++ R-ZWEI KICKERS

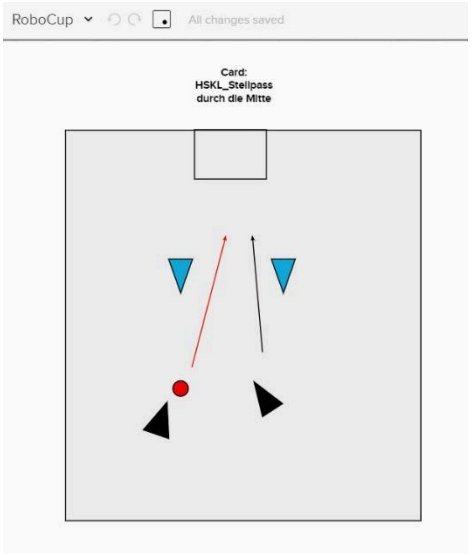
Hochschule Kaiserslautern
University of Applied Sciences

Team Report 2021

Connor Lismore, David Kostka, Thomas Jäger, Andreas Hobelsberger, Felix Mayer, Adrian Müller

Hochschule Kaiserslautern, FB Informatik / MST, Amerika Str. 1, 66482 Zweibrücken, Germany

Revision: October 2021



Content

Introduction **2**

Robot behavior: trigger points – better than rules **4**

Teach-in complex action sequences by demonstration **5**

Recording of a Teach-In Card 5

Gamepad key event handling 6

Q-Learning optimal gameplay tactics **7**

Changing the dispatcher into a Q-Agent 7

Evaluation criteria 8

Training runs in the simulator 9

Improvements in object recognition and image classification **10**
Data Augmentation: 10
Results 13
Challenges in Data Augmentation 14
Special: Our approaches to the SPL local challenges of 2021 **15**
Obstacle Avoidance 15
Path planning 15
A New Path Planner 15
Drawbacks 19
Dribbling 19
Goal Kick 20
Future Expansions 20
Passing Challenge 20
Changes to the Q-Learning approach 20
What went wrong: traps and pitfalls **21**
Read the R2K wiki! 21
Recent and future work, knowledge transfer to the IT industry **22**
Data Version Control and Evaluation: 23
Future Challenges in Dataset: 24
Current Challenges in Data Augmentation 24
Acknowledgements 25
Bibliography **25**

1. Introduction

The R-ZWEI KICKERS team is part of the Smart Machines working group at UAS Kaiserslautern, located in Zweibrücken, Rhineland-Palatinate. We take an agile approach, combining incremental and iterative development with explorative prototyping, applying methods of hybrid AI, and regularly integrating project work and theses of students.

This report is intended for a wide audience, ranging from undergraduate students to technical, industrial managers. It will show you the challenges, problems and solutions which we experienced in our first year in the RoboCup SPL.

The R-ZWEI KICKERS were established in 2020 and currently consist of one professor and about 10 undergraduate and graduate students in the Department of Computer Science, as well as external volunteers. Currently, our code base is a modified version of the B-Human Code Drop 2019. We have modified the robot behaviour control, and the SimRobot environment to allow for automatic test procedures. Completely custom features include the Teach-In of robot behaviour, a data mining analysis, coupled to a reinforcement learning (Q-learning) module. Additionally, we have developed our own path planning algorithm, a unified image recognition and object detection model.

So far, we have (remotely) participated in

- the vRoHOW Workshop 2020 (http://smart-machines.hs-kl.de/event/4821/),
- the 2021 workshop on image annotation requirements (http://smart-machines.hskl.de/event/workshop-image-annotation-requirements/)
- the Open Robotic Workshop RoDEO 2021 , where we successfully managed our Naos on the remote premises, collected log data, and conducted two local challenges on our own premises.
- the German RoboCup replacement in May 2021 GORE 2021 – RoboCup Germany in several 5vs5 remote games
- the RoboCup 2021 – RoboCup Standard Platform League, in two local challenges (cf. section 5)

See more details about the R-ZWEI KICKERS team are listed in the Active SPL Teams Worldwide – RoboCup Standard Platform League. You may visit our YouTube Channel https://www.youtube.com/channel/UCtcAPnzGHqizuypEyWxEW_Q for some videos with live examples.

tbw: *Content of this report*

2. Robot behavior: trigger points – better than rules

B-Humans rule-based card concept and their linear, top-down dispatcher (aka the PriorityListDealer()) were found hard to develop and maintain. The mutual dependencies of the pre-and post-conditions in the head of their behaviour cards, and the need to specify both - the order of application of cards and the fine tuning of suitable thresholds in their preconditions, mutually consistent to all other cards - complicated the development of a robust gameplay. In a nutshell, the initial situation reflects the lack of scalability and the brittleness of classic, symbolic AI.

Our approach to overcome these inherent difficulties is as follows:

1. We introduced a so-called "word model", which is applicable to the current game situation (its *features* are the maximum of available information like position of robots, ball, game time, score, …)

2. During simulation: The human "reach-in" instructor intervenes in a specific game situation, teaching a dedicated behaviour ("playback sequence" for one manually selected robot (see next section for details on the teach-in process). Our program notes the corresponding pair < word model, sequence of action>

   We call them teach-in cards, written TI-cards.

3. Later, offline: the set of world model data is subject to a data mining analysis, which identifies the list of a minimum subset of features with a maximum dissimilarity, the so called "trigger points".

4. Application of TI-cards (game play): we modified the dispatchers search strategy to breadth search. During gameplay, it searches for the best fitting trigger points, and notes the corresponding playback sequences. Each qualified teach-in card will potentially be executed; we use a weighted random selection method.

5. Hybrid execution (during game play): we allow the designer of the game strategy to mix rule-based and teach-in cards into a single gameplay stack. A typical design looks like

   I. Very specific rule-based cards (eg. "offense is free in front of the goal")

   II. Teach-In card, best card selected dynamically for typical game situations (cf. section 3 for TICollection card)

   III. Catch-all rule-based, default cards (eg. "return to your default position")

< show pseudo code of dispatcher here>

This results in an intelligent behaviour of our robots, as the modified dispatcher now combines the specificity of symbolic AI with the robustness of data mining approaches.

However: disadvantages no concept of time

- Architecture: word model as singleton

3. Teach-in complex action sequences by demonstration

Notizen aus der Redaktion vom 21.9.

- erfassung, timing, code architektur,

- *Concept: human instructor sets up a certain game situation (or watches a game and intercepts at a certain*
- *While watching: non trigger points*
- *On key press:*
    - *store trigger point, and*
    - *start recording: store sequence of actions (ie Skills(),*
- *on key press: stop and save (persist to file) for later data mining analysis and playback*

- PS-4 controller Binding of Keys to BHuman Skills is changed wrt to B-Human (static singleton -> cf. section 8) X

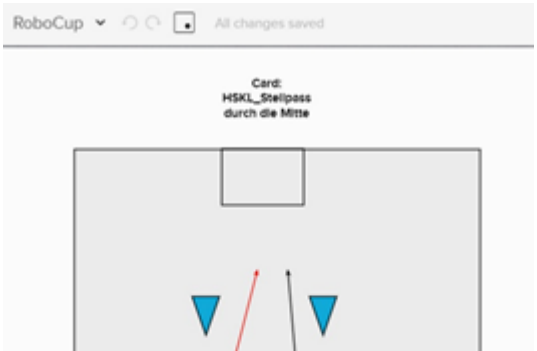
- activ Gamepad control 'over-rules' any other card (recording)

- timestamp - based TI replay sequence, causing problems (c.f. 7)

Adrian, Felix

Recording of a Teach-In Card

As mentioned above (cf. section 2), we found it difficult to maintain several rule-based behaviour cards, especially as each new card requires programming in C++. We developed a dedicated card for recording keypresses (a human instructor pressing action keys on a PS-4 controller). It maps the key presses to skills and actions (partly our own code, partly B-Human code) of the robots, and finally writes the sequence of keypresses to a CSV file.



At the end of teach-in, we have dozens of short-timed, hand designed tactical manoeuvres of a single, or even multiple robots in coordination. We call these cards "Teach-In cards" (TI-Cards), as

opposed to “regular” cards, as provided by the B-Human code base. Note that our variant of “card” does not require reprogramming or re-compiling for development and maintenance.

After finishing the teaching and re-starting the thread for gameplay, all so called “TI-Cards” are loaded at once into the Naos memories. Thus, our search algorithm can loop over all cards of several human “trainers”, and apply the qualification criteria.

Like any other regular card, teach-in cards have entry conditions that must be met for them to qualify. However, since these are not explicitly programmed, a "world model" was introduced for this purpose, as already shown in section 2. This results in the peculiarity that without new compilation of the code, a constant number of regular and a non-constant number of Teach-In cards can be in the system. However, the dispatcher selects from all available explicitly programmed Cards. The selection set of cards is defined by the "gameplayCard.cfg" configuration file of B-Human. This raises the question of how an arbitrarily large set of non-programmed teach-in cards can be selected by the dispatcher. For this purpose, we added a unique, self-programmed card, the "TICollection" card. As the name suggests, this is a container of all available Teach-In cards.

Gamepad key event handling

Key handling is required to control the NAO in the simulator to record a teach-in sequence. B-Human has a built-in key handling for gamepad controllers. However, these are based on a low-level representation and the movements that take place in them are set by motion requests. Examples:

- theMotionRequest.motion = MotionRequest::walk;
- theMotionRequest.motion = MotionRequest::getUp;

However, complex actions like "go to the ball" cannot be represented this way. This complicates the control immensely, since steering a simulated NAO viewed from a bird's-eye view to a lying ball and aligning it in front of the ball, sometimes with millimetre precision, for an efficient kick is very imprecise as a result.

For this reason, we have completely reworked the key handling. We created a static memory class in the heap, the "KeyLogger". This stores all key events recognized by B-Human and can deliver them to all modules from all threads. This way we can do key handling in the Behaviour Thread, which allows us to provide the key event information to cards and skills. With this we can attach complex robot skills to single key presses. Thus, the human instructor can raise complex action sequences automatically at the push of a button. This increases the usability of the gamepad control immensely and allows for more complex actions when creating teach-in cards despite their simplicity.

4. Q-Learning optimal gameplay tactics

- Motivation: value based action to improve rule based system (short), improves random selector (see. sec 2). No more manual rule hierarchy!

- Used definition: q-Function; Evaluation criteria X

-- Training runs inside the Sim; how do they work

We refined the modified dispatcher (see section 2) by providing an “oracle” for the breadth-first approach. It is a “state x action” based oracle, which decides what card (action) is to be drawn for a given state of the world (e.g. positions of the robots on the field). Our search identifies a list of qualifying cards (see section 2 for the qualification criteria), draws one of the top qualifying cards from this list and either

- executes the B-Human’s CABSL-based (c.f. Röfer 2018) behavioural rules in this card, or
- replays the sequence (typically short, i.e. a few seconds) of the pre-recorded actions (see section 3. for the teach-in process), e.g., “FindBallSkill()” followed by “KickAtGoal()” etc.

Then the dispatcher repeats from the beginning

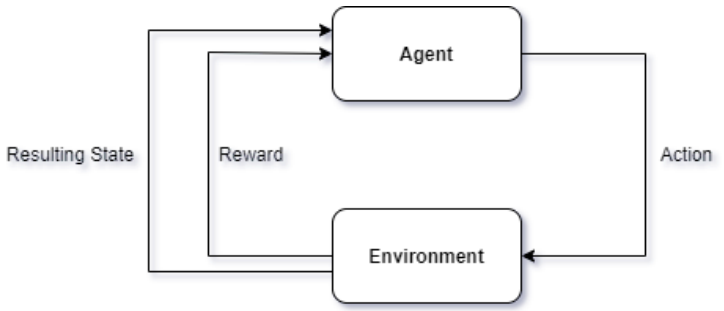
Changing the dispatcher into a Q-Agent

The control unit for behavior control in the B-Human framework is the "dealer" and also known as the dispatcher. To compensate for the discussed disadvantages of the dealer (ToDo: see Section 2), we introduced the "Q-Dealer", a Q-Agent. A Q-Agent learns an action/reward function (or “Q-function”), by being told the expected benefit it will obtain by performing a particular action in a particular state. (Russell and Norvig 2012, p. 961)

forward: next section what is an reward

When the agent performs a certain action A_i in a State S_n, it inevitably influences its environment. That is, every time it interacts with its environment, at the end of the action, the impact of it is measured and evaluated. In the context of Q-learning, the evaluation function is called Q-function Q (S_n, A_i). This determines the benefit obtained when A_i is executed in S_n. This utility represents the reward R.

$$R_{n,i} = Q\ (S_n, A_i)$$



The actions that the Q dealer can choose from are represented by the B-Human cards, just as with the normal dealer. The only difference is that the Q-Dealer considers all cards, both rule-based and Teach In cards, to be of equal value, provided the respective cards are qualified. Therefore, no manual prioritization order is required. The prioritization is done by the Q-Function of the Q-Agent.

Evaluation criteria

The evaluation criteria of a Q-agent determine the reward it receives for the actions it performs. Since these influence the Q-agent's behavior, they should be chosen so that a NAO receives a high reward for beneficial actions and a low reward or no reward for detrimental actions.

Choosing appropriate evaluation criteria in RoboCup SPL is challenging. RoboCup SPL is an only partially observable, sequential, and dynamic environment.

An agent environment is ...(Russell and Norvig 2012, p. 69, p.71)

*"- **partially observable** when an agent has inaccurate or malfunctioning sensors or is unable to perceive parts of its environment and thus does not have all the relevant information to choose an action.*

*- Is **sequential** when the action currently being performed can affect subsequent actions. As a result, long-term effects can occur that can still have an impact on subsequent actions.*

*- Is **dynamic** if the environment can change while the agent is still selecting its action. As a result, the action chosen is not necessarily the best one."*

Let's illustrate these properties of the RoboCup by an example:Image recognition alone is subject to constant interference from fluctuating lighting conditions (partially observable), shooting a ball requires time for it to reach its target (sequential), and even if the NAO does nothing, the other NAOs in the RoboCup multi-agent environment will constantly provide changes to the environment (dynamic).

As a result, the outcomes of the actions are difficult to predict and their evaluation may be influenced by a number of factors that were not taken into account.

talking about xx Therefore, in the first experiment, a single evaluation criterion was chosen, which has observable changes that are also continuous and directly measurable. As a consequence, the NAO is unlikely to learn an optimal game because a number of influencing factors are not taken into account. Instead, it learns to perform actions in certain situations that are advantageous for its pursued goal. This goal is indirectly given by the evaluation criterion.

*talking about xxx*In first experiments we chose the distance of the ball to the opponent's goal as evaluation criterion. The NAO can influence this directly by dribbling or shooting the ball. Even a NAO as a player not in possession of the ball can influence this by positioning himself favorably for a possible pass. Similarly, the effects of actions on this criterion are easier to measure because these effects occur immediately. As a counter example, capturing a goal that has been scored would be an event that can only be perceived some time after the shot has been taken.

For that we defined the state as a set of cards (rule-based and teach-in cards) that were qualified at that time. The localization data of the NAO is implicitly already given with the teach-in cards, which allows it to locally distinguish the state despite missing coordinates in the state space.

As a result the NAO prefered actions, which allowed him to push the ball towards the enemy goal. During exploitation, the NAO, when choosing from several actions, favored the one that brought the ball closer to the goal and deliberately ignored the actions that did not even cause contact with the ball. But a passing game did not result, due to many other influencing factors like obstacles that were not considered in this first experiment.

Training runs in the simulator

KLAUEN

To train the Q-Agent, the simulator is needed, because real tests are both time and resource intensive.

However, the simulator "SimRobot" from B-Human is not suitable for automation, by design. An optimal simulator would make it possible to reset the action A_i in the state S_n directly from the resulting state Sm to the state Sn after the execution of the action Ai. In this way, all conceivable actions in each state could be tested one after the other during the exploration. However, "SimRobot" has not been designed for this. In a sequence of N actions, the first one must always be repeated, even though it has already been explored, just to get to a state that has been completely or partially unexplored so far.

Subsequently, we used different start scenarios for training, in which we executed a sequence of any number of actions and at the end of the sequence automatically reset the simulator to the test scenario in order to then explore another sequence of actions. However, this made training relatively time consuming, as many states were explored repeatedly.

….

5. Improvements in object recognition and image classification

Motivation:

- B-Human JET-Net is based on the YOLO Framework (CNN), but can currently only detect NAOs
- For detection of other objects like balls and field markers, separate algorithms and Neural Networks were used
- JET-Net used lots of techniques to increase the efficiency of the network and allows for fast real time processing of raw images
- The ROBO Framework used a slightly different approach to accurately predict multiple object-types on the playing field, although a lot slower than JET-Net
- Our hypothesis is to unite the detection of all (or most) field objects to a single Neural Network, wich (in theory) allows for easier addition of more object classes because of an End-to-End training pipeline and more efficiency by sharing operations for image processing with a single Feature Extraction stage
- The goal for our first Object Detection prototype was to use the JET-Net Backbone for Feature Extraction and the ROBO Predictor to predict bounding boxes with their respective classes (at first only the Nao and Ball)
- Because we work in an agile environment, we simplify the first proof-of-concept in some aspects to reduce the complexity of the task and development process
- As the project continues, this proof-of-concept will be incrementally extended and optimised to allow for more object classes and greater accuracy

Data Augmentation:

The majority of data utilised for training our object detection model was acquired from the Robocup Team Nao Devils TU Dortmund, which were downloaded from the ImageTagger Domain. One challenge with using data from a single source is that the image data potentially lacks variance. A lack of variance in the dataset can result in the trained model not being flexible enough when being used to detect and classify objects. This is due to various factors that can influence object detection itself, such as the brightness of the lighting on the playing field or the colour of the surroundings.

Expanding a dataset can be difficult and time consuming however. A model requires hundreds or even thousands of images for a single class, all which need to be recorded through various differing factors, such as angles, distance and brightness of the room. In addition, bounding boxes and class labels need to be added manually in order to help the training process with locating the object we wish it to detect, and what class the object belongs to. Due to the time and manual work needed for image labelling being problematic, a different solution was needed

One potential solution that we implemented was to apply Data Augmentation to our dataset. Data augmentation provides more images for the training process, while ensuring that images are not repeating, which would result in creating the problem of a model containing overfitting or underfitting. This is done by taking existing images, making slight modifications to both the image and the bounding box labels, and saving them as new additions to the dataset, in order to artificially increase the size of it.

Original Bild Flipped Brightness Adjustment

Image x: Example of Data Augmentations applied to an image.
Original Image Source:Nao Devils TU Dortmund. 2018

By default, Tensorflow offers data augmentation by utilising either the tf.keras.layers or tf.image functions. It allows for simple augmentations, such as flipping, rotation and resizing of the image. They however do not provide a solution to adjusting the bounding boxes accordingly, which would result in incorrect bounding box placements, and therefore worsening the object detection process. Instead we utilised an external package called Albumentations, a data augmentation library for python, which offers a greater selection in augmentation techniques, such as blurring, adding noise and cropping images, as well as the option to apply data augmentation to bounding box labels alongside their respective images.

Albumentations offers and describes data augmentation in two separate manners: *Pixel-Level Transforms* and *Spatial-Level Transforms*.

Pixel-Level Transforms only change the pixels of an image. Additional data such as bounding boxes are unchanged. Such changes include adjusting the brightness of the image, adding motion blur or shifting the images RGB colour values. These augmentation techniques are easier to use for the beginning due to the lack of need to ensure bounding boxes are modified accordingly.

Spatial-Level Transforms change not only the image, but additional data such as the bounding boxes. Important is to make sure that changes made to the bounding box are matched according to the augmentation utilised. Examples include inverting the image, cropping the image or removing parts of the image. While more difficult to utilise due to the need to ensure the changes with the bounding boxes were applied correctly, they allow for greater and more powerful changes in the dataset.

Albumentations is utilized due to the ease of applying Spatial-Level Transforms, which greatly assist in providing a greater variety of images into the augmented dataset. In addition, it is capable of offering a greater selection of data augmentation techniques, while also being able to handle larger datasets in comparison to the base tf.keras.layers and tf.image options.

Data augmentation in itself can be implemented in two separate ways. One is to take the existing dataset and create new images with augmentations before the training begins. The other is to apply augmentation to the data itself while the training is happening.

The first method attempted was to expand the dataset before training. A function was written that would take all of the images and bounding box labels from the dataset, and create new augmented images and labels from it. The newly created data was then saved into new subdirectories in the dataset, which could then be accessed for the model training process as before.

The results for the first method were initially positive, as newly created images had one or more augmentations on them, while bounding box labels were adjusted correctly. However, the process would increase the size of the dataset by a number of factors depending on how many augmentations were chosen. This would become problematic for people who did not have enough storage space on their local work machines. This method would instead be used to test if augmentation would work correctly, and instead be scrapped in favour of the second augmentation method.

The second method was to create an augmentation pipeline that would run during the model training process. In TensorFlow, a function named the ImageDataGenerator creates real-time batches of data augmented images for the training process. The goal was to create a similar process utilising Albumentations, in order to have data augmentation during the training process.

Compared to the previous method, there is no need to generate new images that take up space. The process of augmentation happens on the go while the training is happening. In contrast, this lengthens the training process by a small factor, and is more difficult to check whether the augmentation is being done correctly, as no data of the augmentation process is saved to be checked on later.

Code Example 1:

```
A.MotionBlur(blur_limit = (3,7), p = 0.4), #Blur Image to random levels
A.RandomBrightnessContrast(brightness_limit = 0.6, contrast_limit = 0.6, p = 0.4),
A.GaussNoise(var_limit = (10.0, 60.0), mean = 0, p = 0.4),
A.RGBShift(r_shift_limit = 25, g_shift_limit = 25, b_shift_limit = 25, p = 0.4)
```

Image x: Example of Data Augmentations techniques using Albumentations.

Augmentation techniques are called upon using the *Compose* class. In it, a number of techniques can be called upon and parameters can be adjusted to one's needs. As an example, the augmentation *RGBShift* allows the user to decide how far any of the separate RGB values can be shifted using the parameters *r_shift_limit, g_shift_limit and b_shift_limit*. In addition, using parameter *p* is the probability that the currently selected augmentation will be applied to the image that is being augmented. This allows for randomization in the augmentation process while also allowing multiple augmentations to be applied on a single image.

Code Example 2:

```
def get_augmented_dataset(path, batch_size, grid_shape, shuffle=True):
    dset = load_tfrecord_dataset(path)

    dset_aug = dset.map(partial(process_data, img_size = 160),num_parallel_calls=AUTOTUNE).prefetch(AUTOTUNE)
    dset_aug = dset_aug.map(set_shapes, num_parallel_calls=AUTOTUNE).prefetch(AUTOTUNE)

    dset_opt = dset.map(lambda x,y: (x, util.preprocess_true_boxes(y, grid_shape)))

    if shuffle:
        dset_opt = dset_opt.shuffle(buffer_size = 512)

    dset_opt = dset_opt.batch(batch_size, drop_remainder=True).prefetch(buffer_size=tf.data.experimental.AUTOTUNE)

    return dset_opt
```

Image x: Part of the Augmentation process to receive augmented dataset.

The augmentation process currently utilised consists of the following steps. First, a set of tfrecord files is loaded. Tfrecord files are the dataset and all needed information that have been converted to a tensorflow binary readable format, which optimises the training process and makes it faster. Data is prepared by using the *process_data* function, which in turn takes our current image and label and applies our augmentation techniques as shown in Image x. These images and labels are saved together for the model training process. Each new training process will augment the images newly, which means that images will be augmented differently each time.

Results

An initial test comparing a dataset with an augmented dataset brought a change in the length of training time for the object detection model. The augmented model required around an extra 50 epochs to train before it would show no further improvements in the training rate. While the accuracy did not increase by much, the conclusion is that the longer training time resulted in being able to detect a further variety of objects in different scenarios.

Testing was first done by loading the model in SimRobot, in order to test whether the object detection model was functional. A live test was done in a testing facility, which consisted of a simple room outfitted with a small playing field and a number of NAO robots. One of the NAO was loaded with the newly trained model, in order to conduct a series of tests, mainly consisting of identifying another NAO placed across it's field of vision. The NAO in front was changed in many ways, such as position and distance. Other factors were also changed for the test, such as changes in lighting and placing other objects next or behind the NAO.

Field testing showed that the model trained with the augmented dataset showed a more successful detection of objects under different lighting conditions, as well as objects from different distances compared to our non-augmented model. In addition, NAO at different angles, such as lying down on the ground, have also received a higher rate of being detected.

Challenges in Data Augmentation

While Data Augmentation provides a simple, yet effective, solution to increasing the size and variety in a dataset, it is not a perfect solution in every scenario.

The first area to pay attention to before using Data Augmentation is to check the quality of the already available dataset. Faulty data, such as bad images or incorrect bounding box labelling, can cause the dataset to worsen when being increased in size with data augmentation. This is due to the augmentation algorithm taking existing data and multiplying it, whether the data is accurate or not. Data augmentation cannot improve the quality of the data itself, it can only increase the size and variety of it. In short, if the dataset is of poor quality, then data augmentation will only create a bigger dataset that is still of poor quality.

Secondly, the types of augmentations that need to be utilised can differ greatly depending on the factors that can influence a given scenario. While lighting and slight image tilting has resulted in an increase of detecting objects, other techniques can cause a decrease in detection if applied incorrectly. For one, inverting images so that objects are upside down is of no use, due to there being no scenario where the NAO robot would stand on its head. Additionally, a limit had to be experimented with on how bright an image could be made before recognition of objects would begin to worsen. In one test scenario for example, too strong of augmentations applied to the dataset resulted in the appearance of false positives on a nearby ladder in the background, which was thought to be a NAO robot.

6. Special: Our approaches to the SPL local challenges of 2021

Obstacle Avoidance

The Obstacle Avoidance challenge was a challenge with many possible solutions. Due to constraints on time, a complex solution was out of the question. The challenge also calls for a solution that can

eventually be adapted to normal 5v5 gameplay, which excludes any approach that simply pre-programs the fastest path for any possible layout. So, to achieve these goals, the challenge was separated into three smaller components: Path planning, Dribbling and a Goal Kick at the end. The Teach-in process described above was also considered as a solution. With Teach-in a solution for every possible configuration of the challenge could have been learned. Because the goal of the Challenge was to create a generalizable solution for any situation, the Teach-in system was not used. Instead, the problem was approached with rule-based behaviour and a planning algorithm.

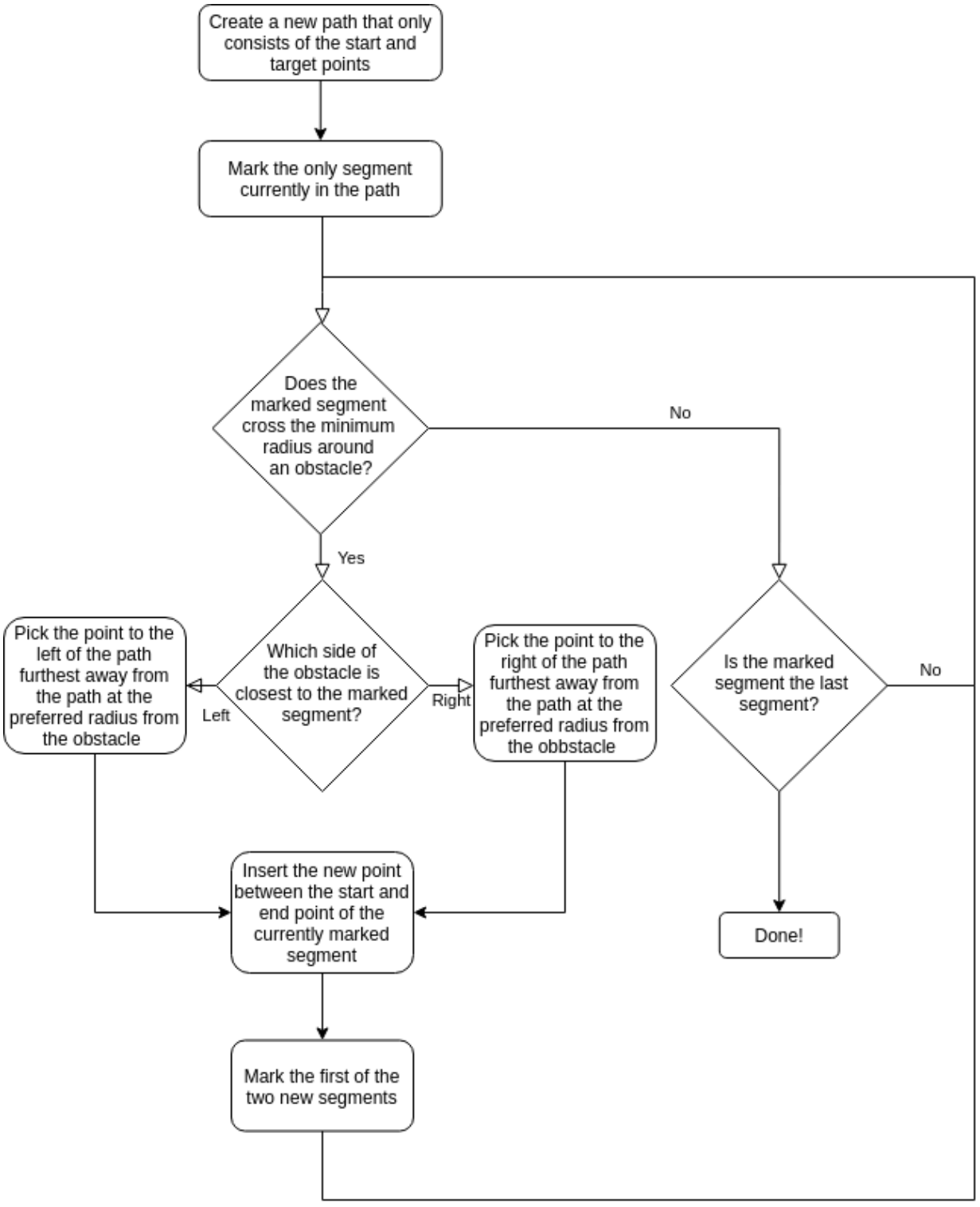
Path planning

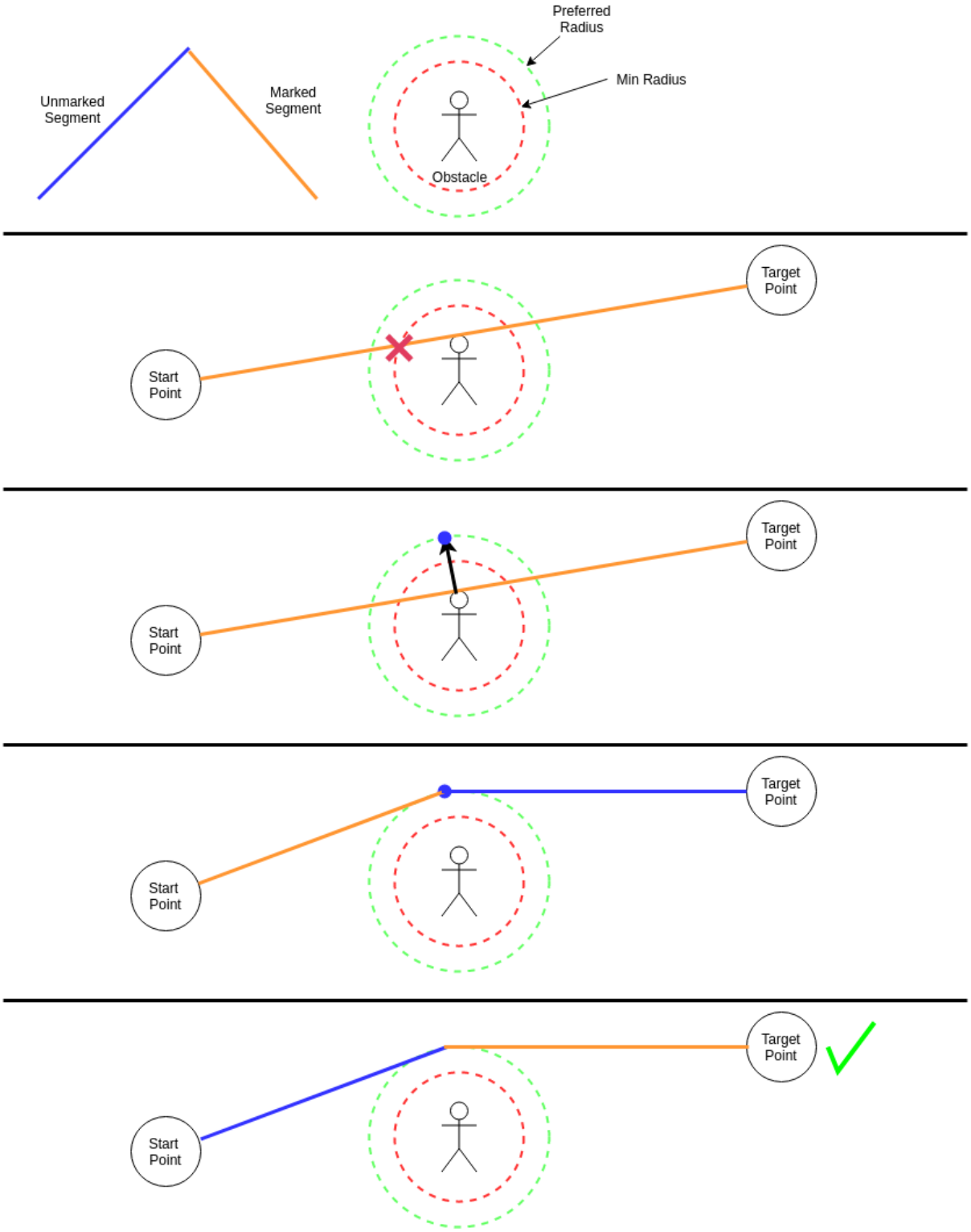
Path planning is a problem already encountered by B-Human. The Path Planner included in the 2019 B-Human Code drop uses an A* Algorithm to find a path for the Robot to walk along. Dribbling poses the additional challenge of having to maneuver the ball around the field as well. This Required that the planned path begin at the position of the ball and that tolerances around obstacles be increased. Due to the Inflexible nature of the B-Human path planning module we decided to create a custom path planner.

A New Path Planner

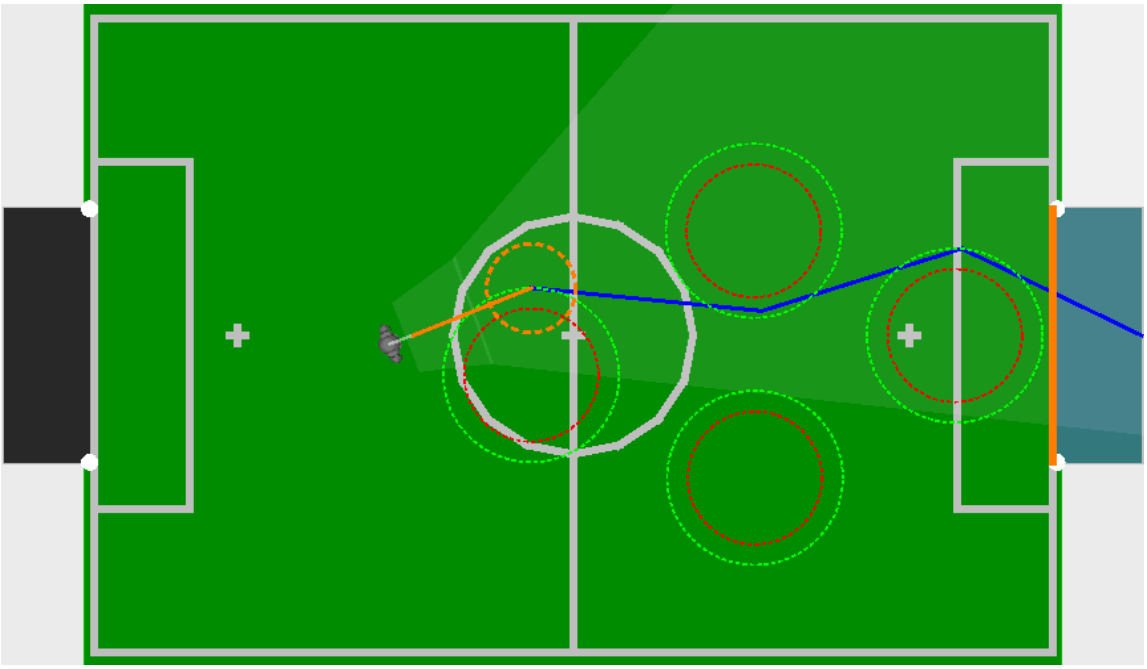
To plan a new Path, three inputs are needed: The starting point, the target and a list of obstacles. Once these are supplied, a path can be generated. A path, In this case, consists of a list of points on the field that will be approached in order. The path that is taken between points is assumed to be a straight line during planning.

Planning occurs following this Flowchart:





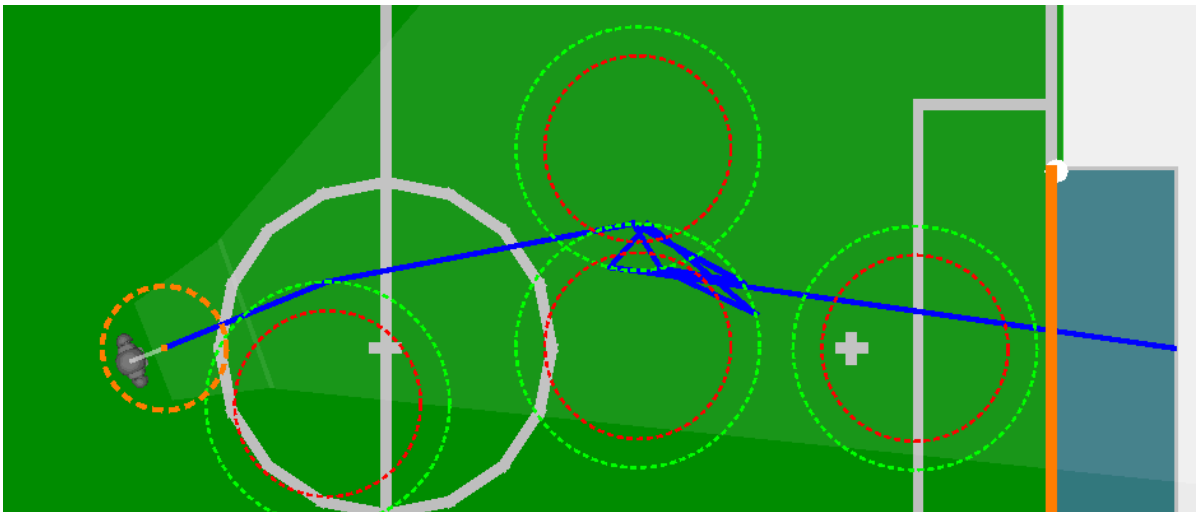
Once a path exists, it is incredibly easy to follow it to its completion. Starting with the first point in the list, The ball is moved toward the current target point by whatever means. Once the ball comes within a certain (configurable) range of the target, it counts as that point being reached. The next point in the list is then flagged as the target. This process can be repeated until the last point on the path is reached.



World State after successful path planning. Blue lines show the planned path. Red and Green circles display the minimum and preferred obstacle distances respectively. The orange line is the path from the ball to the next path point. The orange circle is the area where the ball is considered to have reached the respective point.

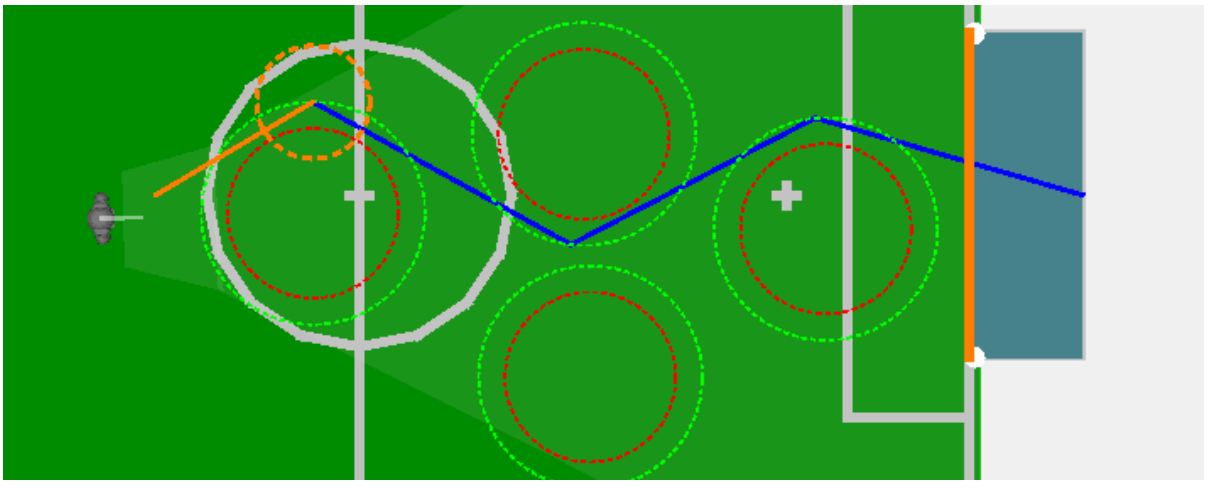
Drawbacks

Of course, a simple solution like this also has its drawbacks. Potential Solutions to these problems are discussed below in the Future Expansions section. Since the Algorithm simply picks a point at a set distance from the obstacle, it is possible for the chosen point to be within another obstacle. In this instance, the planned path would be impossible to complete. This causes a loop in the pathfinding algorithm where the path is recalculated every frame. Furthermore the path also begins recursively wrapping around the problematic obstacles as shown below.



World state in the case of intersecting obstacles showing a broken path

A further argument against this simple pathfinding method is that it can not reliably find an optimised solution. While much of the speed of this method comes from it's simplicity, larger obstacles can cause very wasteful paths to be planned. See image below for an example of a non-optimal path.



World state in the case of intersecting obstacles showing a broken path

Dribbling

Once a path is planned, It has to be followed. Not only with the Robot, but also with the ball. Since the path is made up of individual points, This calls for a dribbling skill that can dribble a ball to a given

point in a straight line. Such a skill can be further broken down into a skill that dribbles in a given direction and one that dribbles to a point.

The first step to dribbling is moving to the correct position relative to the ball. By defining a point near the ball opposite of the direction that the ball should go, the BHuman Path Planner can be used to navigate around existing obstacles and the ball. This will get the robot close to the target position but a finer control must be used once close to the ball. This Zone is defined by the alignment cone as pictured below.

-**Pseudocode!?**

-Alignment cone visual

After some Experimentation with different methods of moving the ball, short kicks using the B-Human InWalkKick engine proved to be the most reliable option. A short kick consists of rotating to the angle given by the skill call, walking to a set offset behind the ball and calling the inWalkKick engine. Each kick moves the ball forward by about 20cm. After kicking the Process of walking to the alignment cone and performing a short kick repeats indefinitely.

To follow a path planned in the previous chapter, the

Goal Kick

According to the Obstacle Challenge rules, once the ball is within **range** of the goal, the Robot is allowed to kick the ball directly into the goal. This is done by using the same alignment tools used by the dribbling skills. Then, instead of using a forward short kick, the robot calls the default inWalkKick built by BHuman.

Future Expansions

As discussed above, The current path planner encounters fatal issues when individual obstacles are placed too close to one another. One method to avoid these problems would be to cluster obstacles that are too close into a single, large, complex shape. By using obstacle clustering, the problematic obstacles can be viewed as a single obstacle and can be successfully planned around.

-path optimization and search

Passing Challenge

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Changes to the Q-Learning approach

7. What went wrong: traps and pitfalls

Intended Reader: Rookies

Magic Number im Teamfunk

Never change the default camera calibration … or get stuck in SimRobot

Automated test procedures, regression testing: not available directly (in SimRobot), so we developed a work around (see Call for Additional Participants, Chap 6.)

Keep global conditions (scenario, scene, set of available skills and their - varying - implementation, global parameters eg. max speed) consistent with teach-in sequences;

Memorizing which teach in data belongs to which game situation (ie bundle sequences by game tactics) tactic tafel wurden zu groß

Teach-in process was cumbersome. Dual input mode (PS4 controller for game play and skill selection ie. action keys + console input for target (as x,y coordinates) (Jonas)

Camera Calibration & Deployment, 1 Nao = Kopf + Body

DVC lost against WandB - why

Data Version Control and WandB:

In order to ease the process of setting up the pipeline for object detection training and downloading the dataset, we attempted to create a setup that would allow individual users to easily train models without the lengthy need of setting everything up step by step through the use of DVC.
Data Version Control (DVC) is a data and Machine-Learning experiment management tool that takes advantage of the existing engineering toolset such as GIT. It allows for the workflow of machine learning to be utilised in GIT, while management of the large dataset and metric data is handled on a remote server through DVC, instead of storing it in the GIT repo itself.

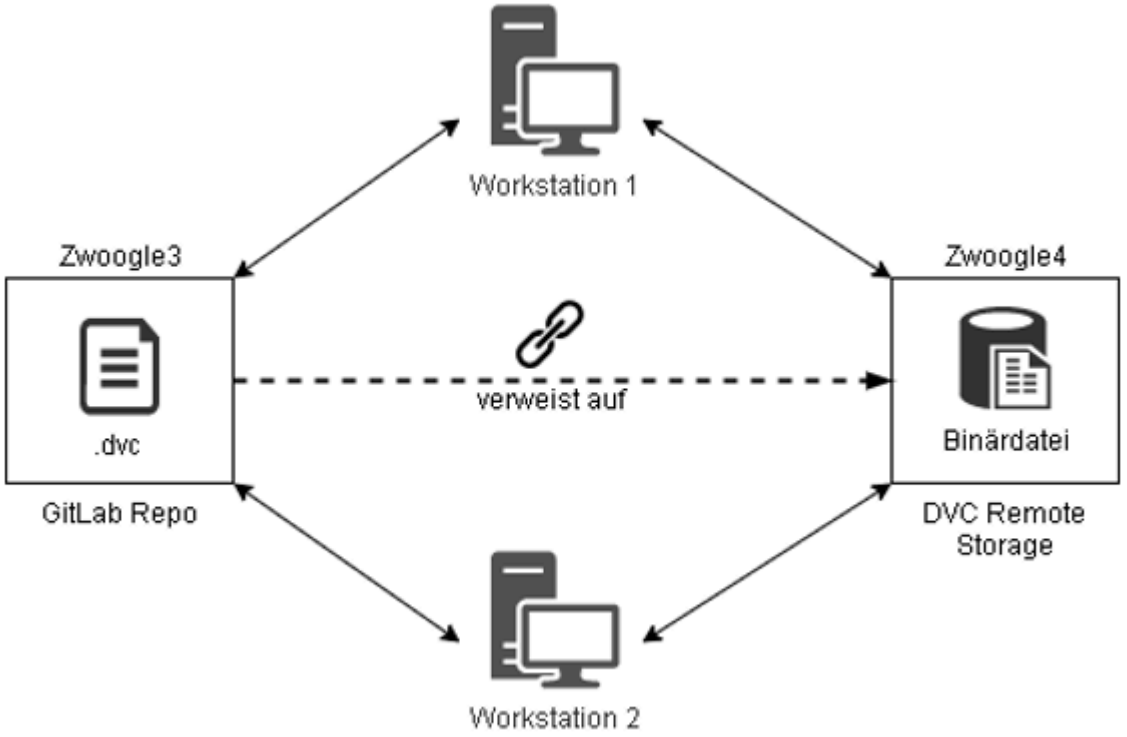


Image x:

DVC runs on top of Git. It works by keeping track of separate files though the use of .dvc files, which in turn can be called upon using Git. Using DVC enables the storage and management of large datasets, which can cause performance issues if stored and utilised in GIT. In addition to including commands that allow for metrics comparisons between old and new models, DVC also allows for the model training process to be carried out using a single command. This way, the process of cleaning the dataset, converting it to a machine-learning readable format such as tfrecord and training the model are all carried out without the need of running each step manually.

At the end however, we would abandon the use of DVC due to a number of minor issues that made the pipeline process complicated. For one at the time, support for Windows was limited on DVC, causing issues that were resolved by writing 2 different object detection pipelines for both Linux and Windows separately.
Another issue came in the form of managing the dataset. Issues related to the need of having 2 datasets available: a sample dataset to test if the object detection process works properly after

installation, and the full dataset for training purposes. DVC had issues trying to keep the datasets apart, and would overwrite the .dvc data responsible for telling DVC where the dataset is stored remotely, causing issues downloading the data.

At the moment, the use of DVC has been replaced in favour of WandB, which possesses a number of data analysis tools for comparing metrics between multiple models, and a remote storage for the dataset to be downloaded separately.

Comparison between DVC and WandB

DVC

Positive

- Dataset Tracking Solutions

DVC enables tracking of files located on a remote server through the use of .dvc files. These files can be used to track directories containing either the dataset or newly created object detection models, which can then be later used for metric comparisons.

Negative:

- Operating System Cross Compatibility issues

At the time of writing, issues exist between the DVC versions of windows, which make cross compatibility between linux and windows users difficult. The windows version of DVC has shown to require other packages that need to be of an older version, or compatibility breaks. Recently however, an update has come out where full windows support has been mentioned, but much testing has not been done at the moment due to a focus on using WandB.

WandB

- Greater Metric Tools Options

WandB provides an online platform in which to save and view any model training progress and compare it for later. In addition, WandB provides its own module for providing callbacks, which are also recorded on the online platform for analysis.

- No Dataset Tracking Solutions

WandB focuses on analysis and comparisons of models trained and therefore does not provide any solutions for tracking and storing datasets on remote servers. A solution is to either store the dataset on a platform with a link for downloading, or combine the dataset tracking of DVC and the data analysis of WandB together.

old.
In order to ease setting up the process for training models with Object Detection, a pipeline was created that utilises Git and DVC, in order for newcomers to the team to quickly be set up and ready.

Currently, this pipeline is supported in Ubuntu 20.04 natively and in Windows 10 through either Miniconda (Python 3.8) or with WSL2. Miniconda is a minimal installer of conda for windows, which allows users to create separate python environments and keep track of each module and their versions. In addition, conda provides a POSIX-like command line interface, which enables DVC to be fully utilised on Windows.

If you are interested in reading about the process, or wish to reproduce the steps taken to install the machine learning process, follow the instructions using the link here. Note that instructions are currently written in German.

Read the R2K wiki!

add: code drop r2k_2021, <link>

Notizen aus der Redaktion vom 21.9.

Hinweis: dies ist B-Human Architektur - dieser Abschnitt ist vor allem für die Rookies geschrieben

- Thomas: Wiki verlinken

- Thomas Nachvollziehbar am Beispiel config - parameter

- Thomas alle Funktionalität waren an Cards gebunden

8. Recent and future work, knowledge transfer to the IT industry

Notizen aus der Redaktion vom 21.9.

Adrian Regel-Anpassungen (Strafraum)

see 7: Keep global conditions (scenario, scene, set of available skills and their - varying - implementation, global parameters eg. max speed) consistent with teach-in sequences; →

see 7: Memorizing which teach in data belongs to which game situation (ie bundle sequences by game tactics)

- Architektur:

- *(recent work) Variierende SIgnaturen der Skills -> mapping erforderlich, Doppelter Check auf korrektes Konfigurieren (Skills via Enum, no programming error, Look-Up in skill registry when firing up the behaviour thread)*

- *(recent work) Trennung file format Recording Trigger Points vs. file format playback sequences, give sample for directory schema with mnemonic path names, simplify code, maintainable*

- *(future work) see 7. "Memorizing which teach in data belongs to which game situation (ie bundle sequences by game tactics)"*

→ - avoid miss-configuration x gameplay tactics ("Unterzahlspiel")

→ Adrian Konzeption Data Mining, TeamBehavior, Tactics und DataMining Integrieren, provided multi teach-in sequences per trigger point (serves as base for q-learning), TI-Cards Playback erneuern (skill abort, isDone(), timing concept für teach-in playback ie max-Time)

see 7 Teach-in process was cumbersome

→ fancy: farbige markierung (DEBUG_DRAWING) auf Feld, bspw. aktuelle tile

New Skills like "WalkToBallAndKickToMate.cpp "

Felix Anfragen an abat+: Konzeption Abbildung Produktionssteuerung

Thomas Sequenzanalyse, Performanz Verbesserung durch Integration MOE incl. Ball in in UpperCamera thread

Simulator "ROSE" , github entry

Regression Test

Regression Testing

In order to ensure that any changes in the future have a positive impact on our development towards participating in the RoboCup, any additions and improvements will need to be compared with previous versions of our system. In order to accomplish this, a plan to implement regression testing is in progress.

The idea is to set up a selection of test cases, which each contain a specialised scenario, in order to test changes or improvements made in the code itself. For example, a recent change in the RoboCup pertains to the reduction of the amount of messages that can be sent between the NAO robots in one team down to 20%. This means whereas before each robot could send a message to the rest of the team each second, each team can only have one robot send a message per second now, for a total of 1 message every 5 seconds.

To check how much of an impact this has in the overall playing capabilities of the NAO team, a test comparison between the old and the new change will have to be made. That way, any improvements needed can then be done afterwards, whereas the improvement will be tested anew to see if we can achieve previous results through optimizations. Such regression tests will also be done for future changes, such as a potential move to the new B-Human 2021 code base.

Sequeuz Analyses TC “Video”

< hybride, teach-in>

“Boosted Gradient Tree” , 15Hz , dynamic load balancing

modul “vision pipeline” ohne nn <who>

rule adapt: ebc

dribbling algorithm + new walk engine = performant?

Data Version Control and Work on a pipeline for easy setup for machine learning

Old

The process goes through installing a python environment, installing the necessary packages into the environment, cloning the git repository containing all files necessary for the model training process, and using dvc to both download the dataset and to save the results and calculate the metrics data.

Future Challenges in Dataset:

While first attempts at data augmentation have shown positive improvements in object detection, a couple of issues were discovered that still remain to be solved.

The first issue pertains to class imbalance in the dataset. Images mostly contain more instances of NAO, while the ball itself only ever appears one at a time at most and never otherwise. This results in the model being better at detecting NAO robots, but having a more difficult time detecting the ball. This issue is currently being worked on by Thomas, who is selecting datasets focusing on containing balls, in order to reduce the imbalance. Other solutions that could solve this by modifying the dataset or the training process in order to balance out classes are still being searched.

Another Challenge is the variety in the background of images in the dataset. In testing, certain background elements, such as the legs of white chairs would occasionally be detected as NAO. This is as mentioned above (Chapter 5) due to the dataset largely being derived from a single source. Repetitive elements in the background behind our classes of objects, such as a NAO and the ball, cause said elements to be identified to belong to one of the classes. This causes issues in recognition, in that these elements are incorrectly identified as belonging to one of our object classes.

Connor Data Augmentation neue techniques

Future Challenges in Data Augmentation

Data Augmentation is challenging, in that the augmentations utilised have to appear in real scenarios for the NAO. Currently, augmentation techniques such as brightening the image assist with the process of including varying lighting conditions. However, colour temperatures, such as lights coming from different sources (natural sunlight, white LED lights, warm lights, etc) also influence object detection. In addition, the lack of varying backgrounds in datasets cause certain background elements to be seen as belonging to one of our object classes.

While Albumentations offers plenty of augmentation techniques, new ideas for augmentation will need to be thought of. One idea would be an augmentation technique, in which objects located in an image are placed over new backgrounds, in order to add variety. Another idea would be to have 2 images placed on top of each other, in order to combine them together in order to add more variety.

Acknowledgements

THX to Jonas, Markus, Daniel, …

9. Bibliography

Thomas Röfer; Tim Laue; Andreas Baude; Jan Blumenkamp; Gerrit Felsch; Jan Fiedler et al. (2019): B-Human 2019 - Team Report and Code Release 2019.

Röfer, Thomas (2018): CABSL – C-Based Agent Behavior Specification Language.

Russell, Stuart J.; Norvig, Peter (2012): Künstliche Intelligenz. Ein moderner Ansatz.

4. Chalup, Stephan, et al. "RoboCup 2019: Robot World Cup XXIII (Lecture Notes in Computer Science, 11531)." The ROBO Architecture - Training, 1st ed. 2019, Springer, 2019, pp. 314–16)