_direction, image_direction	RejectRelativeSameDirectionDirection DIRECTION_RIGHT DIRECTION_RIGHT
RejectRelativeWrongColorDirection	color_word, image_direction	RejectRelativeWrongColorDirection yellow DIRECTION_LEFT
RejectUnknownColor	color_word	RejectUnknownColor beige
RejectWrong1ColorColor	color_word, color	RejectWrong1ColorColor red COLOR_BLACK
RejectWrong2ColorColor	color, color_word	RejectWrong2ColorColor COLOR_WHITE red
UnkColor		UnkColor
UnkDirection		UnkDirection

Table A.1.: **List of Actions**

Question	Parameters	Example
QuestionColor	color	QuestionColor COLOR_black
QuestionColorDirection	color, image_direction	QuestionColorDirection COLOR_white DIRECTION_LEFT
QuestionDirection	image_direction	QuestionDirection DIRECTION_MIDDLE
QuestionDirectionColor	image_direction, color	QuestionDirectionColor DIRECTION_RIGHT COLOR_black
QuestionMultipleColor	color, color	QuestionMultipleColor COLOR_white COLOR_black
QuestionMultipleDirection	image_direction, image_direction	QuestionMultipleDirection DIRECTION_LEFT DIRECTION_RIGHT
QuestionNoClue		QuestionNoClue
QuestionNoIdea	color_word, image_direction	QuestionNoIdea green DIRECTION_RIGHT
QuestionRelativeColorWrongColor	color, color	QuestionRelativeColorWrongColor COLOR_WHITE COLOR_black
QuestionRelativeColorWrongDirection	color_word, image_direction	QuestionRelativeColorWrongDirection white DIRECTION_RIGHT
QuestionRelativeDirectionColor	image_direction, color	QuestionRelativeDirectionColor DIRECTION_LEFT COLOR_black
QuestionRelativeDirectionDirection	image_direction, image_direction	QuestionRelativeDirectionDirection DIRECTION_RIGHT DIRECTION_LEFT
QuestionRelativeWrongColorDirection	color, image_direction	QuestionRelativeWrongColorDirection COLOR_black DIRECTION_LEFT

Table A.2.: **List of Questions**

A.1.2. Image Feature Extractor

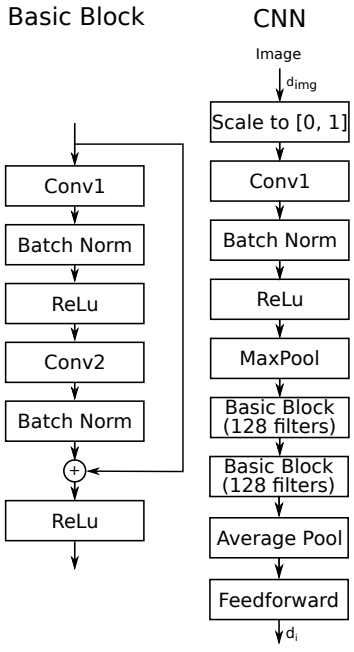


Figure A.1.: **CNN Used for Feature Extraction from Person Images**

The CNN is based on ResNet [21] and uses two basic blocks with 128 filters each. Initially, Conv1 initially applies a $7x7$ kernel with 128 filters to the RGB input image. Further details can be found in torchvision's ResNet implementation: https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py

A.2. Drone System

A.2.1. Hardware

- Drone (1.18kg w/ battery)
  - Futaba T8J Remote
  - Telemetry Radio
  - Student Competition 5" Bundle from GetFPV
    * 1x Black Polyamide Standoff Set (6mm)
    * 4x Black Hex Standoffs 45mm (4 pcs)
    * 1x M3x5 Button Head Steel Screw Set (50pcs)
    * 1x Lumenier Medium Lipo Strap (3pcs)
    * 4x Lumenier LX2205-12 2400KV Motor
    * 1x Holybro PX4 2.4.6 "Pixhawk" & M8N GPS & PM & 100mW Radio Telemetry V3 (915Mhz) OR Pixracer
    * Dampeners for Pixracer
    * 1x Lumenier BLHeli_32 32bit 35A 4-in-1 ESC 2-4s w/ BEC 3A/12v, 1A/5v, DSHOT 1200
    * 1x FrSky R-XSR 2.4GHz 16CH ACCST Micro Receiver w/ S-Bus & CPPM
    * 1x Lumenier 5x5x3 - Butter Cutter Propeller (Set of 4 - Clear)
    * 1x Holybro PX4FLOW Kit v1.31
- ConnectTech Elroy Carrier Board
  - Elroy Power Cable
  - Elroy HDMI Cable
- Nvidia TX2
  - Custom aluminium mounting plate for TX2
  - USB A - Mini A (TX2 => USB Hub)
  - USB A - Micro B (TX2 => Pixracer)
  - Anker USB hub / Pi Zero USB Hub
  - 2 Ports USB 3.0 Female to USB3.0 Motherboard Female 20 Pin Header
  - 90 Degree Angled USB 20-Pin Internal Header Mini Connector - Down Angle
- D435
  - USB A - USB C (D435)
- T265

  - USB A - USB Micro B
  - USB 3.0 Y cable (mit extra power cable)
  - USB 2-6S LiPo Charger
  - 45°angled mounting plate for T265
  - Dampeners
- Battery Cables
- XT60 Connectors