HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

FAKULTÄT ELEKTROTECHNIK UND INFORMATIK

HOCHSCHULE RAVENSBURG-WEINGARTEN

Bachelor thesis in applied computer science

Change Detection in Shrubland Areas on High Resolution UAV Video

Erkennung von Veränderungen in Buschlandgebieten auf hochauflösendem UAV-Video

Author:	Tom Segbers
Supervisor:	Prof. Dr. Daniel Scherzer
Advisor:	Dr. Franziska Funk
Submission Date:	28.02.2021

I confirm that this bachelor thesis in applied computer science is my own work and I have documented all sources and material used.

Weingarten, 28.02.2021 Tom Segbers

Abstract

The increasing performance and rapidly decreasing price of consumer grade drones equipped with integrated camera systems are opening up new avenues for the industry. Together with the decreasing demands of neural networks on hardware, the possibilities for the interaction of optical systems with automatic processing are almost limitless. One of the basic building blocks of many systems is the detection of a change in images. This simple-looking task enables systems connected in series to process the detected change, for example, to recognise objects.
The aim of the work is to develop such a system which can realise this first step. It should be possible to monitor the movements in an observed area with the help of the integrated camera of a flight system.
Although this domain is already well researched, there is a lack of publicly available data sets that are needed to train reliable neural networks. The question therefore arises whether simulated data can be used to train a system which then nevertheless delivers relevant results in the real world.
For this purpose, data sets were first generated with various freely available game engines. Then a neural network was trained on this self-generated data. Finally, the trained model was validated on existing data and the results were compared with those of previous work.
The trained system "ISEEU-NET" is finally able to compete on par with current technology while meeting real-time requirements. It is possible to use the system to detect movement in a monitored area, even when the aircraft is moving in the area of interest. The system is resistant to small movements of the camera, as well as sudden lighting changes and can even detect the movement of relatively small objects, such as a dog.

Kurzfassung

Die steigende Leistungsfähigkeit und der rapide sinkende Preis von Consumer-Grade-Drohnen, die mit integrierten Kamerasystemen ausgestattet sind, eröffnen der Industrie neue Wege. Zusammen mit den sinkenden Anforderungen von neuronalen Netzen an die Hardware sind die Möglichkeiten für das Zusammenspiel von optischen Systemen mit der automatischen Verarbeitung nahezu grenzenlos. Einer der Grundbausteine vieler Systeme ist die Erkennung einer Veränderung im Bild. Diese einfach aussehende Aufgabe ermöglicht es hintereinander geschalteten Systemen, die erkannte Veränderung zu verarbeiten, um z. B. Objekte zu erkennen.
Das Ziel der Arbeit ist es, ein solches System zu entwickeln, das diesen ersten Schritt realisieren kann. Es soll möglich sein, mit Hilfe der integrierten Kamera eines Flugsystems die Bewegungen in einem beobachteten Gebiet zu überwachen.
Obwohl dieser Bereich bereits gut erforscht ist, mangelt es an öffentlich verfügbaren Datensätzen, die für das Training zuverlässiger neuronaler Netze benötigt werden. Es stellt sich also die Frage, ob man mit simulierten Daten ein System trainieren kann, das dann in der realen Welt trotzdem relevante Ergebnisse liefert. Zu diesem Zweck wurden zunächst Datensätze mit verschiedenen frei verfügbaren Game-Engines erzeugt. Dann wurde ein neuronales Netz auf diesen selbst generierten Daten trainiert. Schließlich wurde das trainierte Modell an vorhandenen Daten validiert und die Ergebnisse mit denen früherer Arbeiten verglichen.
Das trainierte System ISEEU-NET ist schließlich in der Lage, mit der aktuellen Technologie mitzuhalten und gleichzeitig die Echtzeitanforderungen zu erfüllen. Es ist möglich, das System zur Erkennung von Bewegungen in einem überwachten Bereich zu verwenden, selbst wenn sich das Flugzeug im interessierenden Bereich bewegt. Das System ist resistent gegen kleine Bewegungen der Kamera sowie plötzliche Beleuchtungsänderungen und kann sogar die Bewegung von relativ kleinen Objekten, wie z. B. einem Hund, erkennen.

Contents

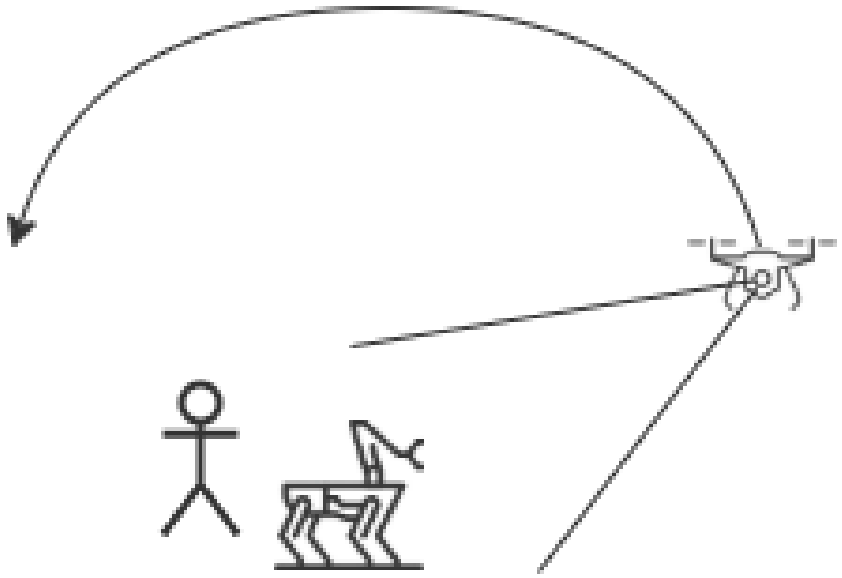


Figure 1.1.: Example scenario: Drone flies around person and robot dog

1. Introduction

1.1. Motivation

Almost every animal has the same reflex: if it notices a sudden change in its environment, it's either fight or flight. While this basic survival instinct mostly relies on reflexes deeply entrenched within our very nature, computer vision took some inspiration from it. It enables systems to react quickly to their surroundings and is often used as a prestage for more sophisticated algorithms like tracking or identification. For this thesis, the domain of change detection is applied to detect moving objects in a scrubland environment. These objects are often humans or vehicles moving through a moderately vegetated area as seen in 1.1. These artificial objects in the scene are to be reliably outlined and annotated. The real challenge begins when we take a look at the intended area of operation, which is aboard an unmanned aerial vehicle (UAV). In this work, moreover, scrubland was selected as the area to be observed because, the tall vegetation causes undesirable background movements.

1.2. System Requirements

1.2.1. Functional Requirements

As the system is intended to be used onboard an unmanned aerial vehicle (UAV), the output that the system produces should be both easily usable by further algorithms aboard the UAV as well as provide visual help to any ground crew or operating personal. Thus, the system should output a heat map where every pixel annotated either corresponds to a changed or an unchanged pixel. This heat map should be computed from a continuous Full HD RGB video stream that is fed from the onboard electro-optical systems of the UAV. If an object only partially moves, the full object should still be annotated. The system should be able to handle a variety of video resolution and provide an easy-to-use API. The output of the system should also be a continuous Full HD video stream, although it is sufficient that this stream is in grayscale colours. A general robustness against illumination changes, small motion of the viewfinder and other interferences of the input image needs to be integrated. Background objects such as trees, shrubs and grasses should be ignored.

1.2.2. Non-functional Requirements

Moreover, the system should be so lightweight that it can continuously inference on a NVIDIA JetsonNX with at least 30fps. This includes possible pre- and post processing. It should also be possible to run the system continuously for more than 3 hours without having to restart it.

1.3. System Environment

The system is a middleware between a hardware electro- optical image input and a software interface for easy access. The interface is a passive component that notifies the user when a change is detected from the system and then displays the information where the change has been noticed. As seen in 1.2 the system dubbed "ISEEU-Net" recieves direct input from the connected optical image system. The output of the system can either be fed to other systems, such as object identification and tracking algorithms, or back to an operator who commands and controls (C&C) the aerial system. All of the calculations required by the ISEEU-Net system are done onboard the Unmanded Aerial Vehicle, as specified in the requirements .

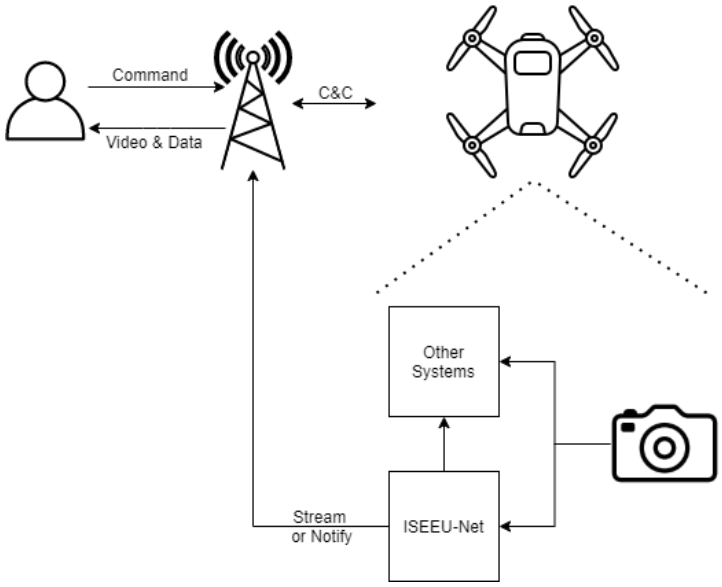


Figure 1.2.: System Environment

1.4. Structure of this Work

First, in chapter 2 we will lay some foundations for a common understanding of the topic. Discuss some challenges and criteria. In chapter 3, we will take a look at what the current tech offers, what has worked historically in the domain of change detection and what we can learn from that.
After the fundamentals, our datasets are introduced in chapter 4, as well as a first deep dive is given into synthetic data generation.
Then the proposed method and its implementation are presented together in chapter 5 with all the necessary selection and preprocessing steps necessary to get the system ready for training.
Training results are then reported together with some highlights that improved the overall results in chapter 6.
To close the work, we evaluate the trained model on some benchmarks and analyse the possible strengths and shortcomings of the system in chapter 7 as well as give a outlook to good future work options in chapter 8.

2. Fundamentals

In this chapter we lay the foundation for the methodologies of this work, define frequently used concepts and point out some challenges that are addressed in later chapters.

2.1. Input Data

2.1.1. About Change in Images

When a definition is made of what change is in an image, the first thing that comes to mind is the pure, pixel-by-pixel change between the values of two pixels that have a relationship to each other. Defined after [1], this pixel relationship is often either spatial or temporal.
Temporal relationships between pixels are the most easily computed, simply take the encoded value of a pixel (eg. RGB or HSV Values) and compare it to the same pixel in the next frame.
Spatial relationships are a little more difficult and have recently gained a lot of attention in the machine learning community. In essence, spatial relationships identify related pixels like the parts of a car. As spacial relationships are essential to detecting and segmenting objects within an image, extensive research has been conducted in the field of image segmentation. The field of image segmentation focuses on identifying coherent fields of pixels on a per object basis and is used in a wide variety of applications ranging from medical to autonomous vehicle applications.
A real problem arises when we combine temporal and spatial aspects. This results in the detection of objects over time, some of which are moving and sometimes even the camera recording the image is moving itself!
We get the domain of change detection when we remove parts of the spatial relationships between frames to include only segments that moved between frames.

To summarise: Change detection trys to extract temporal and spacial relationships from frame sequences while retaining the information of the full object. Figure 2.1 illustrates this principle.

Most of the reasons changes occur in a video feed can be classified either as a result

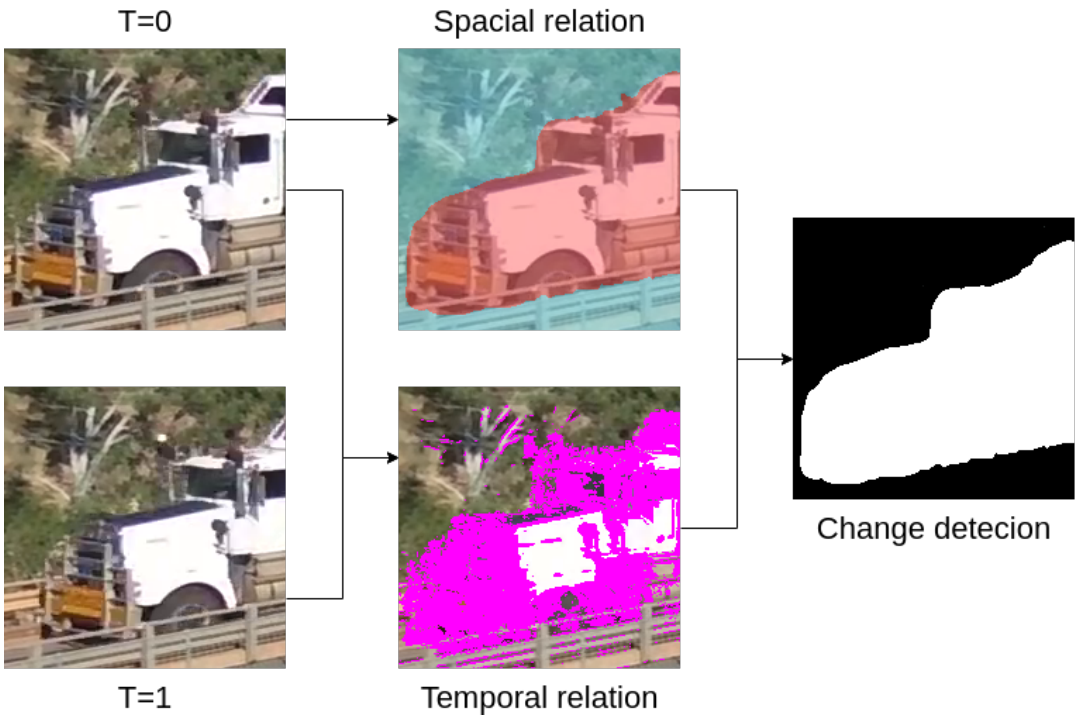


Figure 2.1.: Change = Spatial + Temporal Relationships

of the recording system or as a result of external factors. A overview of the most common causes why change occurs in an image are listed here:

External

E0. Local Environment

E1. Weather

- E1.1 Precipitation (Rain, snow, ...)
- E1.2 Wind
- E1.3 Bodies of Water

E2. Lighting Variation

- E2.1 Daytime
- E2.2 Overcast

E3. Object

- E3.1 Object Movement

E3.2 Object Size

E3.3 Object Color / Camouflage

Internal

I1. Ego-motion

I1.1 Aircraft ego-motion

I1.2 Relative Camera Motion

I1.2 Sensor

2.1.2. Limitations of the System

All mentioned points are causes of change, but can also influence how effectively ISEEU-Net works. As with many things, the excessive occurrence of any of the mentioned phenomena can interfere with reliable detection. In the following sections we want to explore some of these limitations that are inherited from the full UAV system or artificially set to limit the scope of the project.
Later in chapter 4 we are going to take a look at how we can compensate some of these limitations

Local Environment

While there are a lot of sources that can cause changes between two images, only moving objects are of interest to us. Background objects, like trees and bushes, that usually produce changes in harsh weather conditions like wind [E1.2] should be ignored by ISEEU-Net. In the same way, weather conditions like rain or snow [E1.1] pose a major challenge to the system, as they cause constant noise in the picture and limit the view quality. As the ISEEU-Net is designed to be used on board UAV systems, which usually fly in good weather, we also set good weather condions as a requirement for the system and thus will later only train with good weather data.
For simplicity, the allowed observed space is also limited. The system is designed to view a mostly 2D plane from above with a view angle of above forty-five degrees. An example of this can be seen in 2.2. However, when we look at how a UAV system normally operates, and especially at normal flight levels of above 100 Meters this limitation becomes insignificant.

The texture of the observed area also matters. Water and wet surfaces often have reflective and quickly changing textures which can confuse detection systems [E1.3]. See 3.5a for an example of an extremely reflecting surface.
As water surfaces present their own unique challenges, training data with water in it

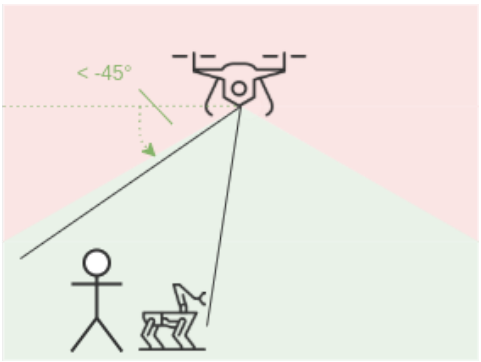


Figure 2.2.: Example of the angle limit imposed on the system

was explicitly excluded. However, in the results chapter we will also look at some cases where the observed areas also include bodies of water while the ISEEU-Net is checking them for change.

Lighting

Likewise, natural weather conditions like sudden changes in lighting [E2.] due to overcast [E2.2] should not make the system less stable or impact detection in any significant way.
As unmanned aerial vehicles get more and more sophisticated, their scope of usage also increases. Thus, the system should be independent of weather conditions, although it will of course only work as long as the optical system can produce a clear enough image of the surveilled area. Nonetheless, the system should also be able to work with infrared or low-light images taken at night.
However, fact is that the system is trained only with daylight images and thus will experience performance degradation while performing with other types of data.

Viewed Object

The most important cause of change is a moving object. That is what we want to detect and annotate.
Of course, there are many factors that influence objects and their corresponding change in images. The most notorious factor is probably the size of the moving object [E3.2]. It ca be both a blessing and a curse. If the object is too big, the system will have trouble to extract the correct special information for the object. If the object is too small, the system might not even detect it. In section 2.1.2 we will explore the theoretical minimum size for a object.
But not only sizes make the extraction of spatial sizes difficult, the texture and colour

Figure 2.3.: CD2014 video "Canoe"[2] with pixelated water in the background

of an object can also significantly sway the results. For example, a white fox will be much easier to spot while running on greenery than running over snow.
Of course, the movement itself also plays a crucial role, most importantly the movement speed in correlation with the frame stack size fed into the system. A fast movement that equals to a big change is clearly easier to detect.
In chapter 6 and chapter 7 we test and explore how big an object has to be to be reliably detected.

System Intern However, not only external factors can pose challenges, but also the camera itself as well as the system in which the camera system is integrated can make reliable detection of motion difficult. When the camera or other electro-optical systems are mounted onto a moving platform such as a UAV or HAPS, the ego-motion of the system can be quite dramatic [I1.1]. A reliable change detection system needs to be able to compensate for such movement in order to maintain a stabilised input, filtering out 'false' detection of changes resulting only from ego-motion. Ideally stabilisation should already be taken care of by other systems before the input reaches the ISEEU-Net, in order to relieve the ISEEU network and achieve a better separation of the areas.
When we look closer at the electro-optical system itself, we can easily identify a

couple of weak points that can influence the reliability of the detection system, such as: Sensor quality, camera motion and vibration compensation, zooming or changes in FOV options. However, as these risks are mostly hardware related, it can be very hard to compensate for these risk in software systems and thus will mostly be ignored in this work.
To ensure that the temporal relationship between the pixels is maintained, a strictly sequential ordering of the information content must also be ensured. In the context of a continues video stream, this means that changes to the order cannot be permitted. However, dropped frames can be allowed and even useful. A big time difference between the frames makes it possible to detect smaller motions, as a larger time window is available for observation. On the other hand, larger time frames can also reduce recognition, since they then amplify all other interfering factors, such as ego-motion.
A sufficiently large image or video resolution is needed in relation of the FOV to capture enough information about objects so that they can be reliably detected. One reliable metric for determining ground resolution is the so-called Ground Sample Distance (GSD)!

The Ground Sample Distance is the **lower** of the following two values:

$$GSD_h = \frac{\text{Flight Height} * \text{Sensor Height}}{\text{Focal Length} * \text{Image Height}} \tag{2.1}$$

$$GSD_w = \frac{\text{Flight Height} * \text{Sensor Width}}{\text{Focal Length} * \text{Image Width}} \tag{2.2}$$

In [3] Adam Van Etten showed that a GSD of 15cm per pixel is more than enough to reliably (F1 = 0.92) detect vehicles with a crossection of 3m. Derived from that assumption, if we want to detect a person with a cross-section of 15cm (from the top) we need a GSD lower than 1cm/px.
Of course, this value becomes even smaller for tiny objects like a rabbit running over a field.
As previously mentioned, the input data should ideally be stabilised. This means that the stream of incoming data and, more precisely, the camera section (or FOV) change between two frames should be minimal. This already outlines one of the limitations of the system, if the change in camera view is too big, especially bigger than 2px per frame, the system will not be able to handle it. The reason for this behaviour will be explored in a later chapter.
To minimise the effect of unstabilised input data, the training data was augmented with randome crops at each border to simulate light shaking of the recording sensor. We explore this technique well in section 5.2.5.

Other considerations

The previous risks and challenges can all be categorised as 'passive', meaning that they are impacts that result from passive actions and/or are a consequence of other, unrelated activities. Of course, there are other considerations, like active camouflage or the usage of the target attacks on neural networks, but these are beyond the scope of this work.

3. Related Works

In this chapter, a brief overview of relevant and recent research is given. For a more in-depth dive into the field of change detection, the study by sehairi et al. [4] and the survey by Komagal und Yogameena [5] are highly recommended.

3.1. Overview of Computer Vision Tasks

Generally, computer vision tasks can be categorised into one of four categories, they can be seen in 3.1. Image classification tries to identify what kind of image it is or what kind of objects an image contains. Object localisation tries to additionally extract location information from the image to segment and localise object roughly with bounding boxes.
Segmantic segmentation ups the game by segmenting objects pixel by pixel according to their category, often making it impossible to distinguish individual instances of objects.
Instance segmentation combines all of the previous techniques and aims to segment each occurring instance of an object pixel by pixel while labelling it with the correct class.

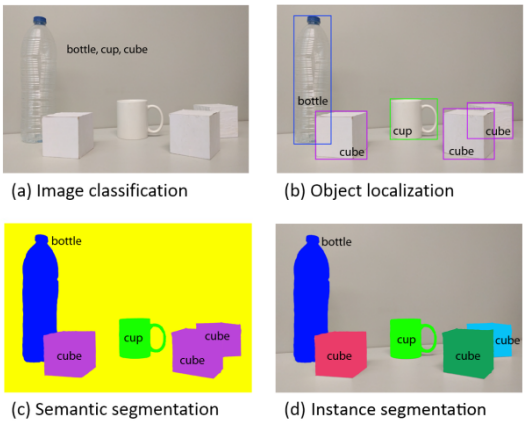


Figure 3.1.: Overview of computer vision tasks, Fig1 from [6]

The task of this work lies in the domain of semantic segmentation, even tough we only have two classes: Changed and unchanged pixels. Similar binary classification

problems are already well researched, notably foreground segmentation and background subtraction are related semantic segmentation tasks that aim to distinguish the background of an image from the foreground.

Change detection adds an additional challenge, not only to consider objects in the foreground, but also to consider only when the object moves between the reference frame and the current frame.
As described in [7] and can be seen in 3.2, all BS algorithms are working in 3 distinct phases:

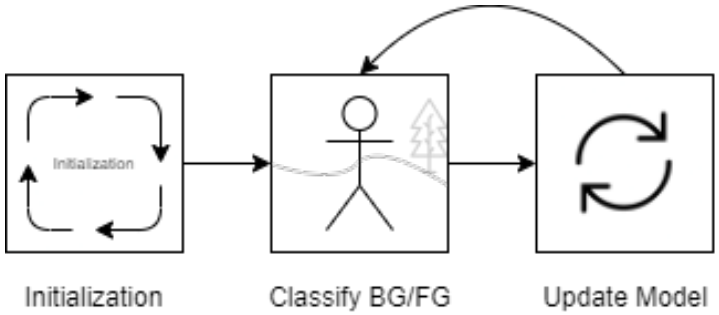


Figure 3.2.: Phases of a BS algorithm [7]

3.2. Traditional Approaches

Of course, the topic of change detection is broadly covered by many researchers since 1960. As the first of their kind, GMM or Gaussian Mixture Models were used to detect changes in low-resolution video feeds. In this section, we will take a look at some traditional approaches that performed reasonably well on a public dataset and benchmark avaliable on `changedetecion.net`(CD2014 [2]) that we will introduce later in the chapter.

3.2.1. GMM - Gaussian Mixture Model

First used in Staufer Grimson et al. [8] in 1999, and later improved by [9] with the improvement of the equations and the introduction of a novel shadow detection scheme. They work by modeling each pixel with a combination of Gaussian distributions and updating those with an on-line approximation.
Zivkovic et al improved GMMs even further [10] by constantly changing the number of Gaussian distributions for each pixel, whereas previous iterations used constant numbers. This improved stability and sometimes even increased inference times.

Elgamall et al. [11] in 2020 chose a non-parametric approach based on KDE (Kernel Density Estimation). In this approach, each pixel is assigned a probability of change. Results where quite good for the limited computing power available at the time. The novel idea to use pixel-based probabilities persist to this day.
An more recent and quite successful 'traditional' model is the SuBSENSE model proposed by [12] in 2015. It introduces a pixel-wise thresholding,evaluating a surrounding grid of 16 pixels. This approach is quite similar to the basic principles of Convolutional Neural Network as it captures spacial and colour information. Thus, if the selected pixels exceed a certain threshold, the current pixel is categorised as either foreground or background.

3.2.2. Background Movement Compensation

To tackle the problem of moving backgrounds, e.g. due to the ego-motion of the camera or harsh environmental conditions, methods have been implemented to compensate for these often undesirable effects, which often lead to false detection of changes.
In [13] Lee et al. use a feedback process to reprocess false positives. They calculate the "blink" value of an area aka how often the area is changed on average per frame. When this value is above a certain threshold, the area is marked as dynamic and is completely put in the background.

3.2.3. Static vs Moving Camera

Traditionally, most change detection algorithms rely on static or uniformly moving cameras, like regular survaliance cameras, to accurately predict change. However, the intended use case for ISEEU-Net is on board a moving UAV, where movements can be quite unpredictable and turbulent at times.
One approach to compensating for a moving camera was the use of "outside" information. Common information used for image alignment purposes is GPS location sequences and motion estimation to calculate the required image alignment steps for an accurate overlap of sequence images. If no metadata is available, image alignment can still be achieved, although this process is often quite computationally intensive and a separate area of expertise. A brief overview of the topic of image alignment is given in [14].

Ego-motion Compensation The authors of [15] use image alignment for moving cameras compensation. After aligning a sequence of images, called image stack, a

trained CNN is used for change detection.
Limitations of this approach occur, of course, when the ego-motion of the camera is large enough that the alignment algorithm fails to align the images. Additionally, most image alignment algorithms are prone to error when there are large illumination changes between image pairs.

3.3. Machine Learning Approaches

Of course, the signifant success of machine learning approaches in other areas of computer vision has also had an impact and brought about numerous improvements in the area of change detection. Most notably, the success of Convolutional Neural Networks (CNN) has had a profound effect on the topic of change detection. As of now (2020), all of the top 10 algorithms used on the CD2014 Dataset [2] rely heavily based on CNN's.

3.3.1. CNN based

[16] was the first one to use CNN in background subtraction in 2016. They extract a gray scale background image from several initialisation frames with a temporal median operation. Then a CNN with filter size 27 [sic!] and step size 1 is used. The input is the current image plus the initialised background. While the idea of the approach seems reasonable, for our use case it is not possible to create an initialisation or reference frame as the ego-motion in our system constantly changes the background and the crop of the recording.

[17] used a different approach, a cascading CNN with multiple scales was used. This resulted in the input being scaled down to 0.5, 0.75 and 1.0. All three are fed in a basic CNN with 4 conv. layers and 2 fully connected layers at the center. Then the images are upscaled again and added up in an average pooling layer. This procedure results in some very good results with an average F1-score of 0.9770 on the CD2014 dataset[2]. Although this seems incredibly good, it has to be noted that the model has been trained on a subset or the complete data on which it was later evaluated. This basically invalidates the score for a general comparison.

[18] also use U-Net, but the input consists of the current frame plus two additional frames with different time scales. This is also a difficult decision for the case of moving cameras, as it cannot be known whether the selected reference frames are unchanged. However, the decision to take reference frames from different time frames instead of a sequential approach is a novel one.

[7] propose a novel multiscale fully convolutional network (MFCN) that can run in real-time and uses transfer learning from a previously trained model to fine-tune

only on the desired dataset. The concept is two-staged:

1. Training stage: Use "a few" input frames with GT label as fine-tuning training
2. Inference stage: Use the fine-tuned model for segmentation of BG and FG.

The problem here is that they "initilise" the model with the first 3000 frames of a desired video. From there they use 200 frames and train the model on those frames. This results in a non-comparable result as this is basically over-fitting on the validation set. Although they have proven that the architecture of the U-Net[19] is very much capable of tackling the topic of change detection.

3.3.2. FgSegNet

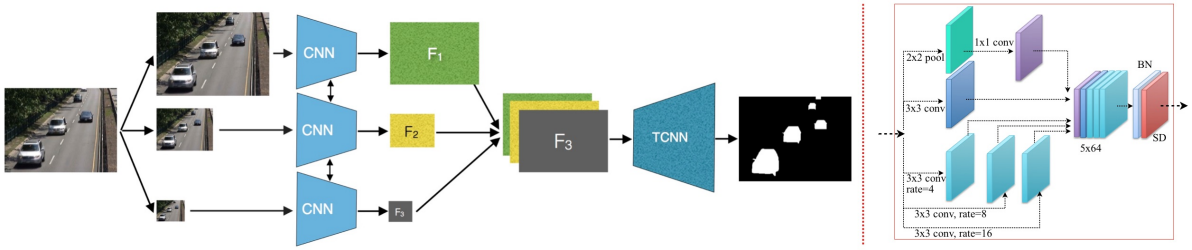


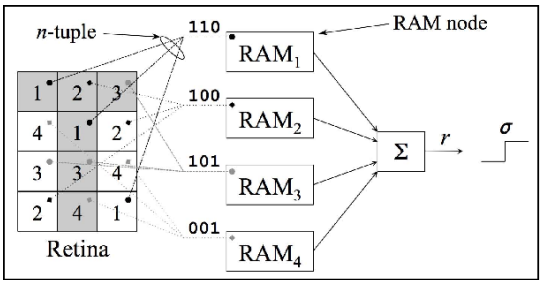
Figure 3.3.: The FgSegNet architecture [20]

The FgSegNet model[20] uses the principle of different sizes of images to extract lower and higher level information from one image. As seen in 3.3 the model rescales images to 2 smaller sizes, feeds all 3 images (org, 0.5x, 0.75x) trough 3 parallel CNNs (shared Weights) , rescales and concatenates the results and feeds them trough FCNN to get a prediction map. The CNNs used are modified VGG-16 architectures with added dropout and removal of FCNN layers. “We reduce the learning rate by a factor of 10 if the minimum validation loss does not improve after 6 epochs.”[20]. The early stopping used here is also very useful not only to combat overfitting but also to save training resources.

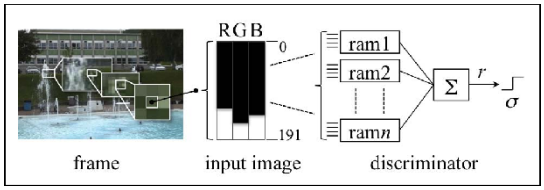
3.3.3. Weightless Neural Network

In [21] a radically different approach to neural networks is proposed. In their paper [21] use a weightless neural network named "CwisarDH" for change detection. The principal behind the system is that the input (Retina) is sliced into random 3-pixel-groups. Each pixel corresponds to one group (RAM Node). The RAM node receives the value from the corresponding pixel (either 1 or 0). The contents of the RAM

nodes are then thresholded and summed up. The resulting value "r" represents a confidence interval if the central pixel in the retina is in the foreground. 3.4 illustrates these principles.



(a) [21] Fig 1. - A WiSARD discriminator.



(b) [21] Fig 2. - CwisarDH input encoding

Figure 3.4.: The Important Components of the Weightless Neural Networks

Additionally, CwisarDH[21] introduces a history of each pixel, which further improves the performance on unseen videos as temporal pixel information is also considered.

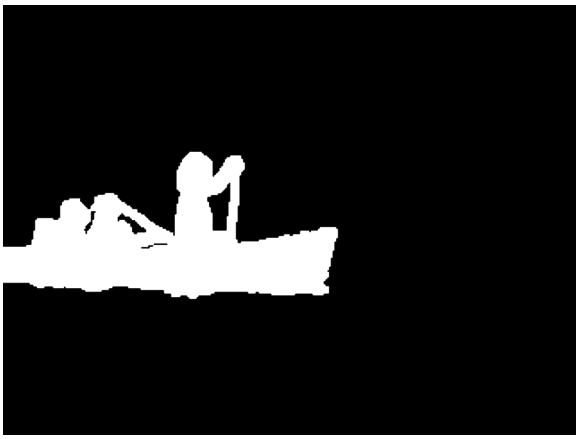
3.4. Datasets for Trustworthy Comparison

For the fair and equal comparison of different approaches on the BS domain space, several datasets have emerged. Those datasets contain pixel-wise annotations of moving objects as well as the original RGB video footage.
Datasets in the domain of machine learning are basically a collection of similar data, they can have almost any structural form and come in a wide variety of prices and licences. For the computer vision space, most datasets contain either videos or pictures and some additional annotations, depending on the use case. Semantic segmentaion datasets often contain pairs of images, one original RGB image, and one image where every object is completely coloured in a colour specific to its class.
For change detection datasets, this annotation gets even easier as there are only two classes. Thus, the annotated image contains either black RGB(0,0,0) or white pixels RGB(255,255,255), where the black pixel stands for "unchanged since the last image" and the white pixel for "changed since the last image". Of course, these images should be sorted on a time axis to preserve the temporal information. One example image pair from the change detection 2014 [2] dataset can be seen in 3.5.

However, only one public dataset exists that annotates only changed objects. This dataset, change detection 2014 [2], provides raw images with corresponding pixel-wise ground truth labels. The ground truth consists of 3 kind of pixels

1. Background (Pixel value of 0)

(a) Frame 93 normal image



(b) Frame 93 ground truth

Figure 3.5.: Frame pair of the "Canoe" video from the CD2014 dataset [2]

2. Edges (Pixel value of 0.5)
3. Moving Foreground Objects (Pixel value of 1.0)

Similar datasets like the the Stanford Drone Dataset [22] and the BMS Dataset [23] provide bounding boxes or pixel-wise annotation for every non-background object image. Although these datasets cannot be used for training or evaluation as they lack the pixel-wise annotation for only **changed** objects, they can still be useful for a more qualitative assessment of the resulting system under different conditions and assumptions.
None of the currently publicly available datasets contain change annotated footage recorded from a moving platform. Although CD2014 [2] does contain some videos in the "Jitter" categories, videos in this category have very limited linear sideways or horizontal jitter. From this lack of datasets, it is clear a synthetic dataset needs to be created.

4. Data Generation

As discussed in chapter 3, the lack of suitable datasets for the specific topic of change detection from UAV cameras necessitates either the generation and annotation of a real world recorded dataset or the creation of a synthetic dataset. As the recording and especially the annotation of a real flight video can take hours or even longer for a few frames, combined with the limited time scope of this project, the decision to generate a synthetic dataset was very obvious. This approach comes with a several advantages, such as precise control of the annotations, control over enviroments and easy addition of new objects.

4.1. Synthetic Data Generation

A game engine was used for synthetic data generation.

(a) Exerpt from RV4 video 11_48.avi

(b) Exerpt from RV4 video 11_250.avi

Figure 4.1.: Examples of the dataset generated in RV4

4.1.1. Real Virtuality 4 Engine

The bulk of the training data was generated in the game engine Real Virtuality 4. RV4 is known for its usage in the popular simulation game Arma 3 as well as the simulation software VBS, both of which are produced by Bohemia Interactive. The

Arma 3 game, that was used for this project, comes shipped with an integrated 3D editor that supports a variaty of possibilities to create ones own scenarios and programmes. It supports a custom scripting language called Status Qou Functions (SQF). Together with the editor and the simple scripting language, it is possible even for beginners to quickly create realistic scenarios for a variety of cases. For this research project, a rudimentary framework was created that procedurally creates scenarios in which a random number of vehicles and humans move to different endpoints. A simulated camera similar to a circulating UAV is used to generate video files with matching ground truth data.

Environment

As Arma 3 comes with three different and huge maps Altis (270km^2), Stratis (20km^2) and Malden (100km^2) the choice of suitable background locations was difficult. After evaluation of the three areas, Stratis was chossen for its volcanic terrain as well as the dense pine forrests paired with thick scrubland plains. Twenty manually selected areas of interest were then chosen on the mainland of Straits. Most of them are in a scrubland or woodland environment, but two points in more urban areas.
Then for each iteration of the image generation process, one area was selected. After that, a random number (10 to 50) of NPCs and vehicles (10 to 25) were spawned in an area surrounding the POI.
All assets used were either stock assets or additionally bought as DLCs.
All spawned objects then receive a move order to the other side of the POI area beyond the FOV of the camera. Luckily, all assets already have a simple AI included that could handle movement at different stances and speeds. Also some basic functions like the spawing of objects in a safe spot surrounding a point are already implemented by RV4.

Moving Objects To diversify the movement and make the model more robust, all objects were given different maximum allowed speeds (0.2 to 25m/s), and the characters were given a random stance (crawling, ducked, walking, running). An overview of the classes of moving vehicles used is given in 4.1.

Weather To limit the scope of the application, only training data with good weather conditions was generated. Parameters for image generation were:

- Overcast 10-30%
- 12:00:00 UTC
- Rain 0%

Type	Different Models
Car	19
Truck	17
Heli	1
Drone	3

Table 4.1.: Different occurring types of vehicles in the self-made dataset

- Lightning 0%

UAV

While objects moved on the ground, a virtual camera, also included in RV4, was spawned 100 meters above the POI. The camera then began to circle around the POI with a radius of 75 meters while recording the moving objects on the ground. This resulted in a UAV-like motion in the recording. This continued until the majority of the objects reached their goal location. Then a new iteration is started, with a new point of interest.

This method results in about 3 image pairs per second. All image pairs include a raw image as well as a ground truth with annotated objects that have moved since the last frame. For the training set used in the final model, 83571 image pairs were created, totalling over 200 gigabytes of .png images. The complete generation time for this final training set was around 8 hours. All images were generated with a resolution of 1920*1200px in RGB space.

Aircraft ego-motion As the designated area of operations is onboard a flying aircraft and the system needs to be able to compensate for the ego-motion of the aircraft, the circling motion was chosen. While in the first step the gimbal target was locked to a specific geo-position, in later iterations a small linear motion was introduced for the gimbal movement. One bias introduced here is, of course, the constant speed and perfect stabilisation of the aircraft. As there is no flight dynamic simulation used. Later, an attempt will be made to supplement this with data augmentation techniques.

Labeling

One of the initial challenges in generating data with RV4 was the correct ground truth labelling of images, as there was no easy way of interacting with the engine to segment objects, a workaround was implemented. For correct labelling, first a

'raw' screenshot was taken. Then the texture of every object that currently had a combined (X + Y + Z) velocity of > 0.2m/s was changed to a bright pink. After the changed texture was applied, another screenshot 'gt' was taken. The locations of the objects are still identical, only the colour of the moving object has been changed. After that, one frame (1/30s) in-game is simulated, the texture is changed back to the original and the procedure is repeated. In postprocessing, the 'gt' screenshots are then converted to binary grayscale images, where 1 indicates that the pixel was changed within the last frame or 0 the pixel is unchanged. This is achieved by converting the RGB screenshot to HSV colour space and setting a threshold for the colour range of bright pink.. All pixels that are within the range are interpreted as a "1" pixel in the binarised ground truth.
A resulting 'bug' can be seen in 4.2. It was discovered while inspecting the training results of an iteration that all red cars where insufficiently recognised. Cars and humans with different coloured textures were mostly recognised correctly. As it turned out, there was a red car in about 10% of the samples in the training set. But in almost all of these samples, the red car was mislabelled as there was a bug in the texture that prevented the postprocessing from correctly recognising the car's colour as changed. As soon as this bug in the training set was resolved, the recognition of red cars during validating was on pair with the rest of the moving objects.

(a) A red car with the ground truth overlayed (before)

(b) After the applied texture fix

Figure 4.2.: Before and after the texture fix

The Final HSW Ranges for bright Pink are:

```
# CV2 HSV Format [0-180, 0-255, 0-255]
lower = np.array([150, 25, 89])  # -- Lower Boundary --
upper = np.array([165, 255, 255 ])  # -- Upper Boundary --
```

As the generation of the dataset in the RV4 was an iterative process, table 4.2 highlights some prior versions of this dataset. Included are the framecounts for each

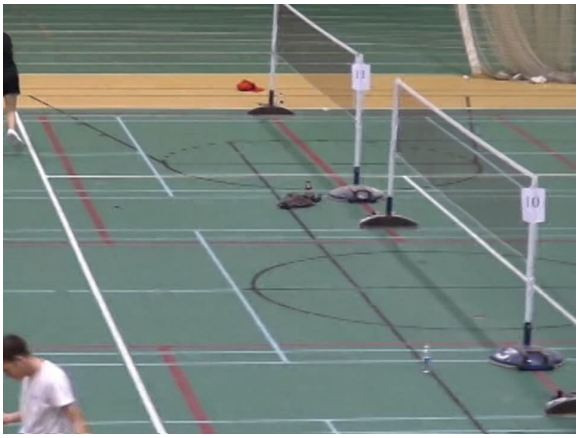
version of the dataset as well as some problems with the dataset that need to be treated. Version 10 fixes all issues previous generations had and is by far the largest of the datasets.

Dataset Version	# frame pairs	Description & Problems
V1	28571	Flightlevel very high, GSD >5cm/px
V2	7857	Error Message blocking screen
V3	4785	Crash of engine
V4	7142	Areas to rural
V5	19295	Vehicles explode
V6	20058	POI partly in jungle
V7	21428	POIs not good yet
V8	38571	Spawn of Vehicles often failed
V9	27857	Annotation not working on red cars
V10	83571	Final dataset for training
V11	11851	Additional benchmark dataset

Table 4.2.: Dataset versions until the final RV4 dataset

4.2. Real Data Acquisition

4.2.1. CD2014 Dataset



(a) Frame 1 from "badminton"

(b) Frame 1 from "office"

Figure 4.3.: Excerpts from the CD2014 Dataset

Released by Goyette et al. [2] in 2014, this dataset includes 53 videos from 11 categories. Figure 4.3 contains examples from two of the included videos, overall the dataset contains approximately 159278 image pairs with a .jpg base image and a corresponding pixel-wise ground truth in .png format. Almost every relevant change detection algorithm has already been evaluated on this dataset and the results can be viewed at `http://jacarini.dinf.usherbrooke.ca/results2014`. Important to note that some algorithms are trained with supervised learning methods on the same data as the evaluation, such as the MFCN [7]. Thus, supervised trained algorithms trained on the same dataset as the evaluation should not be compared to algorithms that were not trained on the same data.

All Categories of the CD-2014 Dataset are listed in 4.3.

The Dataset has been thinned to include only videos where no part of the ground truth is unlabelled (Outside of ROI). This leaves the following videos, which can be seen in 4.4.

Most of the videos included were recorded indoors and thus are not exposed to any weather challenges [E1], and clearly there are mostly good lighting conditions [E2], even tough the thermal videos were not recoded in a traditional wave range. However, the range of motion is quite diverse, from humans walking through the screen to cars driving across a motorway. As mentioned previously, the dataset contains only static camera footage, although the camera jitter videos have a slightly

Video Categories	# frames	Description
Dynamic Background	18871	Water and heavy moving trees
Camera Jitter	6420	heavy Camera Jitter
Intermittent Object Motion	18650	Start and Stop of Moving mid Video
Shadows	16949	Strong presence of Shadows
Thermal	21100	Infrared Videos
Challenging Weather	20900	Outdoor video with difficult Weather
Low FrameRate	9400	Low frame per second count
Night	16609	Videos at night or low lighting conditions
PTZ	8630	Turning camera or start/stop of the turn mid video
Normal	6049	No special setting
Air Turbulence	13700	Mostly caused by heat.

Table 4.3.: Video Categories of CD2014 Dataset

Video Name	# frames	Categorie
badminton.avi	351	Jitter
canoe.avi	390	Dynamic
copyMachine.avi	2901	Intermittent
corridor.avi	4901	Thermal
cubicle.avi	6301	Shadow
diningRoom.avi	3001	Thermal
highway.avi	1231	Baseline
lakeSide.avi	5501	Thermal
office.avi	1481	Normal
park.avi	351	Thermal
pedestrians.avi	800	Baseline
peopleInShade.avi	950	Shadow
PETS2006.avi	901	Normal
traffic.avi	671	Camera

Table 4.4.: Remaining videos of the CD2014 after selection

moving camera [I1].
These videos were used together with their respective ground truth to evaluate models on completely unseen data. The results of this evaluation are reported in the Results chapter.

4.2.2. SBI2015

(a) Frame 1 from HighwayI

(b) Frame 1 from CAVIAR1

Figure 4.4.: Excerpts from the SBI2015 dataset

The Scene Background Initialisation dataset or SBI for short is a dataset released by [24] with the intention to compare traditional approaches of background initialisation algorithms. Nonetheless, we can still use the dataset for evaluating our proposed model.

The dataset consists mainly of static video in different environments, the videos listed in 4.5 are included.

Only the video "Toscana.avi" was removed due to the low frame count.

As this dataset was originaly used for background subtraction, or more precisely to compute the background via an initialisation procedure. There is only one ground truth picture for the background per video. Due to this limitation, we cannot use this dataset for quantitative results as there is no pxiel-wise annotation.
However, some results from a visual analysis are included in the Results chapter.

4.2.3. Self recorded drone flight footage

Additionally, some aerial drone footage was recorded. As drone platform used was a DJI Mavic 2 Zoom equipped with a 4k camera recording at 30fps. The camera can zoom in the range of 24-48mm. The camera sensors are of high quality and are able to achieve a GSD of 0.93px/cm when flying roughly 100 meters above ground.
The recorded videos obviously do not include any ground truth and are purely used to evaluate if the model trained on the simulated data can also work on real world video.
All recorded videos can be downloaded at `www.github.com/TomSeestern/ISEEU-Net`

Video Name	# frames	Description
Board.avi	228	Two people move in front of a blackboard
Candela_m1.10.avi	350	Person in waiting room
CAVIAR1.avi	610	Hallway in a mall
CAVIAR2.avi	460	Hallway in a mall
CaVignal.avi	258	Person moving outside
Foliage.avi	394	Foliage in front of cars
HallAndMonitor.avi	296	Hallway
HighwayI.avi	440	Highway with cars
HighwayII.avi	500	Highway with cars
HumanBody2.avi	740	Humans walking trough a room
IBMtest2.avi	90	Another Hallway
PeopleAndFoliage.avi	341	Synthehic video with artifical foliage (bad Quality)
Snellen.avi	321	Foliage in front of static background
Toscana.avi	6	Broken video

Table 4.5.: Videos of the SBI2015 Dataset

and some examples of the inference of the final model can be seen in figure 4.5.
A list of the recorded videos including a brief description can be seen in table 4.6.

(a) Dog jumps over small stream

(b) Flight over some Fields

Figure 4.5.: Some of the self recorded material

As can be seen in figure 4.5, all the above videos are recorded in good weather conditions [E1] and good lighting conditions [E2] and contain only two types of moving objects, "human" and "dog". Also, the recording drone system almost always has no ego-motion and no sensor motion [I1].

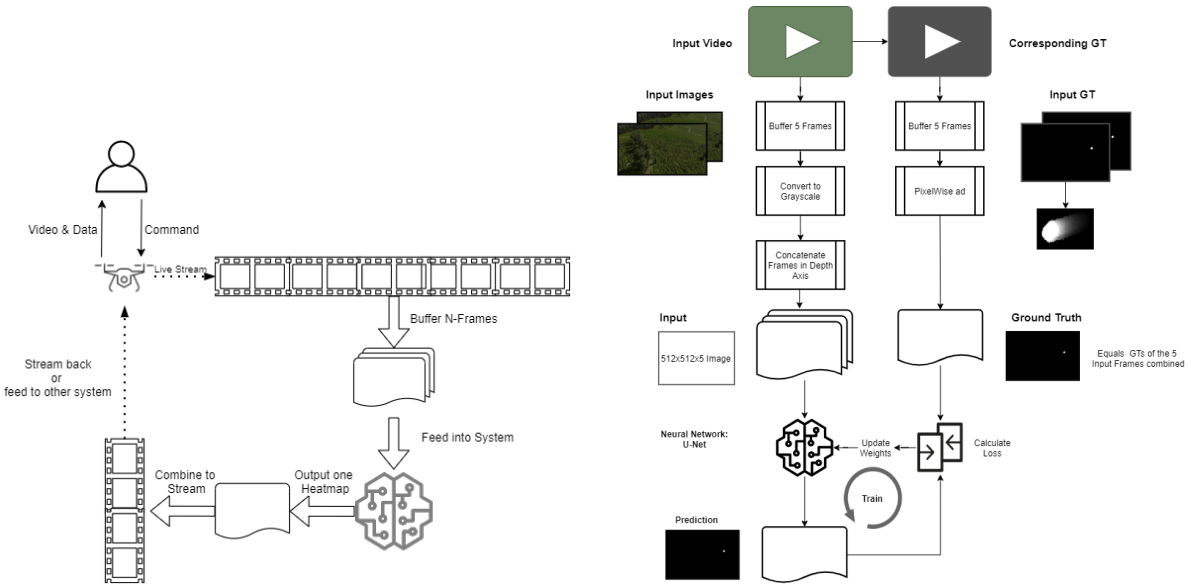
Video Name	# frames	Description
WalkingOnRoad	669	Walking on an Asphalt road viewed from above.
CatchAndRunning	504	Dog running after a ball
CatchFromAbove	502	Dog running after ball
DogSearching	904	Drone moving backwards dog is in view.
ClimbAndTilt	2732	Drone climbing and tilting camera
RunningPassover	1080	Fly over dog while he is running
Passover	207	Flyover Dog and Humans while everybody standing
DogOverRiver	134	Dogs jumps over river while drone hovering

Table 4.6.: Self recorded drone videos of the dog "Portos"

5. Methods and Implementation

This chapter explains the methods and system used, which together result in the reliable change detection system: "ISEEU-Net".

5.1. System Design



(a) System loop while integrated

(b) System loop while training

Figure 5.1.: High-Level system overview

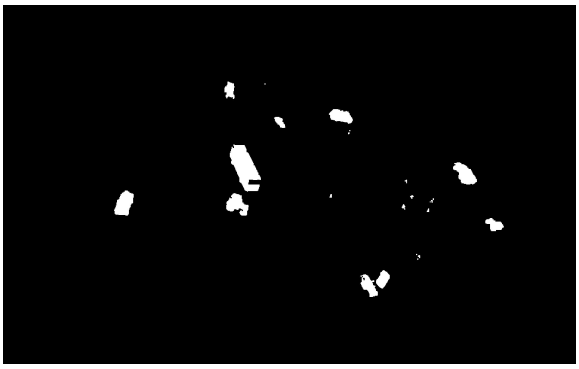
As defined in section 1.2, in the system requirements, the system takes an RGB video stream of arbitrary length, with a variable but at a fixed resolution, and creates a secondary video stream based on this stream.
The figures in 5.1a illustrate the working principle of the system. The recieved data stream is buffered and fed into the ISEEU-Net.
The ISEEU-Net infers the buffered frame stack and produces a heatmap with the same resolution as the inputed frame stack. Each pixel on the heatmap describes the certainty of the system that this pixel was changed in the current step.

The combined heatmap is then recombined into a stream and is ready for further use, either streaming back to an operator or as input for other systems such as an identification and tracking algorithm.
As discussed in the related works section, neural networks, and more specifically convolutive neural networks, are currently the most reliable methods of detecting change. Although, as seen in `http://changedetection.net/`, supervised methods prevail, for this use case with a system that cannot be initialised and is constantly moving, another method than training a CNN on part of the evaluation data must be used.
As the system needs to be able to function without being initialised, the objective was clearly that the network should be pre-trained. Moreover, to accommodate the many possible types of landscapes, climate zones, and vegetation, and the lack of a suitable dataset, the decision was made to generate a synthetic dataset. This dataset is then used to train the network to reliably detect change.
As input for the Neural Network (NN), a preliminary number of five frames was chosen, which proved to be controversial for the following reasons: In theory, a reference to a "change" should be sufficient to calculate the difference between them. However, when every frame is fed into the system together with the previous frame, the latency would be huge as there would be an iteration of the system for every frame received. This would certainly limit the real-time capabilities of the system. If the gap between the frame pairs is widened to skip every n-th frame, the risk of "missing" a small motion increases. As it is, the system was trained with a framestack of 5 input frames. See the Outlook chapter for more work with ideas on how this can be changed.
After buffering the framestack from the livestream, the data is prepared to be fed into the system. This includes resizing, normalising, and converting to grayscale.
When the framestack is fed through the system, the result is a frame in binary format indicating which pixel has changed in the framestack, with a value between zero and one indicating the certainty that this pixel has changed.
We will look at all those steps in detail in this chapter.

5.2. Prepossessing Data

All data fed into the system, especially training data during development, is pre-processed before entering the system. The difference between pre-processing the data during training of the system and later pre-processing the data during use of the system is simply that ground truth is also pre-processed during training and some data augmentation techniques are applied during training.

(a) One original input frame



(b) Corresponding Ground truth

Figure 5.2.: Example training pair

5.2.1. Segmentation Video to common format

First, all training examples have been converted into a common format so that they can be used together. For this purpose, all training videos were resized to the common video format .avi and ground truth was binarised (1 or 0 -> pixel changed or not). The binarisation of the ground truth comes with several drawbacks, like the loss of context for shadows and the missing distinguishing between an object background and the fuzzy pixels in between. However, as we are only interested in the general area where the change ocurred, the assumption is that this information is sufficient. For the dataset generation, video pairs were generated with matching frame counts and time frames. See figure 5.2 for an example pair.

As seen in the example 5.7, for large objects such as trucks with trailers, the entire truck is annotated as "changed" even though large parts of the solid-coloured trailer have moved, but there is only a minimal change in pixel value. For the purposes of the system, we still want the whole truck to be annotated, even though this makes the prediction much more difficult as the system not only has to learn what the change is, but also learn to extrapolate the annotation to a full object.

5.2.2. Grayscale Transformation

If the original raw image of the Input frames are RGB they are converted to grayscale using the following formula (CCIR 601): $grayPixel = 0.299R + 0.587G + 0.114B$ This is done to eliminate the chance of colour becoming a main reason why the system detects a change. In addition, this was quite useful to eliminate two out of three dimensions for input, making the system more resource friendly.

Figure 5.3.: t=0

Figure 5.4.: t=1

Figure 5.5.: changed pixels

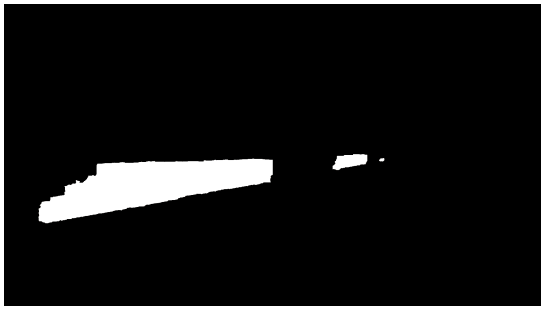


Figure 5.6.: Groundtruth (t=1)

Figure 5.7.: Changed Pixels over time in video from [25]

5.2.3. Thresholding and Binarization

As mentioned in chapter 4, datasets like CD2014 do not include binary ground truth, but also edges with a float value of 0.5, thus a threshold of 0.5 is applied to every ground truth frame. Essentially, this creats a binary heat map for every pixel in the frame. A value of 255 (or 1 for float range) thus means there was change between the last and the current frame, and a value of 0 means there was no change.

5.2.4. Data Selection

In the training data, only framestacks with more than 1% change were allowed. For this, every framestack was evaluated and the values of the grayscale pixels were summed up. This sum must then be greater than 20736 (1920*1080*1%). Stacks where the sum was below that threshold were discarded. This ensured the dataset contained only "valuable" examples where the actual change was happening. This proved to be very valuable, especially with the constrains on available RAM on the training machines. It resulted in an increase in the information density of the dataset and a corresponding decrease in training time.

5.2.5. Data Augmentation

Data augmentation has been proven[26] to drastically improve performance of neural networks as well as robustness. Thus, for this project, the following augmentation techniques have been applied.

General Robustness

[27] made a broad comparison on different methodologies for adding noise to images and the performance improvements that these augmentations bring. As a result of this Gaussian noise, N(0–0.001) is randomly added to each pixel.

Brightness variation

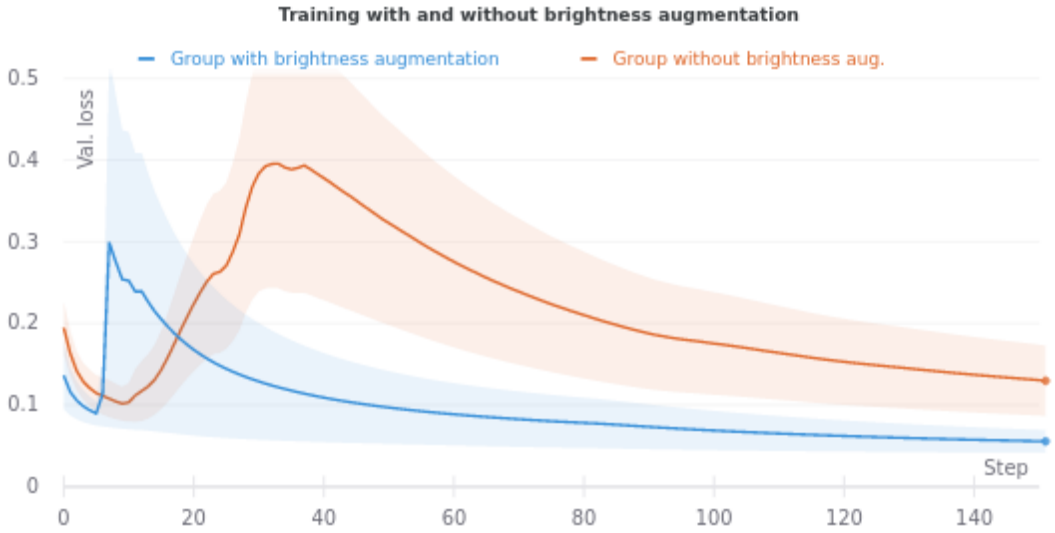


Figure 5.8.: Training with and without Brightness Augmentation

Since [18] uses offset/addition to RGB values to simulate the global change in illumination, which in turn increases robustness to illumination changes and bias in sensor data, a brightness offset for each frame stack was added to the training data Each frame in a framestack was given a brightness offset in range [-35%,35%]. This resulted in a general robustness to different illumination levels. Even rapid illumination changes in the framestack can now be compensated. A comparison of validation losses with and without brightness augmentation can be seen in figure 5.8. As can be seen in the figure, the validation loss is significantly lower with the brightness augmentation, indicating a better generalisation of the model for different illumination enviroments and obviously associated with a lower error rate.

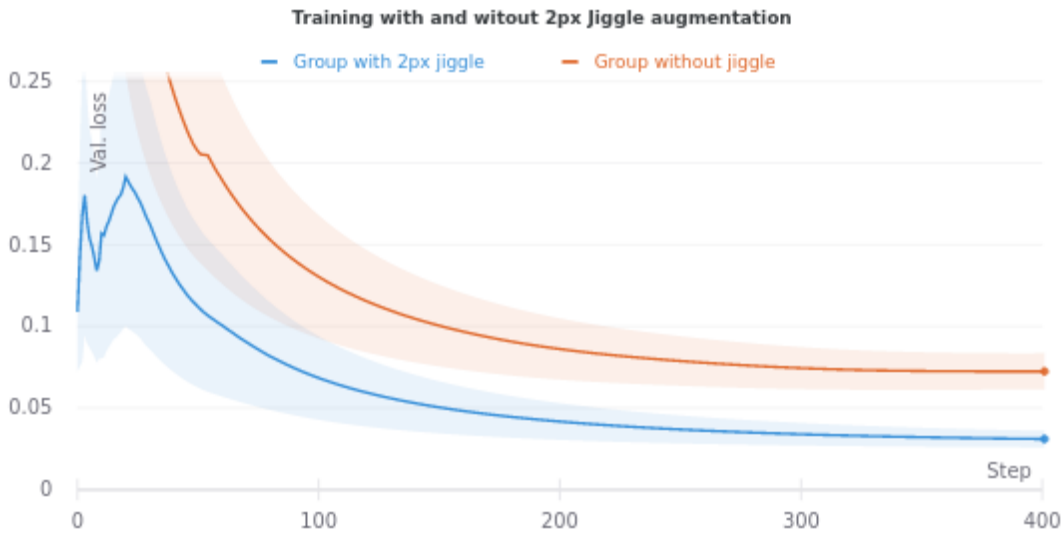


Figure 5.9.: Training with and without a 2px cropping aka. Jiggle

Jitter

To compensate for unstabilised gimbals, the training data was augmented with a random crop-off of a maximum of 3px for each border. This resulted in a jitter-like motion. Sadly, this jitter-like motion drastically reduced the scores on all benchmarks by a significant margin as can be seen in figure 5.9. It is speculated that when the jitter is bigger than the receptive field of den CNNs, it results in problems with the CNN to accurately match the pixels to create a spatial and temporal connection.

5.3. Training's Dataset Generation

From the now standardised files, a dataset was generated. For this purpose, all designated videos were combined into one large dataset. For every 5 frames, the input frames were concatenated in the depth axis and the 5 ground truth frames were summed up. This results in a new subset of samples with the dimension:

```
samples x height x width x 5  for the Input Dataset
samples x height x width x 1  for the ground truth dataset
```

For ease of storage, the data is then packed using the python libary Pickle [28]. Pickle serialises the data held in numpy 4D arrays into one big file for easy storage and transfer. This serialised data is easily deserialised later and is then directly ready for use in training the neural network.
For the training, 5 frames were selected with a 5 frame "step" so that no frame is

present more than once in the dataset. It is however possible to swap this for a rolling window model. This of course increases computing time by a factor of 4, but also minimises the chance to miss a change between two full steps. For this project, the first approach were the stepsize equals the framestack size, was chosen due to memory and storage constrains.

5.4. Model Architecture

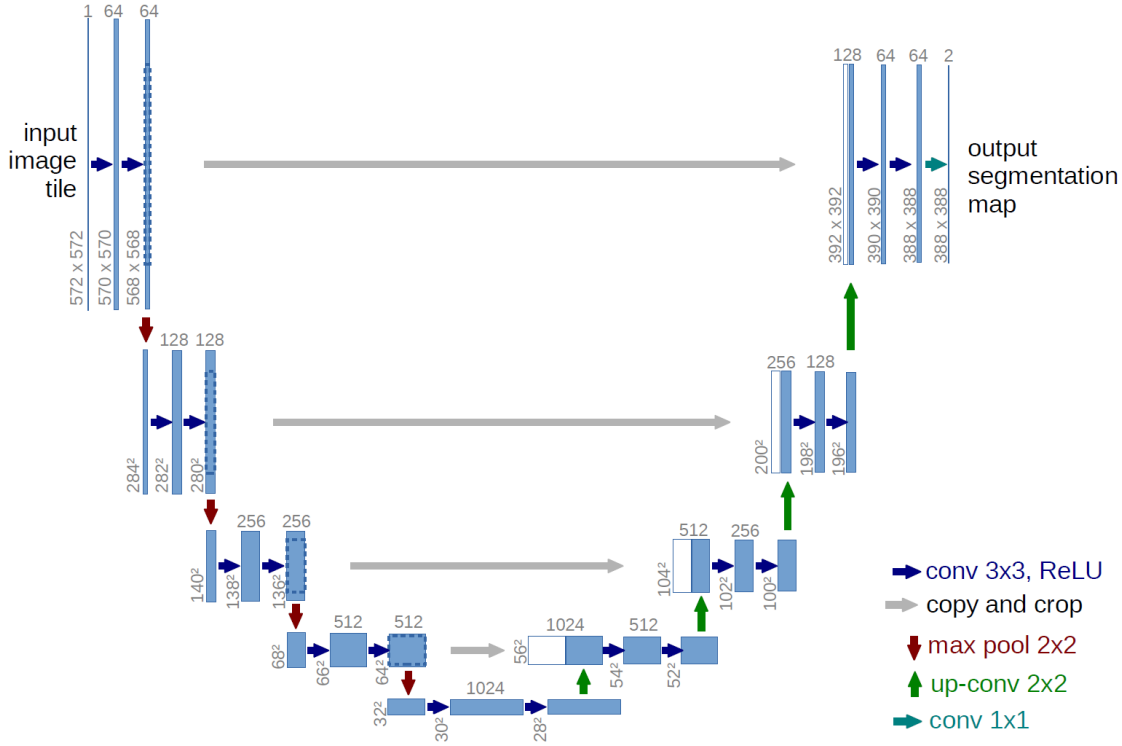


Figure 5.10.: The U-Net Architecture from [19]

Tezcan et al. showed, as previously mentionend, in their work [18] that the U-Net architecture is capable of reliably detecting changes despite the caviates mentioned above. Since the architecture has otherwise proven [18] to be reliable and relatively lean in computational overheads, it was selected for this project.
The U-Net architecture proposed in [18] can be seen in figure 5.10. It is based on an initial convolution of the framestack downwards and then a series of deconvolution layers back to the original image size. It contains about 31,042,434 trainable parameters. However, unlike the reference frame used by [18] in their work, no reference frame is used in ISEEU-Net. This choice was made simply because of the lack of a

"clean" or "true" reference frame in the domain of moving aerial video footage. The fact that the aircraft is constantly moving makes the creation of a true reference frame impossible, as the crop of image that is recorded constantly changes.
Although there are strategies to adapt the reference frame over time, this is not suitable for our case as we can never say when would be the right time to choose the reference frame.
For example, if the reference frame is simply the first frame at the start of the programme, there may be many moving objects that are currently in that frame. These objects or the wider background behind them would be annotated as long as the reference frames remained unchanged.
Therefore, for this project, it was chosen to use a "framestack" as input. The currently received frame + the last 4 frames concatenated in the depth/channel axis are fed into the architecture. This enables the system to work without any reference frame or external meta-data.
The drawbacks of this approach are, of course, that there is no knowledge from outside the 5 frames, as there are no memory modules or other input data. The system has no access to other information such as the camera metadata or previously moved objects.

5.4.1. Changes to the Original U-Net Architecture

Inspired by the study of Garbin et al.[29] on the impacts of dropout and batch normalisation layers in Convolutional Neural Networks, the original U-Net architecture, as seen in figure 5.11, was modified to replace the dropout layers with a normalisation layer after each (de-)convolution. This resulted in much quicker convergence during training and a slightly improved final loss, as can be seen in 5.11. The detailed table with an in-depth comparison can be found in Figure A.

5.5. Hyperparameter Space

Model hyperparameters govern the entire training process. They dictate how aggressively the model learns, how long it runs and many other things. Hyperparameters are high level and can be observed over multiple training configurations. From this, conclusions can be drawn about how a specific parameter affects the model's overall performance. In figure 5.12, the used parameters can be seen. A high importance means that the parameter has a big impact on the model performance. The correlation indicates if the parameter size affected the performance positively or negatively. For example in 5.12, the epoch row indicates that a bigger epoch number correlates positively with better results. A bigger batch size, on the other hand, correlated with

lower performance.

While training different combinations of hyperparameters, the following parameters were changed:

- epochs - How many epochs to train maximum, early stopping was also used.
- batch_size - How many training samples must be processed before the model's internal parameters are updated.
- learn_rate - The Learn rate of the used Adam[30] optimizer.

Early stopping

Early stopping is a technique used in machine learning to avoid overfitting on a training dataset. It is one of the oldest methods for regularising a neural network.

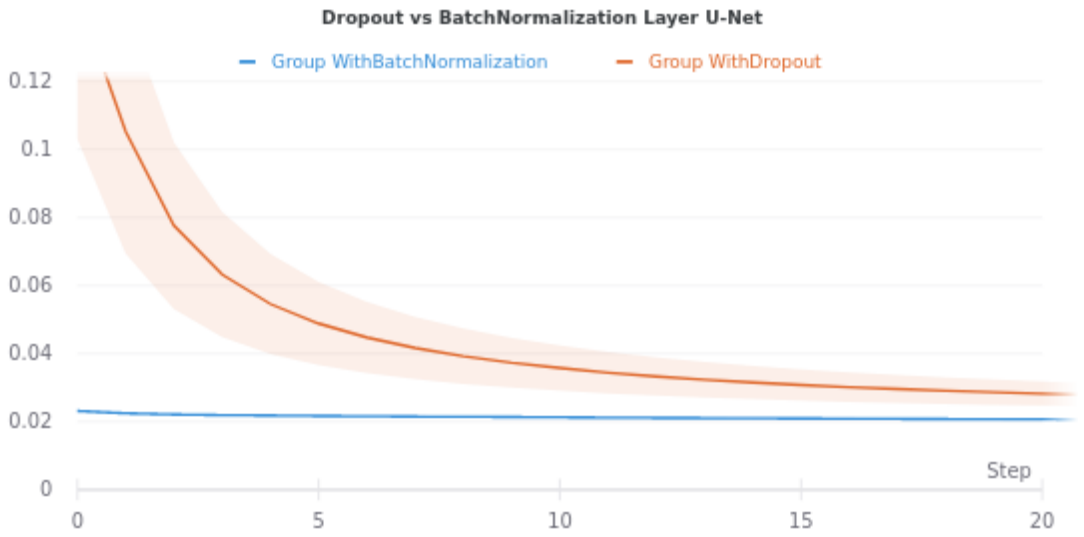


Figure 5.11.: Training process with Batch Normalization and with Dropout Layers

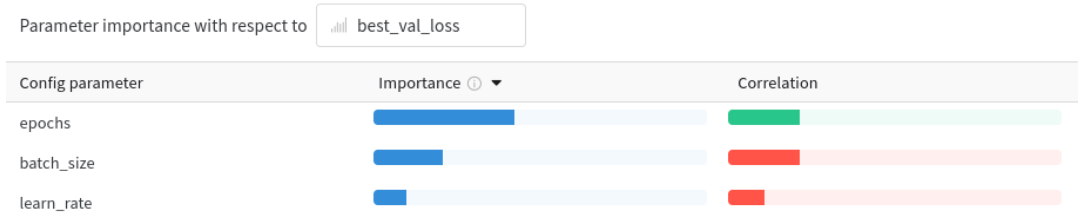


Figure 5.12.: Tested Hyperparameters with their Calculated Importance and Correlation

In principle, it works by monitoring the training loss and the validation loss of the model during training. If the validation loss starts to increase while the training loss decreases, the model is assumed to be overfitted on the training data, and thus the early stopping is triggerd.
For this work, early stopping was implemented with the Hyperband Method, see [31] for a detailed explanation of how this method work.s

5.6. Training Enviroment

Data

As previously stated, due to the lack of suitable real world training data, only the synthetically generated training data from RV 4 was used. See chapter 4 for a full description of the dataset. The dataset was limited to 6000 framestack and ground truth pairs, all with a resolution of 512x512px. Before training, the first 80% of the dataset are assigned as training set, the next 15% as the validation set, and the last 5% as the test set.

Environment

The U-Net[18] architecture is implemented in the Keras [32] using TensorFlow [33] as the back end. The source code for the architecture and the training can be found here `www.github.com/TomSeestern/ISEEU-Net`. Python 3.6 was used as the primary language for writing the training scripts, although some preprossessing scripts are Bash scripts and most of the TensorFlow back end is in either C or CUDA Code.
For the complete program code as well as the specific library version, visit `www.github.com/TomSeestern/ISEEU-Net`. The network was mainly trained on Amazon Web Service (AWS) EC2 instances. Specifically, on a "g4dn.xlarge" instance with 4 vCPUs, 16GB RAM and one NVIDIA T4 (16GB GDDR5) GPU. Resulting training times average 2.2 steps per second at a resolution of 512x512px and a batch size of 4. This results in a training time of 2 Days (n=6000, epochs = 250).
The development was done with Pycharm Professional. All experiments, i.e. different hyperparameter combinations, were monitored using Wandb [34].
The Nvidia JetsonNX was also used for speed evaluation, it is introduced in section 7.5. On this board, the model was run with a C++ compatible API integrated in tensorflow [33], so that the system can also be called from other Languages.

6. Result

In this chapter we evaluate how the used method performed on the previously introduced datasets.
As there are many different configurations of models and hyperparameters throughout the project, one final configuration had to be selected. The training run with the codename "bknulmgz" was selected due to its best performance on unseen synthetic data. (Validation Split)
It was trained exclusively on the synthetic dataset and with the parameters listed in 6.1. These parameters were discovered while performing Bayesian hyperparameter optimisation.

The Metrics seen in table 6.1 resulted in a F1 score of 0.7027 on the validation split of the training data.

As seen in 6.1 the final selected model "bknulmgz" performs reasonably on most benchmark videos from the CD2014 dataset [2]. The dataset contains only human subjects in high altitude aerial footage. This results in poor performance as these human figures are generally worse annotated in the training set. Nonetheless, most of the occurring persons are at least partially annotated.

Name	Value
epochs	500
batch size	16
learn rate	0.00005868

Metric	Value
runtime	74553s
epochs	230
val_TP	183330
val_FN	843684
val_FP	709202
val_TN	326915232
val_loss	0.0152
val_mean_io_u	0.4959
val_absolute_error	0.0063

Table 6.1.: Parameters and results for training "bknulmgz"

As Zeng et al. noted in [7], an F-Measure6.6 above 0.94 and a PWC 6.5 value below 0.9 suggest that the result is almost pixel perfect. One or two wrongly placed pixels already have an F-Measure below these thresholds and do not really matter in a real case.

6.1. Metrics

To compare results quantitatively, seven performance metrics from [2] were used: Probability of correctly classifying a positive pixel as positive.

$$\text{Recall} = \frac{tp}{tp + fn} \tag{6.1}$$

Percentage of positives correctly classified out of all positives.

$$\text{Precision} = \frac{tp}{tp + fp} \tag{6.2}$$

Percentage of objects incorrectly classified as positive that are actually negative.

$$\text{FalsePositiveRate(FPR)} = \frac{\text{FP}}{\text{N}} = \frac{\text{FP}}{\text{FP} + \text{TN}} = 1 - \text{TNR} \tag{6.3}$$

Percentage of positive objects misclassified as negative out of the total number of positive objects.

$$\text{FalseNegativeRate(FNR)} = \frac{\text{FN}}{\text{P}} = \frac{\text{FN}}{\text{FN} + \text{TP}} = 1 - \text{TPR} \tag{6.4}$$

PWC weights false categorizations against the total number of occurrences. The lower the PWC score, the better. A low score means fewer false detections. See 6.5 for a detailed explanation.

$$PWC = \frac{100 * (FP + FN)}{TP + FP + TN + FN} \tag{6.5}$$

The harmonic mean is used to combine precision and recall to get the F1 score. The score has a range from 0 to 1, where 1 means a perfect match with the ground truth.

$$F - Measure = \frac{2 * precision * recall}{precision + recall} \tag{6.6}$$

In [35],it is argued that MCC [36] is more suitable for unbalanced class classification problems than the often used F1 score, since true negatives are also considered. (check

[20] page 7 for details). In general, the closer the F1 value is to 1, the better.

MCC or Matthes correlation coefficient is equal to the Pearson/Boas-Yule phi coefficient.

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FN)(FN + FP)}} \tag{6.7}$$

The Range is -1 to 1, where 1 is complete agreement with the ground truth.

6.2. Evaluation on the RV4 dataset

Figure 6.1.: Colourmap for overlay of predictions over Video

In this section we will evaluate the finalised ISEEU-Net system on our synthetic datasets.
To illustrate the results, the resulting prediction map with a colormap applied to it is overlayed on the input video . This results in a colour gradient from red to white around moving objects. Bright white means that with 100 percent certainty there has been a change between the last frame and this one. Yellow is around 75 percent and red below 50 percent certainty, see 6.1 for the colour range.

As seen in table 6.2 the performance of the selected models on the synthetic benchmark is very mixed. Especially on the video "11_48.avi", which consists of video generated in RV4. The lack of performance is easily explained by the absence of vehicles in the video "11_48.avi". The training dataset contains a lot of cars that are on average ten times bigger internally when viewed from above. This results in an imbalance towards a more reliable detection of cars, especially since they are just bigger.
6.2 contains examples of both "11_48", which contains only humans in a more urban setting, and "11_1", which contains cars and humans in a more vegetated environment. Especially noteworthy are the predictions in 6.2a where objects are detected even near tree tops.

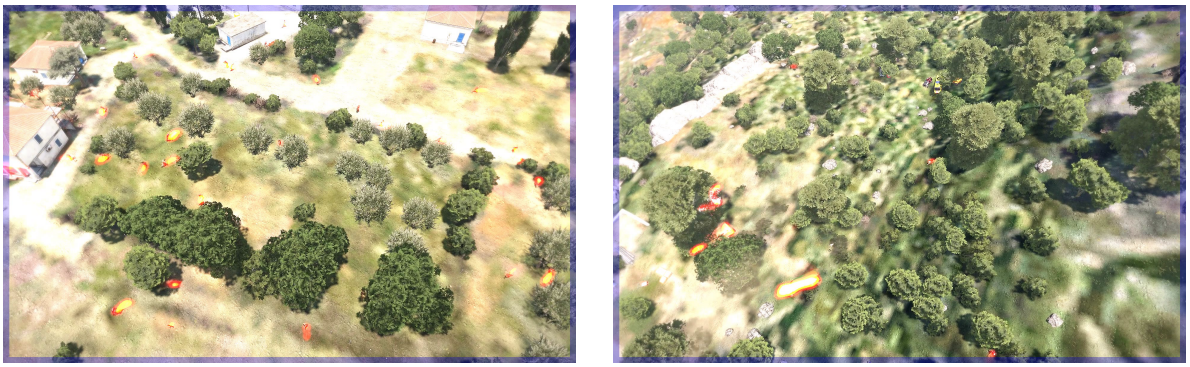
6.3. Evaluation on CD2014 dataset

The change detection dataset 2014 introduced earlier was used for evaluation. The performance on this dataset gives a clear indication that the transfer of a model

trained on synthetic data can also be used in real world scenarios, although with some decrease in performance.
The average F1 score across all CD 2014 videos used is *0.86104*, which is on par with the top-ranked methods on `www.changedetecion.net` that do not perform supervised learning on any part of the validation data. A small summary of performance on the CD2014 benchmark can be found in 6.3, a full report is available in Table A.
Figure6.2 includes some examples with overlaied change prediction map. Especially 6.3b is a good example to highlight the difficulty of correctly predicting shadows and accurately segmenting changed shadows versus static shadows.

6.4. Other Visual Examples

As the SBI 2015 dataset as well as the self-recorded flight video do not contain any ground truth, they can only be used as visual examples. As shown in fig. 6.3b, the predictions for cars on roads work reasonably well. However, the guard rail is also partially detected. The resulting videos can be found in the attachments and on `https://github.com/TomSeestern/ISEEU-Net` for the resulting videos.



(a) RV4 Video 11-48 with overlayed prediction

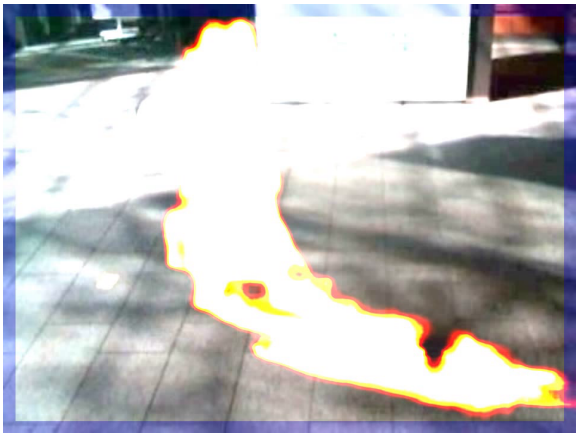
(b) RV4 Video 11-1 with overlayed prediction

Figure 6.2.: Clips from the inferenced RV4 videos overlaid with the predictions.

videoName	PWC	MCC	F1-Score
11_150.avi	0.401	**0.751**	**0.744**
11_250.avi	1.195	0.688	0.677
11_50.avi	**0.323**	0.615	0.607
11_48.avi	0.157	0.118	0.111

Table 6.2.: Excerpt from the evaluation on part of the RV4 Dataset

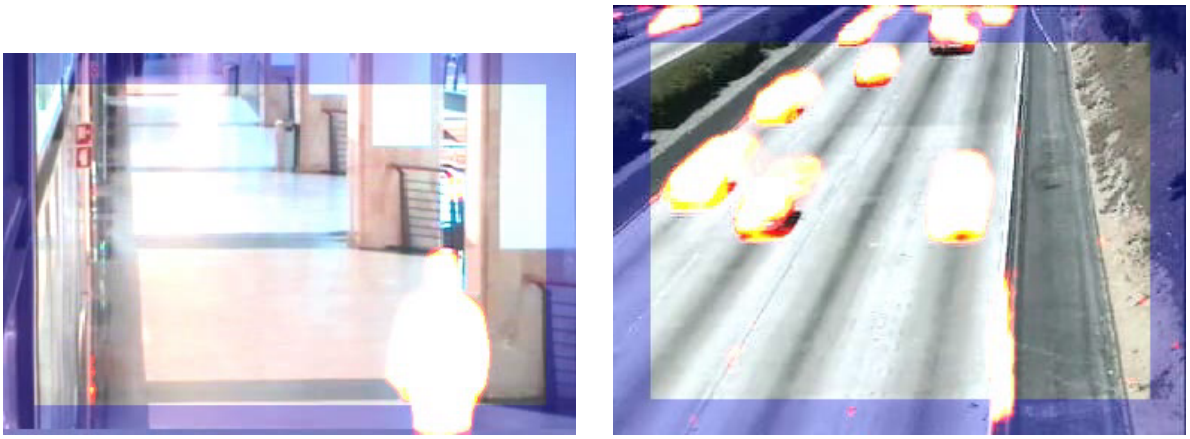
(a) CD2014 Video "Office" with detecion Overlay



(b) CD2014 Video "people in shape" with detecion Overla

videoName	PWC	MCC	F1-Score
PETS2006.avi	0.741	0.832	0.826
badminton.avi	2.323	0.787	0.789
canoe.avi	1.429	0.874	0.875
copyMachine.avi	1.144	0.926	0.930
corridor.avi	0.823	0.900	0.905
cubicle.avi	**0.637**	0.884	0.886
diningRoom.avi	1.973	0.887	0.896
highway.avi	2.511	0.842	0.854
lakeSide.avi	0.683	0.888	0.890
office.avi	0.753	**0.953**	**0.957**
park.avi	1.955	0.618	0.617
pedestrians.avi	1.239	0.744	0.745
peopleInShade.avi	3.432	0.788	0.792
traffic.avi	5.662	0.707	0.713

Table 6.3.: Excerpt from the Benchmark on CD2014



(a) SBMI Video "CAVIAR1" with detecion Overlay

(b) SBMI Video "HighwayII" with detection Overlay

Figure 6.3.: Clips from the inferenced videos overlaid with the predictions.

6.5. Evaluation on self recorded Drone Footage

To test the hypothesis that a neural network that is trained solely on synthetic data can also perform on unseen real world flight footage, the system was applied to self-recorded drone footage from the previously introduced drone system.
As there is no ground truth data available for this data, we can only evaluate it visually.
The examples given in figure 6.4 include the overlayed change detection.
As seen in 6.4b, the system actually works on real world data! It annotates the moving dog with a confidence that a human observer would consider sufficient, and even works when the recording drone is slightly moving. The model even ignores the moving trees and bushes that are shaking slightly in the wind. One thing to note, however, is that slight motion, such as slight turning of a body, are not yet detected. This most likely is also a direct result of the downscaling of the image and the grayscale conversion, as a lot of information is lost there.
Nevertheless, if the movement is significant enough, such as a step forward, the movement is be detected.

(a) Video "11_CatchAndRunning" (b) Video "19_DogOverRiver"

Figure 6.4.: Self recorded Videos with detecion Overlay

7. Discussion of Results

In this chapter, different aspects of the results are looked at.

7.1. Sizing and Change Intensity

One of the first limitations we set in chapter section 2.1.2 is the size of the object. For that purpose, we referenced the minimum ground sampling distance in section 2.1.2 and used it as guideline for our minimum size. This resulted in the minimum GSD of 1cm/px to detect an object 10cm in diameter. Figure 7.1 illustrates this hypothesis with a unused part of the RV4 Dataset. There we can see that even a person with only a 50px diameter is reliably detected, despite having a resolution three times lower than set in the limitations [I1.2].

Figure 7.1.: Zoom on a 1.8m person, equaling 50px and a GSD of around 3.6cm/px

7.2. Shadows

One difficult topic that was underestimated at the beginning of the project was shadows. In the RV4 dataset, shadows of objects are not annotated, which leads to a bias towards dark objects in the model. This phenomenon can be observed quite well in 4.4a and 6.3b.
In 4.4a as well as the other examples containing cars, there is a slight bias not to

detect dark or black cars. This results in later detection or incomplete detecion when the segmentation between object and object shadow fails. On the contrary, the system detects shadows such as in 6.3b, even though it was trained not to detect them. These false positive detections result in preference degradation as the predictions for the detected shadow are very unstable and result in a flicker effect.
This can most likely be resolved by adding more dark colour vehicles to the training dataset and creating scenes where more shadows of objects occur.

7.3. Environmental comparisons: Land vs Suburb

6.2b illustrates pretty well that the ISEEU-Net can handle even densely planted areas with large obstacles in the form of trees and bushes. In 6.2a the suveyed area includes a rather rural suburb with some buildings. Here we can cleary see that some false positives occur around some buildings. This indicates that the system will most likely have a higher false positive rate in a more suburb environment. This assumption makes sense as the training set only included scenes from a scrubland environment. Nonetheless, videos like 6.3b show that vegetated areas work even in the real world. Although videos like this should make it harder to detect changes as the vegetation shifts, the system works better there. Perhaps the ISEEU-Net has only learned how to detect straight edges, such as from a car, and so has difficulty with straigt corners of buildings.

7.4. Special case: Water

One Special case that was explicitly excluded in the limitations chapter was regions with water as input to the ISEEU-Net. In CD2014 [2] there is one video that contains mostly water, it is the "canoe" video. In 7.2 an example from the video is shown, overlaid with the predicted change. As you can see, the prediction works with only small false positives. Thus, it can be concluded that, at least from this perspective, water areas represent a small challenge to the system.

7.5. Model Performance Comparison (CPU / GPU / TPU)

The tests on the hardware specified in section 5.6 show the performance metrics of the system while running with live stream data. See 7.1 for an overview. The NVIDIA Jetson NX was used as a test environment for a possible mobile platform.

As the Jetson NX comes with 384 CUDA compute units, 48 Tensor compute units and 6 ARM CPUs @ 1.2 GHz, it is more than capable to handle the load provided by the system. A even greater performance increase is achievable when the model is converted to the TensorRT Standard. TensorRT is an SDK from NVIDIA to optimise deep learning applications on NVIDIA Hardware. However, due to time constrains, the conversion to TensorRT was not included in this work.
The performance values in 7.1 include pre- and post-processing steps described in chapter 4.

Name	FPS
CPU only	5.28
CPU + GPU	24.94
Jetson NX	3.23

Table 7.1.: Framestacks that can be handled per Second while inference.

Figure 7.2.: CD2014 Video "Canoe" with overlayed change detecion

8. Summary

8.1. Conclusion

This work showed that it is possible to train a neural network on purely syntethic data and still get reliable results on real data.Current methods were shown to give reliable results and one machine learning method was used to create a neural network capable of detecting changes from aerial camera recordings. Next, there was a complete walkthrough of the process for this machine learning project.
Starting with the available datasets and how synthetic ones were generated, to the steps that need to be followed to make them suitable for training. Other datasets like the self-recorded drone footage were also explored.
After training, the model was examined in terms of some optimisation techniques and finally the results were analysed. Our work is comparable to current methods for change detection, with the added benefit that it does not require initialisation annotation of the same scene prior to the first use of the system. The proposed algorithm was tested on multiple datasets, including official external benchmarks and self-recorded drone videos in scrubland environments.
Additionally, the improved system is able to accommodate the effects of various lighting variations [E2] and camera ego-motion [I1] introduced in capturing the input images, since the augmentation techniques have been added.

8.2. Outlook for further work

Although the first steps have been taken to provide a reliable change detection system for UAV video streams, there is still a lot to be improved.
The current lack of suitable Training's dataset in the change detection domain is the first thing that should be addressed. A suitable dataset with corresponding ground truth based on real recordings is one of the keystones of a successful model. This also includes broader variations within the datasets, such as bad weather [E1] conditions, which were explicitly excluded in this work.
Also, the peak resolution tracked during the training sessions was 512x512px, which is still considered a very low resolution, especially for the detection of small objects like a rabbit or an RC car. If sufficient computing power is available both during training and on the inference hardware, the input resolution should be increased accordingly. Otherwise, if the downsizing is unavoidable, the algorithm used, which is currently realized with the integrated CV2 [37] resize function, should be adapted to be more CPU efficient.
In this work, the main focus was on using the U-Net architecture to perform the change detection. As other works have shown, other architectures such as FgSegNet[20] also perform well in background segmentation. Thus, it should be evaluated how other architectures compare in the domain of change detection.
Furthermore, there is no metadata, neither from the gimbal nor the aircraft used. Metadata could be used as another input to the neural network or perhaps for an image alignment step.
The timeframe between two frames could also be increased. Currently, the synthetic input videos have around 24 frames per second with a fixed frame stack size of 5 frames. A promising alternative is to use fewer frames per frame stack and thus reduce the input system size. Or use only two frames and calculate the change based on these two, with a variable delay between them.
Furthermore, preprocessing has been shown to drastically improve results. Although only a limited variation could be explored in the case of image alignment, the results were already promising. Here, a multitude of options are available such as image alignment, value averaging or normalisation. Some, like the variation of the grayscale generation, must also be done for the training data. However, some like colour normalisation, can be performed only on the inference system to gain a small advantage in inferencing at the expense of a significant amount of computing power.
To summarise, there are a lot of opportunities for improvement and a look at the results shows that there is still a lot of potential.

A. General Addenda

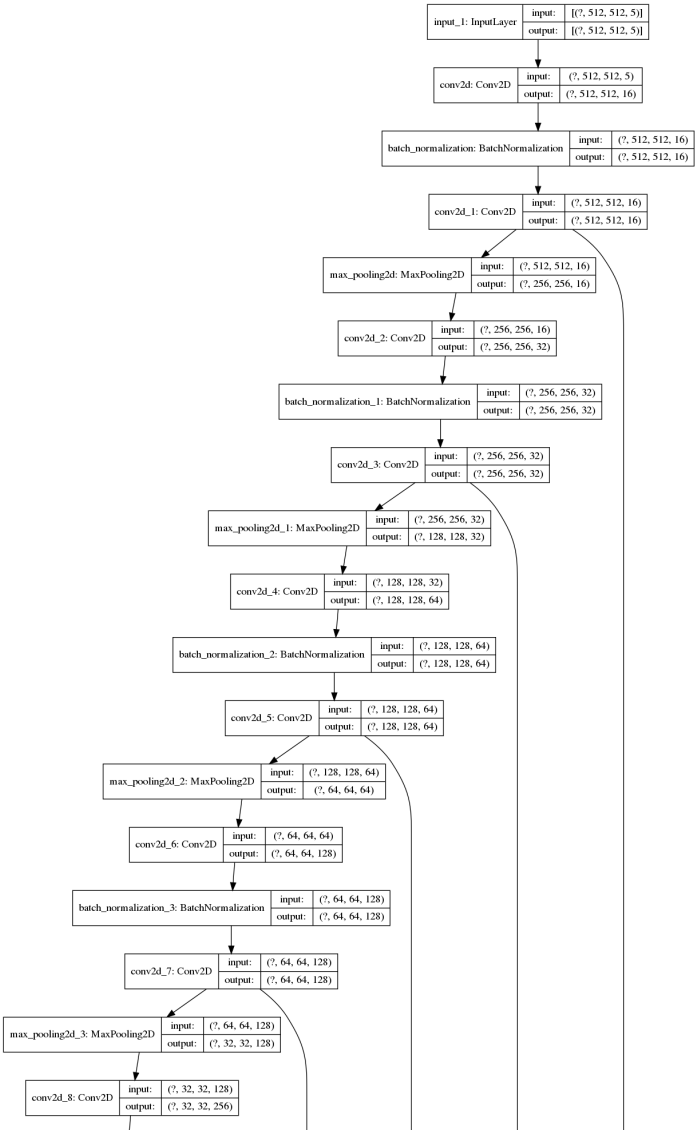


Figure A.1.: ISEEU-Net architecture visualised Part 1 of 2

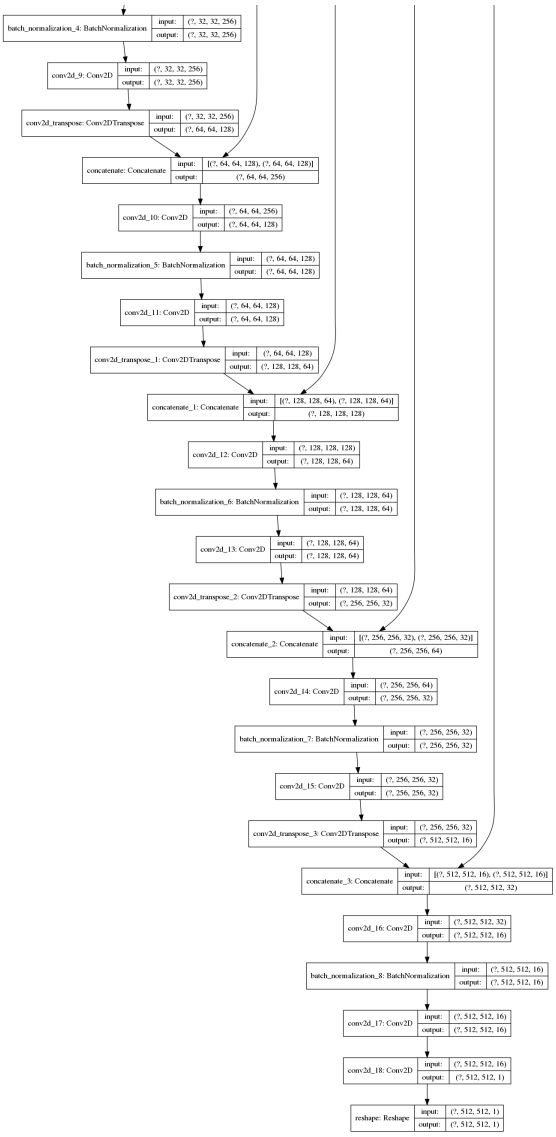


Figure A.2.: ISEEU-Net architecture visualised Part 2 of 2

Ours		**Original**	
Type	***Output Shape***	***Type***	***Output Shape***
InputLayer	None, 512,512,5	InputLayer	None, 572,572,5
Conv2D	None, 512, 512, 16	Conv2D	None, 572, 572, 16
BatchNormalization	None, 512, 512, 16		
Conv2D	None, 512, 512, 16	Conv2D	None, 572, 572, 16
MaxPooling2D	None, 256, 256, 16	MaxPooling2D	None, 284, 284, 1
Conv2D	None, 256, 256, 32	Conv2D	None, 282, 282, 128
BatchNormalization	None, 256, 256, 32		
Conv2D	None, 256, 256, 32	Conv2D	None, 280, 280, 128
MaxPooling2D	None, 128, 128, 32	MaxPooling2D	None, 140, 140, 128
Conv2D	None, 128, 128, 64	Conv2D	None, 138, 138, 256
BatchNormalization	None, 128, 128, 64		
Conv2D	None, 128, 128, 64	Conv2D	None, 136, 136, 256
MaxPooling2D	None, 64, 64, 64	MaxPooling2D	None, 68, 68, 256
Conv2D	None, 64, 64, 128	Conv2D	None, 66, 66, 512
BatchNormalization	None, 64, 64, 128		
Conv2D	None, 64, 64, 128	Conv2D	None, 64, 64, 512
MaxPooling2D	None, 32, 32, 128	MaxPooling2D	None, 32, 32, 512
Conv2D	None, 32, 32, 256	Conv2D	None, 30, 30, 1024
BatchNormalization	None, 32, 32, 256		
Conv2D	None, 32, 32, 256	Conv2D	None, 28, 28, 1024
Conv2DTranspose	None, 64, 64, 128	Conv2DTranspose	None, 56, 56, 512
Concatenate	None, 64, 64, 256	Concatenate	None, 56, 56, 1024
Conv2D	None, 64, 64, 128	Conv2D	None, 54, 54, 512
BatchNormalization	None, 64, 64, 128		
Conv2D	None, 64, 64, 128	Conv2D	None, 52, 52, 512
Conv2DTranspose	None, 128, 128, 64	Conv2DTranspose	None, 104, 104, 256
Concatenate	None, 128, 128, 128	Concatenate	None, 104, 104, 512
Conv2D	None, 128, 128, 64	Conv2D	None, 102, 102, 256
BatchNormalization	None, 128, 128, 64		
Conv2D	None, 128, 128, 64	Conv2D	None, 100, 100, 256
Conv2DTranspose	None, 256, 256, 32	Conv2DTranspose	None, 100, 100, 128
Concatenate	None, 256, 256, 64	Concatenate	None, 100, 100, 256
Conv2D	None, 256, 256, 32	Conv2D	None, 198, 198, 128
BatchNormalization	None, 256, 256, 32		
Conv2D	None, 256, 256, 32	Conv2D	None, 196, 194, 128
Conv2DTranspose	None, 512, 512, 16	Conv2DTranspose	None, 392, 392, 64
Concatenate	None, 512, 512, 32	Concatenate	None, 392, 392, 128
Conv2D	None, 512, 512, 16	Conv2D	None, 390, 390, 64
BatchNormalization	None, 512, 512, 16		
Conv2D	None, 512, 512, 16	Conv2D	None, 388, 388, 64
Conv2D	None, 512, 512, 1	Conv2D	None, 388, 388, 2
Reshape	None, 512, 512, 1		

Table A.1.: Our Architecture vs original U-net architecture

datasetName	A3_v11	A3_v11	A3_v11	A3_v11
videoName	11_150.avi	11_250.avi	11_50.avi	11_48.avi
Prevalence	0,007	0,015	0,003	0,001
Accuracy	0,996	0,988	0,997	0,998
Sensitivity	0,883	0,861	0,736	0,171
Fall-out	0,003	0,010	0,002	0,001
Miss-Rate	0,117	0,139	0,264	0,829
Specificity	0,997	0,990	0,998	0,999
predPositive	780898,000	1223567,000	407192,000	63650,000
predNegative	85464478,000	53302385,000	84003176,000	53413726,000
Precision	0,643	0,558	0,516	0,082
FalseDiscoverRate	0,357	0,442	0,484	0,918
FalseOmissionRate	0,001	0,002	0,001	0,000
NegativePredValue	0,999	0,998	0,999	1,000
LR+	271,066	85,488	314,260	156,341
LR-	0,118	0,141	0,264	0,830
DiagnosticOddsRatio	2306,392	607,632	1188,503	188,360
PWC	0,401	1,195	0,323	0,157
MCC	0,751	0,688	0,615	0,118
F1-Score	0,744	0,677	0,607	0,111

Table A.2.: The Benchmark on the RV4 v11 Dataset for 'bknulmgz'

datasetName	CD2014	CD2014	CD2014	CD2014	CD2014	CD2014	CD2014
videoName	PETS2006	badminton	canoe	copyMachine	corridor	cubicle	diningRoom
Prevalence	0,018	0,047	0,051	0,078	0,043	0,026	0,1
Accuracy	0,993	0,977	0,986	0,989	0,992	0,994	0,98
Sensitivity	0,976	0,929	0,987	0,982	0,914	0,937	0,855
Fall-out	0,007	0,021	0,014	0,011	0,005	0,005	0,006
Miss-Rate	0,024	0,071	0,013	0,018	0,086	0,063	0,145
Specificity	0,993	0,979	0,986	0,989	0,995	0,995	0,994
predPositive	5758456	5741998	6419854	65735588	55984526	48591659	71183458
predNegative	229122568	84959826	94505586	693433436	1227472498	1601866965	714199966
Precision	0,716	0,685	0,786	0,884	0,896	0,84	0,942
FalseDiscoverRate	0,284	0,315	0,214	0,116	0,104	0,16	0,058
FalseOmissionRate	0	0,004	0,001	0,002	0,004	0,002	0,016
NegativePredValue	1	0,996	0,999	0,998	0,996	0,998	0,984
LR+	137,503	44,459	68,696	90,264	192,735	193,879	145,742
LR-	0,024	0,072	0,013	0,018	0,087	0,063	0,146
DiagnosticOddsRatio	5615,663	615,964	5268,003	4937,365	2222,318	3054,159	1001,045
PWC	0,741	2,323	1,429	1,144	0,823	0,637	1,973
MCC	0,832	0,787	0,874	0,926	0,9	0,884	0,887
F1-Score	0,826	0,789	0,875	0,93	0,905	0,886	0,896

Table A.3.: The full Benchmark on CD2014 for 'bknulmgz' Part 1 of 2

datasetName	CD2014	CD2014	CD2014	CD2014	CD2014	CD2014	CD2014
videoName	**highway**	**lakeSide**	**office**	**park**	**pedestrians**	**peopleInShade**	**traffic**
Prevalence	0,081	0,033	0,088	0,03	0,021	0,068	0,075
Accuracy	0,975	0,993	0,992	0,98	0,988	0,966	0,943
Sensitivity	0,909	0,848	0,963	0,523	0,844	0,958	0,931
Fall-out	0,019	0,002	0,005	0,005	0,009	0,034	0,056
Miss-Rate	0,091	0,152	0,037	0,477	0,156	0,042	0,069
Specificity	0,981	0,998	0,995	0,995	0,991	0,966	0,944
predPositive	29346514	42599384	34308291	1900858	5659796	23958220	21229636
predNegative	292042030	1398144040	352616253	88800966	202744684	223767860	153358268
Precision	0,806	0,938	0,952	0,751	0,667	0,675	0,577
FalseDiscoverRate	0,194	0,062	0,048	0,249	0,333	0,325	0,423
FalseOmissionRate	0,008	0,005	0,004	0,015	0,003	0,003	0,006
NegativePredValue	0,992	0,995	0,996	0,985	0,997	0,997	0,994
LR+	47,156	446,457	206,193	97,344	91,294	28,386	16,745
LR-	0,093	0,153	0,038	0,479	0,157	0,044	0,074
DiagnosticOddsRatio	505,746	2922,63	5494,145	203,177	580,182	648,804	227,59
PWC	2,511	0,683	0,753	1,955	1,239	3,432	5,662
MCC	0,842	0,888	**0,953**	0,618	0,744	0,788	0,707
F1-Score	0,854	0,89	**0,957**	0,617	0,745	0,792	0,713

Table A.4.: The full Benchmark on CD2014 for bknulmgz Part 2 of 2

List of Figures

List of Tables

Bibliography

[1] A. Elqursh and A. Elgammal. "Online Moving Camera Background Subtraction". en. In: *Computer Vision – ECCV 2012*. Ed. by D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid. Vol. 7577. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 228–241. ISBN: 978-3-642-33782-6 978-3-642-33783-3. DOI: 10.1007/978-3-642-33783-3_17. URL: http://link.springer.com/10.1007/978-3-642-33783-3_17 (visited on 10/14/2020).

[2] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. "A Novel Video Dataset for Change Detection Benchmarking". In: *IEEE Transactions on Image Processing* 23.11 (Nov. 2014), pp. 4663–4679. ISSN: 1941-0042. DOI: 10.1109/TIP.2014.2346013.

[3] A. V. Etten. *The Satellite Utility Manifold; Object Detection Accuracy as a Function of Image Resolution*. Apr. 2017. URL: https://medium.com/the-downlinq/the-satellite-utility-manifold-object-detection-accuracy-as-a-function-of-image-resolution-ebb982310e8c.

[4] K. Sehairi, F. Chouireb, and J. Meunier. "Comparative study of motion detection methods for video surveillance systems". In: *Journal of Electronic Imaging* 26.2 (Apr. 2017). Publisher: International Society for Optics and Photonics, p. 023025. ISSN: 1017-9909, 1560-229X. DOI: 10.1117/1.JEI.26.2.023025. URL: https://www.spiedigitallibrary.org/journals/journal-of-electronic-imaging/volume-26/issue-2/023025/Comparative-study-of-motion-detection-methods-for-video-surveillance-systems/10.1117/1.JEI.26.2.023025.short (visited on 10/13/2020).

[5] E. Komagal and B. Yogameena. "Foreground segmentation with PTZ camera: a survey". en. In: *Multimedia Tools and Applications* 77.17 (Sept. 2018), pp. 22489–22542. ISSN: 1573-7721. DOI: 10.1007/s11042-018-6104-4. URL: https://doi.org/10.1007/s11042-018-6104-4 (visited on 10/13/2020).

[6] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. "A Review on Deep Learning Techniques Applied to Semantic Segmentation". In: *arXiv:1704.06857 [cs]* (Apr. 22, 2017). arXiv: 1704.06857. URL: http://arxiv.org/abs/1704.06857 (visited on 02/04/2021).

[7] D. Zeng and M. Zhu. "Background Subtraction Using Multiscale Fully Convolutional Network". In: *IEEE Access* PP (Mar. 2018), pp. 1–1. DOI: 10.1109/ACCESS.2018.2817129.

[8] C. Stauffer and W. Grimson. "Adaptive background mixture models for real-time tracking". In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. IEEE Comput. Soc, 1999, pp. 246–252. ISBN: 978-0-7695-0149-9. DOI: 10.1109/CVPR.1999.784637. URL: http://ieeexplore.ieee.org/document/784637/.

[9] P. KaewTraKulPong and R. Bowden. "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection". In: *Video-Based Surveillance Systems*. Ed. by P. Remagnino, G. A. Jones, N. Paragios, and C. S. Regazzoni. Springer US, 2002, pp. 135–144. ISBN: 978-1-4613-5301-0. DOI: 10.1007/978-1-4615-0913-4_11. URL: http://link.springer.com/10.1007/978-1-4615-0913-4_11.

[10] Z. Zivkovic. "Improved adaptive Gaussian mixture model for background subtraction". In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 2. Aug. 2004, 28–31 Vol.2. DOI: 10.1109/ICPR.2004.1333992.

[11] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis. "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance". In: *Proceedings of the IEEE* 90.7 (July 2002). Conference Name: Proceedings of the IEEE, pp. 1151–1163. ISSN: 1558-2256. DOI: 10.1109/JPROC.2002.801448.

[12] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin. "SuBSENSE: A Universal Change Detection Method With Local Adaptive Sensitivity". en. In: *IEEE Transactions on Image Processing* 24.1 (Jan. 2015), pp. 359–373. ISSN: 1057-7149, 1941-0042. DOI: 10.1109/TIP.2014.2378053. URL: http://ieeexplore.ieee.org/document/6975239/ (visited on 01/31/2021).

[13] S.-h. Lee, G.-c. Lee, J. Yoo, and S. Kwon. "WisenetMD: Motion Detection Using Dynamic Background Region Analysis". In: *Symmetry* 11.5 (May 2019), p. 621. ISSN: 2073-8994. DOI: 10.3390/sym11050621. URL: https://www.mdpi.com/2073-8994/11/5/621.

[14] A. Efros. *Automatic Image Alignment 15-463: Computational Photography*. 2011. URL: https://web.archive.org/web/20210201120708/http://graphics.cs.cmu.edu/courses/15-463/2011_fall/Lectures/feature-alignment.pdf (visited on 02/01/2020).

[15] V. García Rubio, J. A. Rodrigo Ferrán, J. M. Menéndez García, N. Sánchez Almodóvar, J. M. Lalueza Mayordomo, and F. Álvarez. "Automatic Change Detection System over Unmanned Aerial Vehicle Video Sequences Based on Convolutional Neural Networks". In: *Sensors* 19.2020 (Jan. 2019), p. 4484. DOI: 10.3390/s19204484. URL: https://www.mdpi.com/1424-8220/19/20/4484.

[16] M. Braham and M. Droogenbroeck. *Deep background subtraction with scene-specific convolutional neural networks*. journalAbbreviation: IEEE. May 2016, p. 4. DOI: 10.1109/IWSSIP.2016.7502717.

[17] Y. Wang, Z. Luo, and P.-M. Jodoin. "Interactive deep learning method for segmenting moving objects". In: *Pattern Recognition Letters*. Scene Background Modeling and Initialization 96 (Sept. 2017), pp. 66–75. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2016.09.014. URL: http://www.sciencedirect.com/science/article/pii/S0167865516302471.

[18] M. O. Tezcan, P. Ishwar, and J. Konrad. "BSUV-Net: A Fully-Convolutional Neural Network for Background Subtraction of Unseen Videos". In: *arXiv:1907.11371 [cs]* (Jan. 14, 2020). arXiv: 1907.11371. URL: http://arxiv.org/abs/1907.11371 (visited on 10/13/2020).

[19] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv:1505.04597 [cs]* (May 2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[20] L. A. Lim and H. Yalim Keles. "Foreground segmentation using convolutional neural networks for multiscale feature encoding". In: *Pattern Recognition Letters* 112 (Sept. 2018), pp. 256–262. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2018.08.002. URL: http://www.sciencedirect.com/science/article/pii/S0167865518303702.

[21] *Change Detection with Weightless Neural Networks*. URL: %5Curl%7Bhttps://www.researchgate.net/publication/262181320_Change_Detection_with_Weightless_Neural_Networks%7D.

[22] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. "Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes". In: *ECCV*. 2016. DOI: 10.1007/978-3-319-46484-8_33.

[23] Y. Wu, X. He, and T. Q. Nguyen. "Moving Object Detection With a Freely Moving Camera via Background Motion Subtraction". In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.2 (Feb. 2017), pp. 236–248. ISSN: 1558-2205. DOI: 10.1109/TCSVT.2015.2493499.

[24] L. Maddalena and A. Petrosino. "Towards Benchmarking Scene Background Initialization". In: *arXiv:1506.04051 [cs]* (June 12, 2015). arXiv: 1506.04051. URL: http://arxiv.org/abs/1506.04051 (visited on 12/31/2020).

[25] O. O. Family. *Drone and night footage of massive Big Rig Truck and Roadtrains*. 2016. URL: https://web.archive.org/web/https://www.youtube.com/watch?t=103&v=zFScuHBvfP8&feature=youtu.be (visited on 02/01/2020).

[26] N. M. Romero Aquino, M. Gutoski, L. Hattori, and H. Lopes. "The Effect of Data Augmentation on the Performance of Convolutional Neural Networks". In: Oct. 2017. DOI: 10.21528/CBIC2017-51.

[27] M. E. Akbiyik. "Data Augmentation in Training CNNs: Injecting Noise to Images". en. In: (Sept. 2019). URL: https://openreview.net/forum?id=SkeKtyHYPS (visited on 02/01/2021).

[28] G. Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[29] C. Garbin, X. Zhu, and O. Marques. "Dropout vs. batch normalization: an empirical study of their impact to deep learning". In: *Multimedia Tools and Applications* 79.19 (May 1, 2020), pp. 12777–12815. ISSN: 1573-7721. DOI: 10.1007/s11042-019-08453-9. URL: https://doi.org/10.1007/s11042-019-08453-9 (visited on 01/20/2021).

[30] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Jan. 2017). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 02/01/2021).

[31] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". In: *arXiv:1603.06560 [cs, stat]* (June 2018). arXiv: 1603.06560. URL: http://arxiv.org/abs/1603.06560 (visited on 02/22/2021).

[32] F. Chollet et al. *Keras*. https://keras.io. 2015.

[33] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker,

Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[34] L. Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[35] S. Boughorbel, F. Jarray, and M. El-Anbari. "Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric". In: *PLOS ONE* 12.6 (June 2017), e0177678. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0177678. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0177678.

[36] B. W. Matthews. "Comparison of the predicted and observed secondary structure of T4 phage lysozyme". In: *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405.2 (Oct. 1975), pp. 442–451. ISSN: 0005-2795. DOI: 10.1016/0005-2795(75)90109-9. URL: http://www.sciencedirect.com/science/article/pii/0005279575901099.

[37] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).