TU WIEN Informatics

Erkennung, Segmentierung und Klassifizierung von Halstüchern

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Sebastian Krähsmaier
Matrikelnummer 12122535

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller

Wien, 21. August 2025

Sebastian Krähsmaier	Eduard Gröller

TU WIEN Informatics

Detection, Segmentation and Classification of Scarves

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Sebastian Krähsmaier
Registration Number 12122535

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller

Vienna, August 21, 2025

Sebastian Krähsmaier	Eduard Gröller

Erklärung zur Verfassung der Arbeit

Sebastian Krähsmaier

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 21. August 2025

Sebastian Krähsmaier

Danksagung

Zunächst einmal möchte ich der Pfadfinder*Innenbewegung für ihre Existenz danken, die in mir eine Leidenschaft entfacht, die sich nicht in Worte fassen lässt. Diese Arbeit zu erstellen und dabei verschiedene Schals zu erkunden und neue Themen rund um die Pfadfinder*Innen zu entdecken, war ein wirklich lohnender Prozess, für den ich dankbar bin.

Meinen Bachelor-Betreuer Eduard Gröller, der mich immer motiviert hat und für unsere regelmäßigen Bachelorarbeitstreffen, bei denen es nicht immer um die Arbeit ging, sondern auch um andere gemeinsame Themen und vieles mehr.

Ich möchte meiner Schwester für ihre Hilfe bei der Erstellung von Datensätzen danken, ebenso wie meinem besten Freund Maximilian Negedly, an den ich mich immer wenden konnte und der meine Arbeit überprüft hat.

Acknowledgements

First of all I want to thank the scout movement for its existence, that fires a passion inside me that could not be described. Working on this, exploring different scarves, discovering new topics on scouts was a truly rewarding process I am thankful for.

My bachelor supervisor Eduard Gröller, for always keeping me motivated and for our regular bachelor meetings which were not always about this but other common topics, mind teasers and much more.

I want to thank my sister for helping creating datasets, as well as my best friend Maximilian Negedly I could always turn to as my rubber duck and looking over my work.

Kurzfassung

Halstücher sind ein grundlegendes Element in der Pfadfinderbewegung. Sie können die Zugehörigkeit zu einer Nation, einem Versprechen, einer Leistung, einer Organisation oder einer lokale Pfadfindergruppe symbolisieren. Da es so viele davon gibt, ist es ziemlich schwierig, sie alle voneinander zu unterscheiden. Das Ziel dieser Arbeit ist es, Halstücher in zwei verschiedenen Formen zu erkennen, zu segmentieren und zu klassifizieren. Die erste Form, in der Halstücher auftreten können, ist, wenn sie flach sind. Es werden zwei Ansätze untersucht: ein traditioneller Ansatz des maschinellen Lernens und ein Ansatz, bei dem ein vortrainiertes Netzwerk zur Klassifizierung verwendet wird. Die zweite Form sind Halstücher, die aufgerollt sind und um den Hals getragen werden. Ein Modell wird trainiert, um ein Halstuch oder mehrere Halstücher in einem Bild zu erkennen und zu segmentieren. Die Klassifizierung dieser erkannten und segmentierten Bereiche erfolgt mithilfe eines anderen Modells, das auf einem synthetisch generierten Datensatz basiert. Eine wichtige Erkenntnis dieser Arbeit ist, dass die Verwendung eines vortrainierten Netzwerks die Gesamtleistung verbessert. Die Verwendung eines synthetisch generierten Datensatzes ist komplexer als erwartet.

Abstract

Scarves are a fundamental item in the scout movement. They can represent the affiliation to a nation, a promise, an achievement, an organisation, a local scout group. As there are so many of them it is quite difficult to distinguish between all of them. The aim of this thesis is to detect, segment, and classify scarves in two different forms. The first form scarves can appear as if they are flat. Two approaches are explored, one being a traditional Machine Learning approach and the other one using a pre-trained network to classify them. The second form are scarves rolled up and worn around the neck. A model is trained to detect and segment one or multiple scarves from an image. The classification of these detected and segmented patches is realized using another model that is powered by a synthetically generated dataset. A key finding of this work is that using a pre-trained network improves the performance overall. The usage of a synthetically generated dataset is more complex than anticipated.

Contents

CHAPTER 1

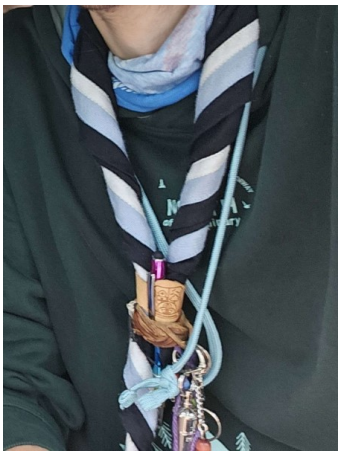
Introduction

Scarf

Scarves have been introduced to scouts with the founding of the scouts. The founder of the scouts, Robert Baden-Powell says,

"... the scarf or neckerchief which is folded into a triangle with the point at the back of the neck. Every Troop has its own scarf colour, and since the honour of your Troop is bound up in the scarf, you must be very careful to keep it clean and tidy. It is fastened at the throat by a knot, or "woggle", which is some form of ring made of cord, metal, or bone, or anything you like. The scarf protects your neck from sunburn and serves many purposes, such as for a bandage or as an emergency rope." [fBBPoG08]

A scarf, sometimes also called a neckerchief or necker, is traditionally worn by scouts. It is a symbol that shows where someone belongs to. This can be either the local scout group, the state, the country, the troop. A scarf can also represent the participation in a scout camp or event, a position, or an achievement. In addition to the "woogle" a scarf can also be decorated with other trinkets. There are no strict rules on the designs of scarves, only to respect each other design and to not copy familiar scarves. The most dominant feature of a scarf are the stripes close to the edge, when the scarf is rolled up they are seen the most. Having no stripes is also possible as seen in Fig 1.1f. In Fig 1.1 more examples of scarves and what purpose they serve can be seen.



(a) Scarf of a local scout group known as "W63 - Ottakring"

(b) Scarf of a local scout group formerly known as "W56113 - Andreas Hofer"

(c) Scarf of a local scout group known as "W78 - Nibelungenviertel"

(d) Scout leader scarf for a scout camp organized by Upper Austria

(e) Austrian national contingent scarf for a scout camp in Norway

(f) Scarf of members from the World Organization of Scout Movement (WOSM) [wos25]

Figure 1.1: Different kinds of scarves and where they belong

CHAPTER 2

Related Work

As the topic of this thesis is a niche use-case there is not much academic work done on the topic as far as I can tell. Therefore, the related work is about other symbols that represent their affiliation, one of the most prominent ones are flags.

2.1 Flag Detection with Convolutional Network

The focus of the work of Gu et al. [GHQ18] is to detect if a specific flag is present in the image. Due to the structure of a flag, it gets easily deformed, obstructed or a change of lighting that alter the appearance of the flag. The model design for the Convolutional Neural Network (CNN) is rather simple. The first few layers have a convolutional layer with a 3x3 kernel, at layer six and seven a 1x1 convolutional layer is used. Compared to a traditional fully-connected layer it is a very flexible method for either increasing or decreasing the dimension. In their model the number of parameters is reduced and a slightly better accuracy is achieved by using a 1x1 convolutional layer. Additionally, they introduce a Multi-Scale Matching Strategy, to reduce the computation time. Each image is classified into either Full-Scale, Middle-Scale, and Micro-Scale which describes how much space the flag takes in the image. Based on this the parameters for the sliding window are adjusted. For example the striding of the sliding is higher for a Full-Scale image than for a Micro-Scale image and also only one sliding window is needed instead of multiple ones. Their method shows better results than the previous method the paper it is comparing to, but for our use-case it is not suitable. The method relies on a big dataset with more than 800 images and an untrained model. Furthermore, the goal of the paper was only to detect one flag in an image without classification.

2.2 Flag Net

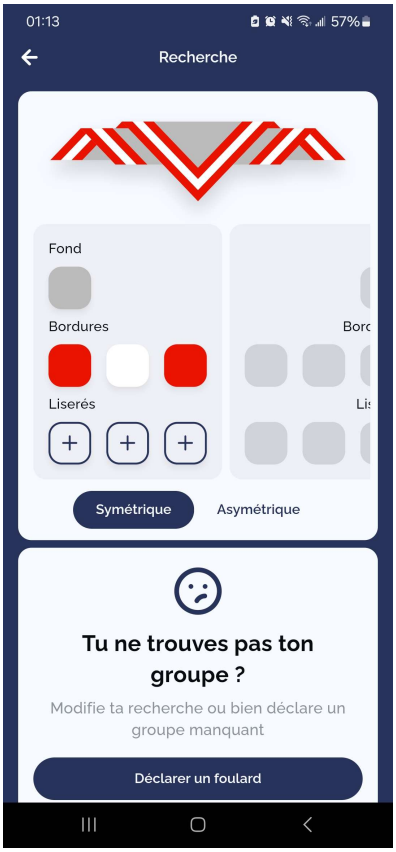
For the flagnet [Man18] the network creation, especially the dataset, is more interesting than the network architecture. To detect all country flags of the United Nations members, the dataset was created using a synthetic process. Through a web application, a flag has been simulated to fly in the wind attached to a hoist. The background is also interchangeable. This is the foundation of the dataset, for each image, there is a separate file describing the additional information about the image and the bounding box of the flag inside the image. Some examples can be seen in Fig 2.1

Figure 2.1: Examples of synthetically generated dataset by Flag Net (images generated by Vukašin Manojlović)

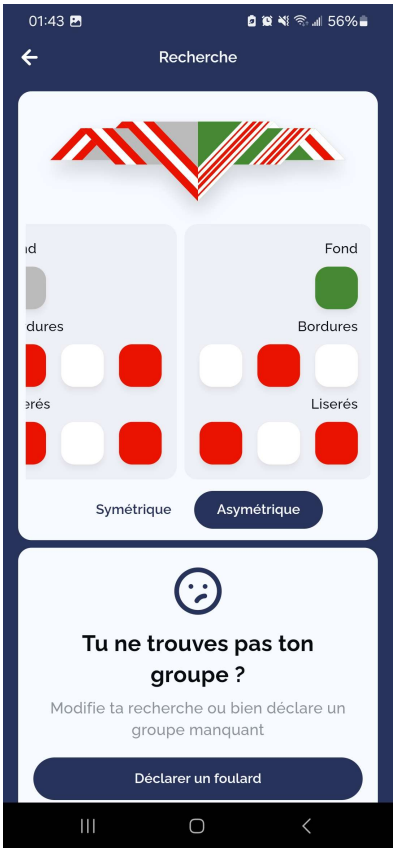
2.3 Scarves App

The only related work with the same objective as this thesis, but with a completely different approach, is an app called Scarves [Sca24]. Instead of using an image to identify the scarf, the scarf can be build with their scarf editor as seen in Fig 2.2a. The main color of the scarf has to be chosen and additionally up to three stripes at the border and up to three edges that run inside the stripes can be selected. To add even more customization the scarf does not have to be symmetrical and each side of the scarf can

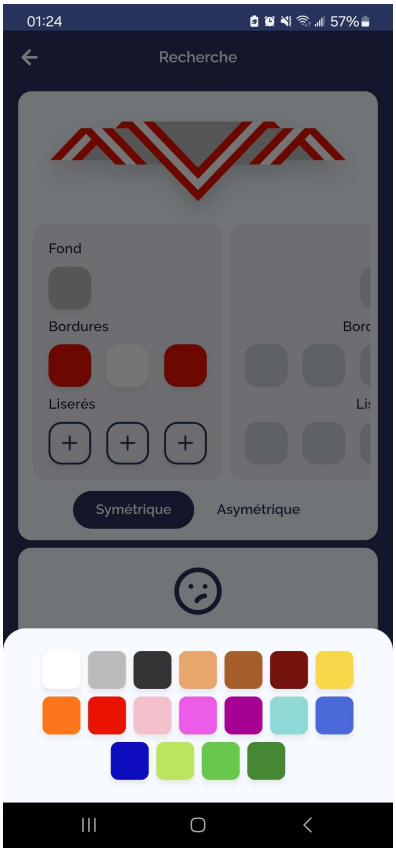
be designed differently as seen in Fig 2.2b. The color pallet offers 18 different colors to chose from as seen in Fig 2.2c. Many scout movements have already contributed to the app, as seen in Fig 2.2d, but most only provide their logo and not the scarf which makes the search for scarves unusable. Furthermore, the support for other patterns or some logos on the scarf is not given.



(a) Scarf builder



(b) Asymmetrical scarf and edges



(c) Color choice

(d) List of national scout movements

Figure 2.2: Different views of the scarves app

CHAPTER 3

Methods

This chapter describes the different approaches to achieve the classification, detection and segmentation of scarves. For the flat scarves only classification is used, whereas for the rolled scarf around the neck segmentation, detection, and classification are used. One model will be trained on classifying flat scarves. For the rolled scarves the first model will be trained to detect and segment the scarves in general and not on specific classes of scarves. The segmented patch will then be used as the input for another model to classify the scarf. The dataset for the second model will be synthetically generated.

3.1 Flat Scarf

The flat scarf is a simple triangle with stripes of different colors (Fig 3.1) that is rolled together if worn around the neck. Two different approaches are explored to classify them. The first is using a traditional Machine Learning method, the second is using a pre-trained Deep Learning network.

The traditional Machine Learning methods have to extract hand-crafted features from the images. Using the images without preprocessing, would be a too big and fine-grained input. The picked features can be known features such as Scale-invarant feature transform (SIFT), texture based features or color based features. Multiple types of features can be appended to each other to represent the image in a single feature vector. These features are then used as an input for a Machine Learning method such as a Support Vector Machine (SVM) that will act as the classifier. An abstraction of the process pipeline can be seen in Fig 3.2. The Deep Learning approach is simpler. Training a whole network from scratch takes a lot of data and is therefore not suited for the task. Using an already pre-trained model and re-training it to only classify the scarves boost the accuracy of the model. To evaluate how the pre-trained model performs, two approaches are taken. In one approach the inner layers are frozen and in the approach not. Additionally, for

Figure 3.1: Example of a flat scarf

both approaches Data Augmentation is applied during training. An abstraction of the process pipeline can be seen in Fig 3.3.

3.2 Rolled Scarf

The rolled scarf is what is most often seen as all scouts wear it around the neck. Some examples of a rolled scarf around the neck can be seen in Fig 1.1, an example of a rolled scarf not around the neck can be seen in Fig 3.4

There are different approaches to classify rolled scarves. In this case, two different models are used. The first model detects the scarf in an image and segments it. In the dataset for the first model, for every image in the dataset there will be a corresponding trimap that describes what part of the image is important and what to ignore. The detected scarves will then be cut out from the image and are used as the input to the second model. The second model is trained to classify the scarf and has a special dataset that is synthetically created. The dataset is described in the Section 3.3 in more detail. Similar to the Deep Learning approach with flat scarves, a pre-trained model is adapted to the use-case. An abstraction of the process pipeline can be seen in Fig 3.5. Another approach would be to only train one model, that directly detects and classifies the scarves. This would need a much larger dataset as every different scarf needs enough images on its own. It is not feasible to gather that amount of data and not as easily expandable. The advantage of having two separate models is the amount of data that is needed for each task. Furthermore, it is easily expandable as the segmentation model only needs to be trained once. As the data for the classifier model is synthetically generated, the images for new scarves can be easily added, and the model can be retrained with the new scarves.

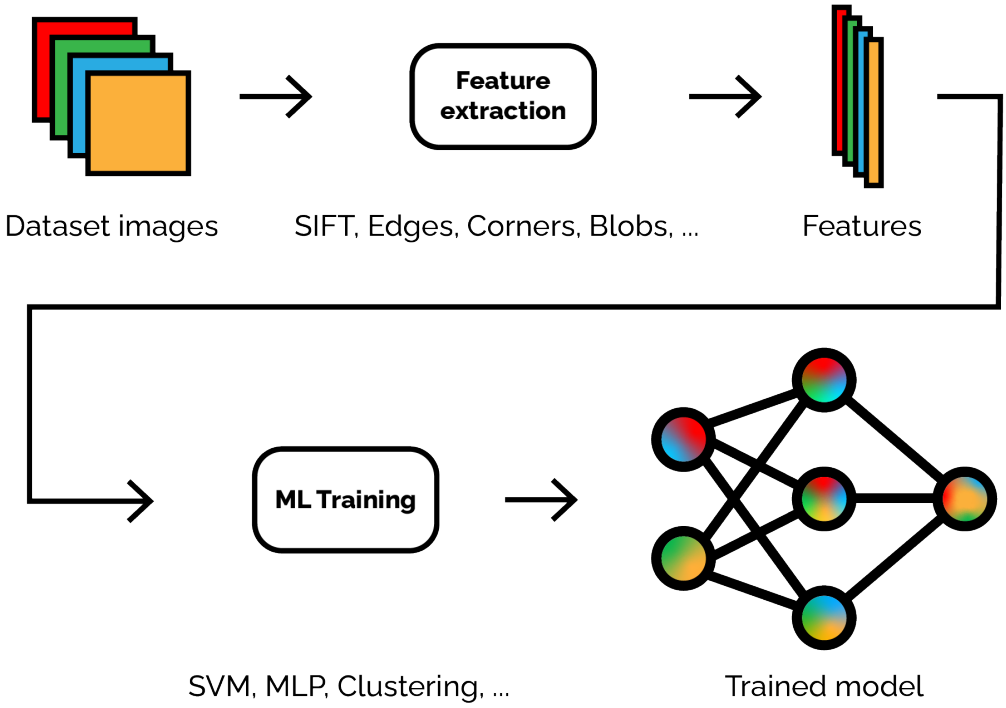


Figure 3.2: Workflow of training a traditional Machine Learning model

3.3 Synthetic Dataset

There a multiple ways to create a synthetic dataset, this can be done through rendering, editing the images per hand, or using an AI system for generation. Using another AI model does not solve the dataset issue as there would be another need to train the AI system for the task. Editing images by replacing the scarf is a possibility, but takes a lot of time for the large number of images needed. Rendering takes a lot of inital work to create all assets, setup the scene, and do it in a way to make it easily expandable for new scarves. One way to create a 3D model of a scarf would be to 3D photoscan the scarf. Every new scarf that is to be added would need its own 3D scan as replacing the texture for it would not be possible. Additionally a 3D photoscan leaves multiple artifacts that would have to be removed manually afterwards. The approach taken in this thesis is to create the 3D model through modelling. The scarf is constructed by a simple triangle that is rolled up. The rolling can be approximated using an Archimedian spiral. The texture of the scarf can be applied to the triangle and is then transformed through the modelling process. The texture and colors of the scarf should be close to a realistic appearance to ensure a good dataset. The biggest benefit is that the scarf texture only needs to be created once to create multiple scarves in the dataset. To create different

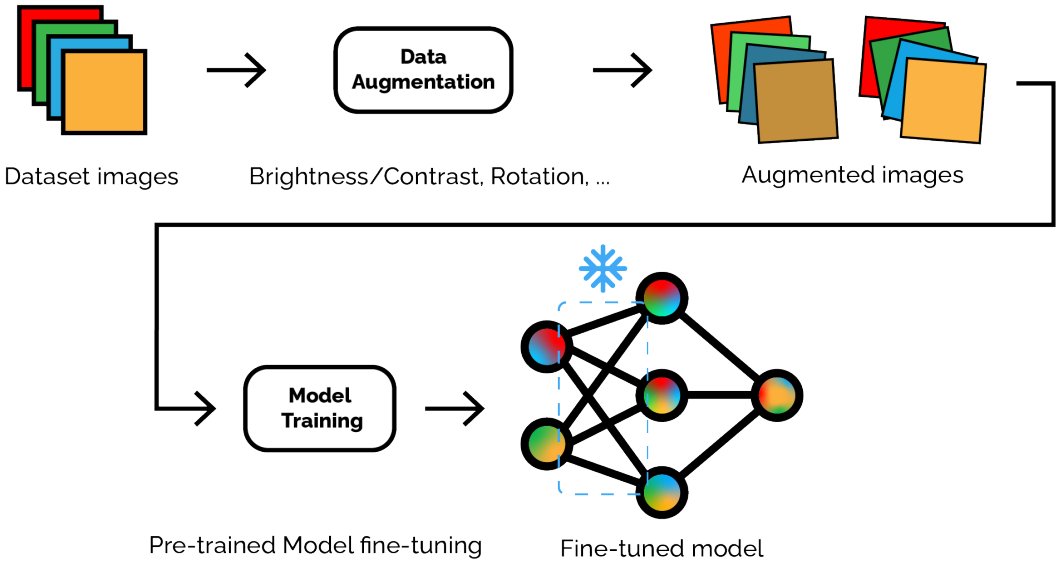


Figure 3.3: Workflow of training a Deep Learning model

Figure 3.4: Example of a rolled scarf not around the neck

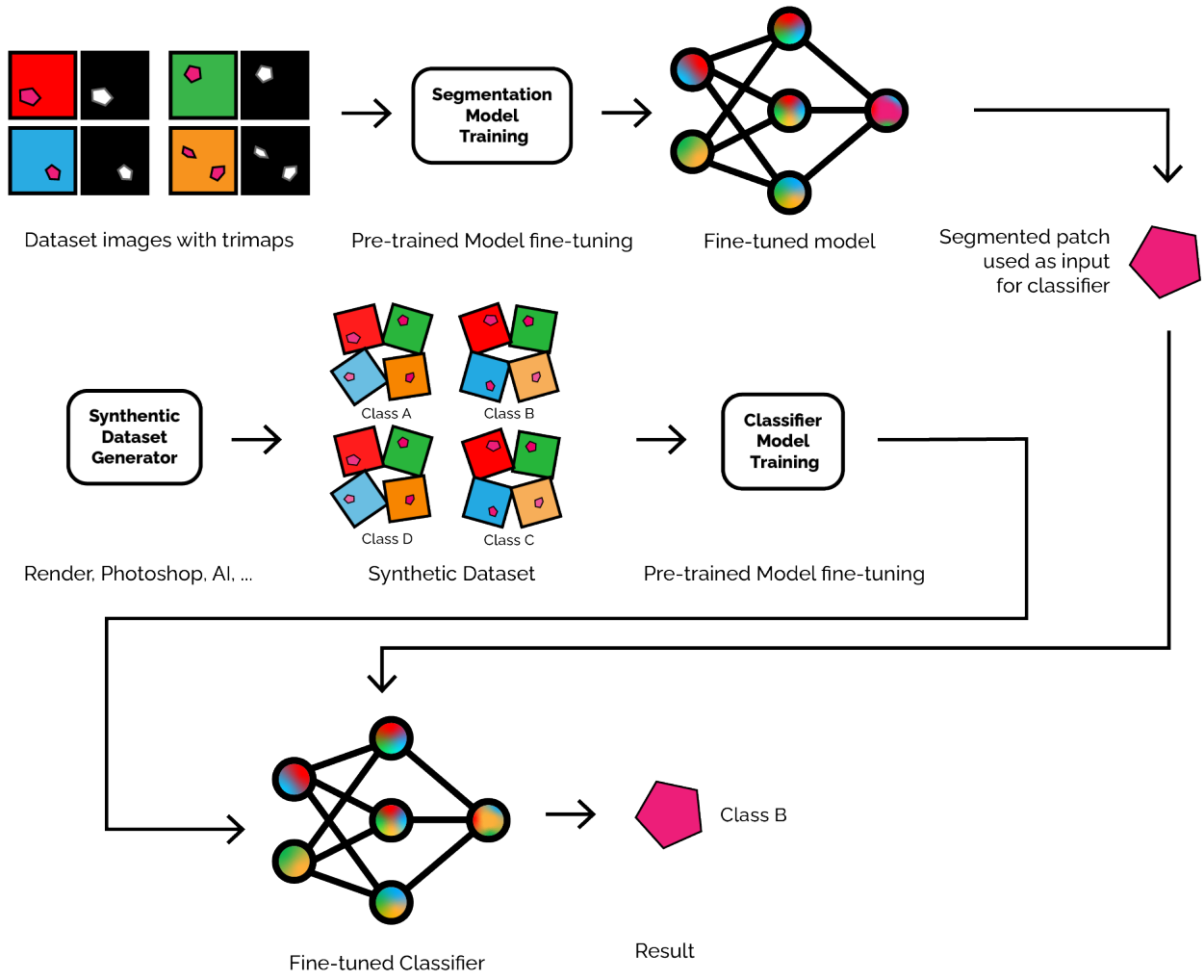


Figure 3.5: Detection, segmentation, and classification to process rolled scarves

views in the rendering of the scarves, an automatic pipeline can be used to streamline the process.

CHAPTER 4

Implementation

This chapter describes the different implementations to achieve the classification, detection, and segmentation of scarves. It describes which features, which python libraries, which pretrained models are used. In general python 3.12 is used if not stated otherwise and all libraries that are used were gathered through pip.

4.1 Flat Scarf

The flat scarf is the most trivial representation of a scarf. Its a triangle made of fabric with a specific pattern on it.

4.1.1 Dataset

The dataset is focused on the scarves from different scout groups in Vienna, all of them belong to the Vienna Scouts [wpp25], as there are a lot of them and readily available.

Two datasets were created in the process. The first one comprises a few photos, each one of them prioritising the key features of a scarf and additionally a few photos of the scarf crumbled up as a pile. The dataset contains a total of 58 scarves and of each scarf six images are taken. Each photo was post-processed to correct color differentiations due to lighting and camera settings. Every image was taken using an aspect-ratio of 1:1. The camera that was used to take the images is a LUMIX DMC-G7 and as the post-processing software Lightroom Classic [Ado25] was used. It took about eight hours to capture all images. This dataset contains all scarves from the scout groups belonging to the Vienna Scouts. An excerpt from the dataset can be seen in Fig 4.1.

The second dataset is composed of about one hundred photos per scarf. The goal was to take as many pictures as possible in different environments with the scarf as the prominent object. Because of this massive number of pictures and time constraint at

Figure 4.1: Excerpt from the first dataset of flat scarves

accessing the scarves only 17 scarves have been added to the dataset. The pictures were taken with an aspect-ratio of 1:1 and two mobile phones a Samsung Galaxy S23+ and a Samsung Galaxy S23. It took about ten hours to capture the images. An excerpt from the dataset can be seen in Fig 4.2. Of this dataset, two different sub-datasets exist, one subset where the background of each scarf is removed and replaced with a solid color and the other subset has no alterations to the pictures.

4.1.2 Traditional Machine Learning Approach

The colors are the most prominent feature of a scarf, and are the basis of the hand-crafted features. Instead of the commonly known Red-Green-Blue (RGB) color model, the Hue-Saturation-Value (HSV) color model is used to create the color histogram. Additionally, in order to not only be dependent on the colors, a second hand-crafted feature is extracted.

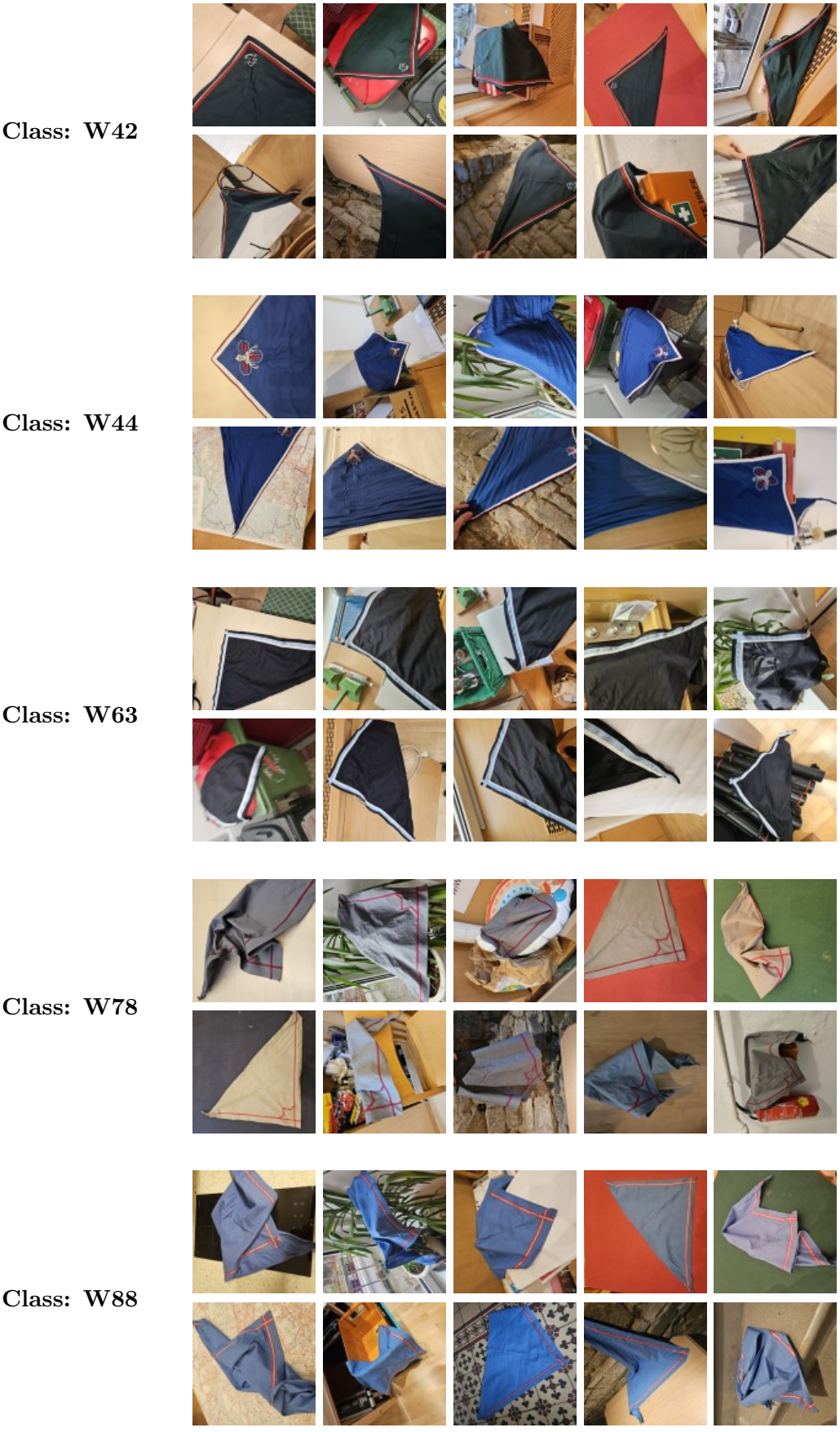


Figure 4.2: Excerpt from the second dataset of flat scarves

The focus of this feature are not the colors but rather the textures of the scarves. The so-called Haralick texture [HSD07] is used as it can represent the texture of a fabric. It does this by storing for each pixel the amount of each different neighbouring of this specific pixel in a gray-level co-occurrence matrix (GLCM). Then each of this elements in the matrix is given a weight. Similar gray values are assigned a low weight, whereas dissimilar gray-values are assigned a high weight. Combining these two features in a vector is our hand-crafted description of an image.

For the sub-dataset where the background was replaced with a solid color, this color is subtracted from the color histogram and ignored for the haralick texture. For the sub-dataset without any alterations no changes to the process are made. The features of both sub-datasets are the input to the classifier. The classifier is a Support Vector Machine (SVM) and two models - one with a linear kernel and one with a RBF (Radial Basis Function) kernel - has been trained.

Both the implementations extracting the hand-crafted features and performing the classifier are done using the python libraries scikit-learn [sld25], OpenCV for phyton [Tea25b], NumPy [ea25], Mahotas [Coe25].

4.1.3 Pretrained Network Approach

The second approach of using a pre-trained network is a common practice for special use-cases. Instead of training the whole network from the beginning, an already trained network will be used and only the last layers that are responsible for the classification are replaced with new ones that serve the new objective. The pre-trained network is provided by Torchvision [Tea25d] and is called *efficientnet_b3* [TL19]. Because of its small size, its speed and its transfer learning accuracy this model was chosen. This model is trained on the ImageNet [Tea25a] dataset. The importation of the dataset is done with PyTorch [Tea25c].

As for the Data Augmentation, the python library Albumentations [Igl25] is used. In particular the following properties of the images are randomly adjusted. The Normalization is depended on the training data as *efficientnet_b3* has been trained with ImageNet the mean and the standard deviation are deprived from the ImageNet dataset.

- Horizontal Flip - 50 % chance of occurrence
- Vertical Flip - 50 % chance of occurrence
- Random Rotation - max. +/- 15 degrees with a 50 % chance of occurrence
- Random Brightness and Contrast - max. +/- 0.2 brightness and contrast with a 50 % chance of occurrence
- Resize to a resolution of (256,256) - to match the networks requirements
- Normalization - to match the networks training data for optimal accuracy

To retrain the last layer for the objective as a Loss function CrossEntropyLoss and for the optimizer Adaptive Moment Estimation [KB17] (ADAM) with a learning rate of 0.001 is used. Additionally to training the last layer, the inner layers are fine-tuned as well. For the optimizer, also ADAM is used but with a learning rate of 0.0001. For both approaches the model is exported afterwards. Furthermore, both deliver different accuracies with the same dataset as can be seen in the Results section 5.1.

4.2 Rolled Scarf

The rolled scarf is complicated to detect as it is most of the time decorated with trinkets and therefore not always clearly visible.

4.2.1 Dataset

In this process, again, two datasets are created but each one serves a different purpose. The first dataset focuses on scarves in general without distinction to which class they belong. The purpose is to segment any scarf from a picture. The images were extracted out of a private archive manually and the dataset consists of about 120 images. For each of these images a trimap was created. A trimap is a mask that describes an image. It has three regions, i.e. foreground, background, and unknown region. These regions describe the important parts, the parts that are not interesting for the objective and the part that is unclear. The unclear part can be mostly seen as a padding zone around the foreground in this case. An example on how this looks like can be seen in Fig 4.3.

Figure 4.3: Image with two scarves and the corresponding trimap

The second dataset one is a synthetically generated one that focuses on classifying the scarves. How the synthetic dataset is generated is explained in the Section 4.3.

4.2.2 Detection, Segmentation, and Classification

For the detection and segemention of rolled scarves YOLOv11 [JQ24] (You Only Look Once) is used. The library, through which YOLOv11 is used, is provided by ultralytics

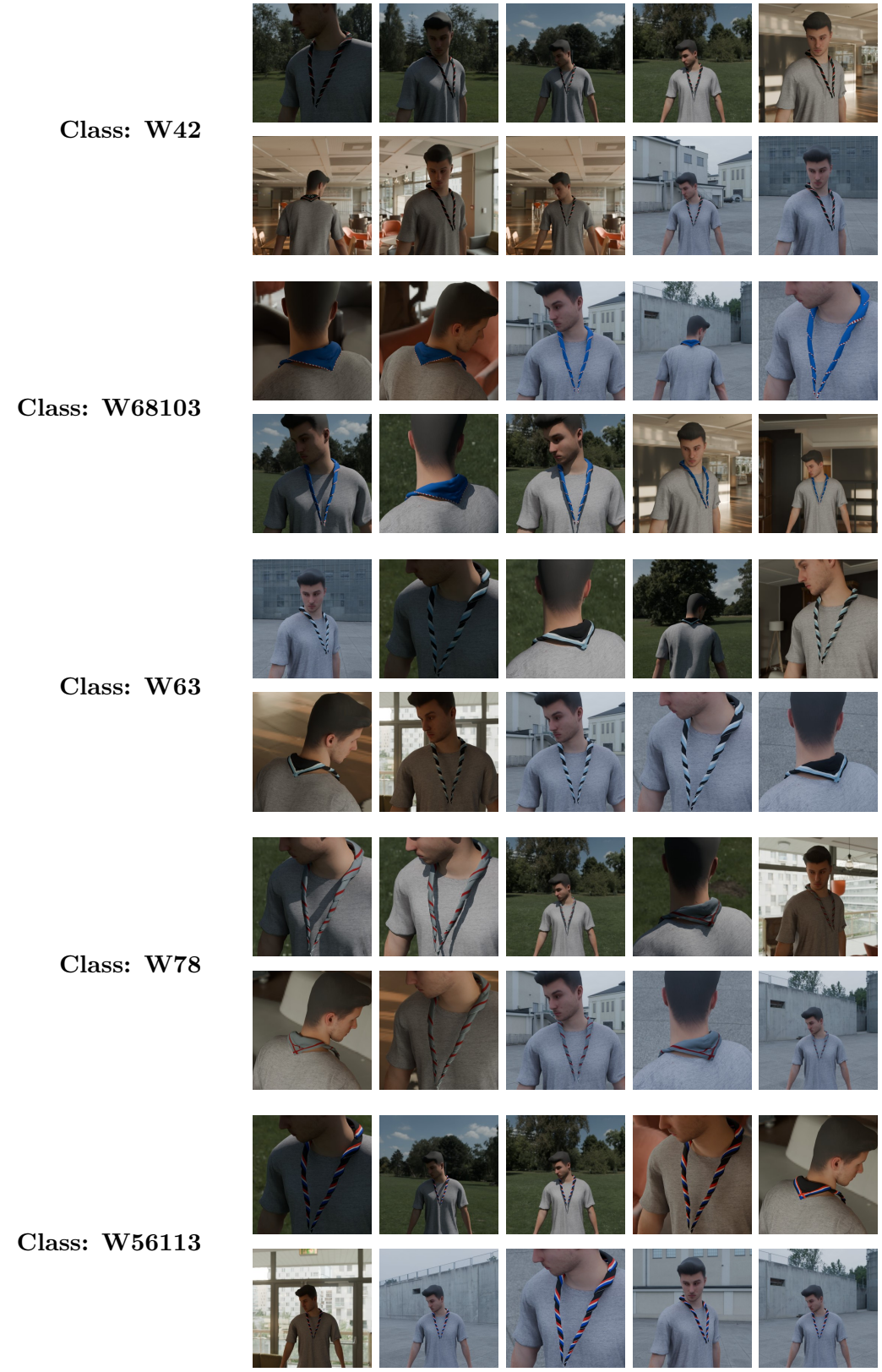


Figure 4.4: Excerpt from the synthetic dataset

[Ult25]. The model for the classification has the same architecture as the one for the Deep Learning approach for flat scarves.

The current dataset, especially the trimaps, are not useable in this form as the YOLO model requires the trimaps in polygon describing the foreground of the image. The trimaps have to be transformed to polygons that describe the important parts in the images. The OpenCV library for python offers a function called *findContours*. The extracted contours are filtered so only polygons with a minimum number of points (coordinate pairs) are used. Furthermore, the filtered polygons are simplified. That means the number of points which are used to describe a polygon are reduced. The maximum distance of the approximation and the original curve should not exceed 0.6 % of length of the original curve. This is achieved by another function from the OpenCV library called *approxPolyPD*. This function also ensures that it produces always a closed polygon so the starting point is equal to the end point. The last step is to write this information into a text file with the following structure: Each row in the text represents a scarf in a specific image, if the file has multiple rows there are multiple scarves in the corresponding image. The first number specifies the class, in this case, it is always zero as the model is only trained to recognize scarves in general. After the class identifier a list of points is appended with the X and Y values of the coordinate pair alternating. This values are relative to describe a point in the image.

Algorithm 4.1: Configuration file for YOLOv11 training

```
path: Data/YOLO/sorted
train: images/train
val: images/test

names: [scarf]
```

The training of the model is rather simple. By setting the path to the dataset and defining the classes in a specific file, that can be seen in Algorithm 4.1, the model has all information to start the fine-tuning process. During the fine-tuning process the model will not recognize any classes learned before but only the class scarves. The results of the fine-tuned network can be seen in the Results Chapter 5.2. Afterwards the model is exported and can be used anytime.

4.3 Synthetic Dataset

The synthetic dataset is created using Blender [Fou25]. The texture for a flat scarf is created digitally and with Blender the triangular plane is modelled into a rolled scarf. The process of swapping the texture of a scarf, the environment texture and camera positions happens automatically. This is to render multiple scarves sequentially and diversify the dataset and will be called automation pipeline.

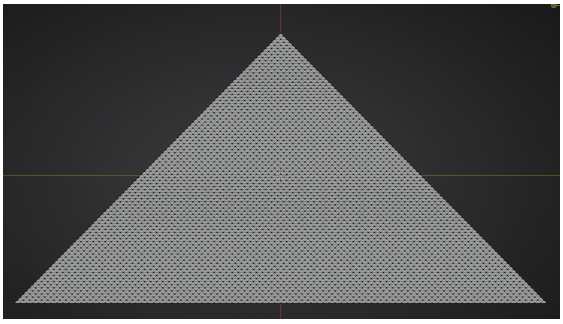
4.3.1 Creation of the model

Modelling The process of creating a rolled scarf 3D model is similar to how it happens in the real world. It first starts with a flat triangle that has a lot of subdivisons, resulting in a total of 7744 triangles, so it is bendable later (Fig 4.5a). The next step is to roll the scarf, the roll pattern can be approximated using an Archemedian spiral (Fig 4.5b). By applying a curve modifier in Blender that lets the plane follow along the spiral curve, the scarf is now rolled but still in a straight line (Fig 4.5c). To create the curvature of the scarf a skeleton is added to make it easier to bend (Fig 4.5d). Some manual work is necessary to prevent the rolled up surfaces from overlapping too much. The last step is to flatten the scarf as it is not a perfect cylinder in the real world. This is achieved by hanging the scarf onto a human model (Fig 4.5e) and applying a cloth simulation. The cloth simulation flattens the scarf to make it look more realistic. With some simple additional manual adjustments to remove too excessive overlapping of the surfaces, the base for a 3D model of a scarf is created (Fig 4.5e).

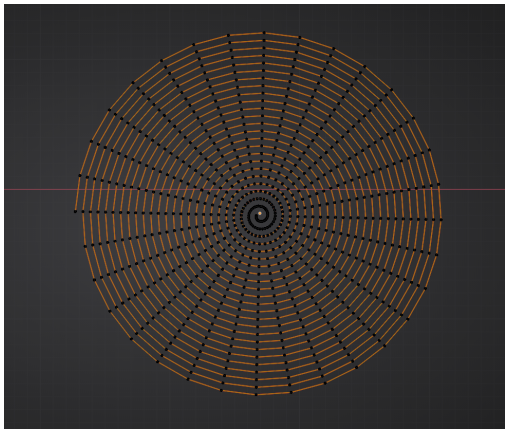
In the real world, scarves are not always rolled the same way, some are rolled more tightly some are rolled more loosely. To simulate this as well, the flat scarf is moved along the Archemedian spiral to simulate different curvatures.

Texturing After the creation of the 3D model, the scarf needs a texture to display the correct pattern. For this, a simple graphic for the scarf is created as seen in Fig 4.6a. To ensure that the graphic is correctly displayed onto the model, a UV map (Fig 4.6b) is created for texturing. A UV map assigns to every point on the model surface coordinates from the graphic. How the scarf looks like with only the scarf graphic applied as a texture can bee seen in Fig 4.6c. The next step is to make the scarf look more realistic. By creating a material (Fig 4.7) and adjusting the properties like changing the roughness, adding displacements to the surface and applying fabric texture to the material, the final 3D model can be seen in Fig 4.6d. This is just a basic material and only approximates the properties a scarf has in the real world. The material setup is chosen, as As the rendered image will be in a low resolution, creating a complex material is not necessary. This ensures that the render does not take too long.

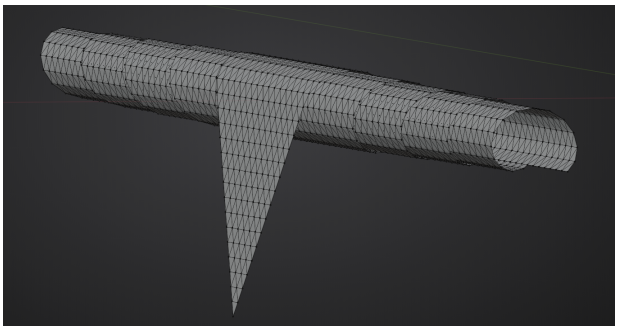
Setup for the automation pipeline To simplify the process for swapping the texture of a scarf, the environment texture and camera positions and the scenes needs to be prepared. How exactly the automation pipeline works is explained in Subsection 4.3.2. To approximate the real world, the scarf is put onto a human model and a background texture is added (Fig 4.8a). The background is modelled with an environment texture. Unlike for arbitrary surfaces textures, environment textures are designed to be mapped around a cube or a sphere enclosing the entire scene. Displaying the environment texture on a flat surface, it is stretched as can be seen in Fig 4.8b. The environment texture has the additional information stored to simulate light sources. The light sources that can be seen in the environment texture (e.g., sun, light bulbs, LEDs, ...) are also the light sources for the scene. Similar to the scarf and the texturing process the human model has the same properties. The human model has already an UV map and only a



(a) Flat scarf subdivided into triangles

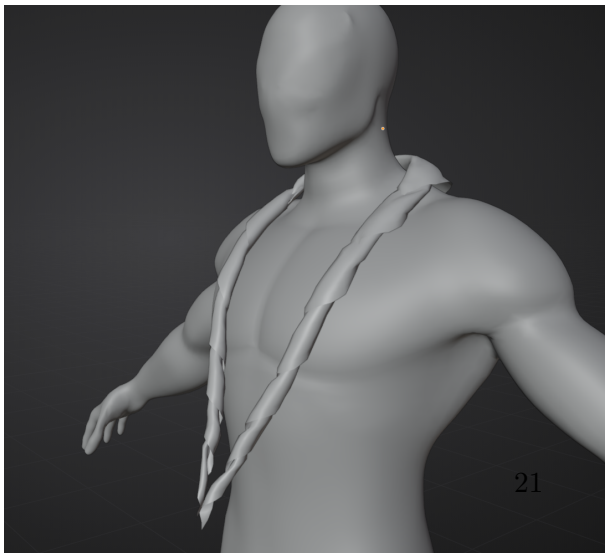


(b) Archemedian Spiral



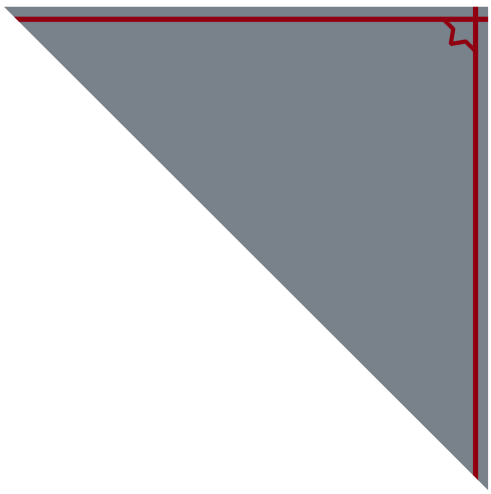
(c) Rolled scarf along a straight line

(d) Rigged rolled scarf

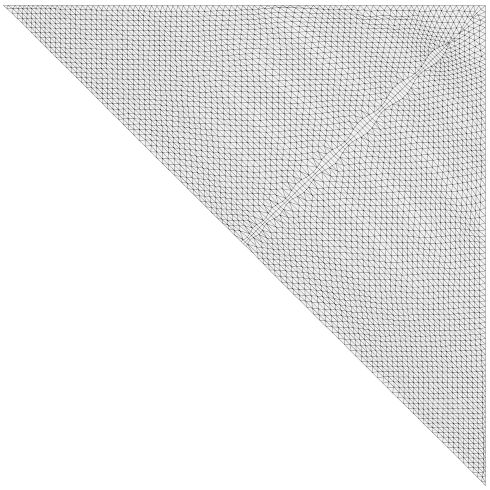


(e) Rolled scarf on a human model before and after cloth simulation

Figure 4.5: Procces of modelling a scarf in 3D



(a) Scarf graphic



(b) UV Map for the scarf

(c) Scarf with image only applied

(d) Scarf with material applied

Figure 4.6: Texturing process of the scarf

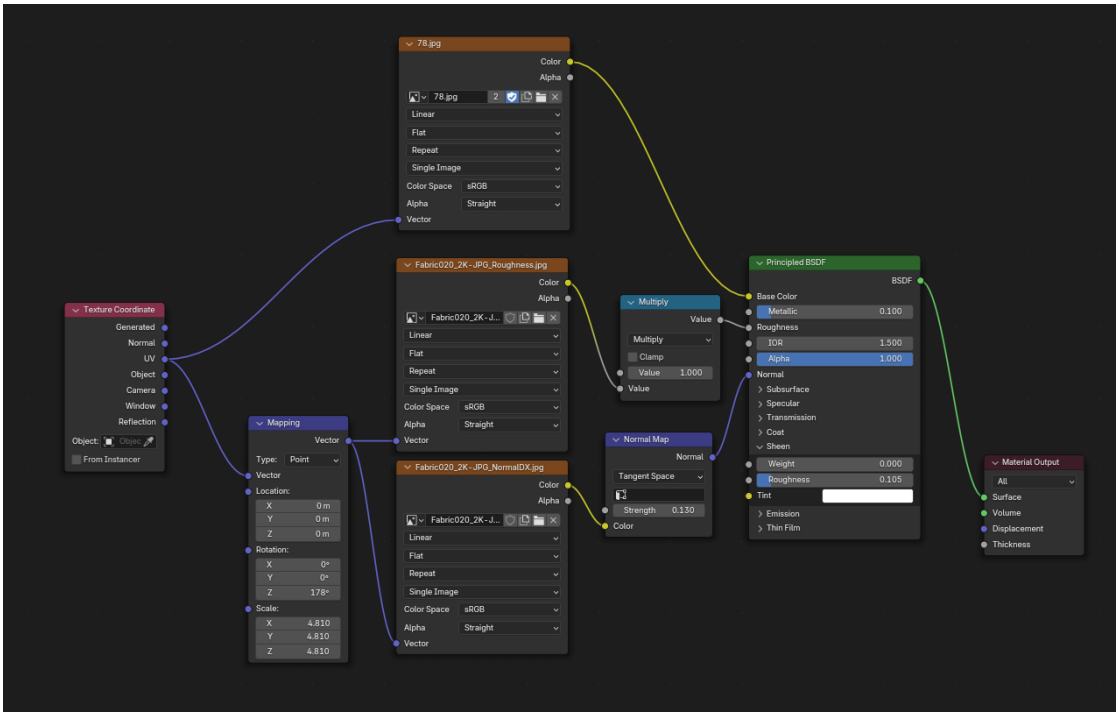
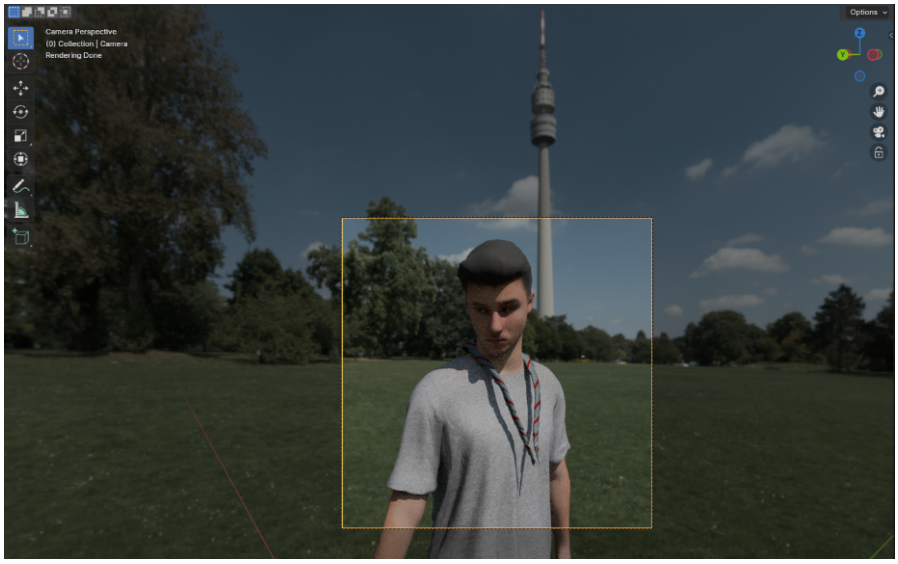


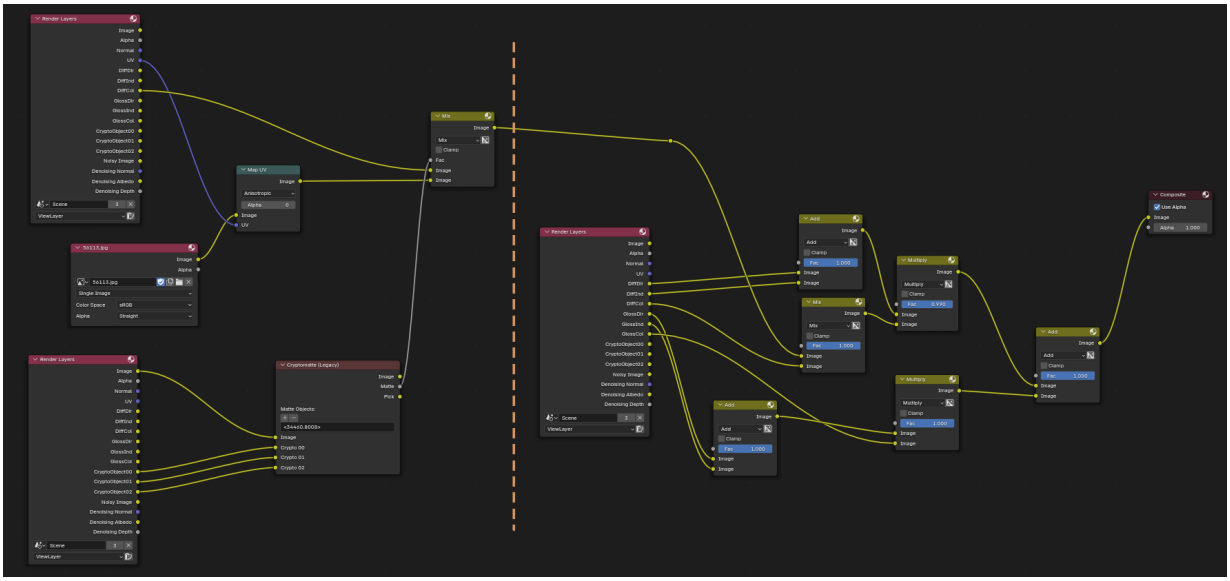
Figure 4.7: Material nodes for the scarf

single texture is applied to the model. The process of creating the UV map and creating the texture for the human model, is far more complex and it is imported, and was not created during this work. An alternative to swapping out the scarf texture in the material of the scarf model, it can be replaced in the Compositing editor. The advantage of this is per scene setting it only has to be rendered once and the scarf texture can be replaced afterwards without re-rendering. The Composting editor can be compared to post-processing an image with the benefit of having additional information what is in the image. This is a more complex step than before, as the properties (shadows, reflection, colors, lighting) that were calculated during the rendering for each object in the scene have to be recombined but with the other texture for the scarf. The node setup can be seen in Fig 4.8c. The left part of the node setup replaces only the scarf texture (Fig 4.9a) without considering any shading. Afterwards (on the right side in the node setup), the properties are recombined as seen in Fig 4.9b. The recombining is combining the shadows, reflection, colors, lightning using the information that were created during the rendering. During the process a few artifacts (Fig 4.9c) appear that would not happen during a rendering where the scarf is not replaced in the Compositing editor. Lastly, to ensure the scarf texture, the environment texture and the rotation of the environment texture can be altered during the rendering process, each node, that is responsible for the respective task is given a unique name to identify afterwards.



(a) Scene view

(b) Environment texture



(c) Compositing setup with replacing the scarf texture (Left of the orange separator line) and recombining shadows, reflection colors, lighting (Right of the orange separator line)

Figure 4.8: Automation setup (Part 1/2)

(a) Replaced scarf texture

(b) Replaced scarf texture with added properties

(c) Artifacts that happen during replacing the scarf texture in the compositing editor can be seen as purple/green jitter

Figure 4.9: Automation setup (Part 2/2)

```
Pipeline/
├── scenes/ # Scenes are stored in this folder (Blender files)
├── hdris/ # The environment texture are stored in this folder
│   (HDR Files)
├── images/ # The scarf texture are stores in this folder
├── output/ # The output folder where the rendered images are
│   saved to
└── pipeline.py # The pipeline created in python
```

Figure 4.10: Pipeline folder structure

4.3.2 Python Blender Automation Pipeline

The rendering pipeline is written in Python 3.11 utilizing the Blender Python Package (BPY [Ble25]). The basic structure can be seen in Algorithm 4.2, every component combination between different scenes, environment textures, scarf textures is rendered. To make the process expandable, a specific folder structure is presented (Fig 4.10). By adding the files into their corresponding folders, the synthetic dataset can be easily extended. In the output folder each category of scarves has its own folder to already prepare for the training afterwards. As mentioned earlier, specific nodes received a unique name to identify them more easily. As can be seen in Algorithm 4.3 and Algorithm 4.4, by replacing the scarf and environment texture the scene is changing. In addition to replacing to environment texture it also is rotated, this shows as the background looks different and the light sources emit from a different position. This is only a simple setup for a synthetic dataset. By adding randomization to various properties, the dataset can be diversified. How the final results looks can be seen in Fig 4.4.

Algorithm 4.2: Rendering pipeline for scenes with environment and scarf textures

```
Data: Scenes, EnvironmentTextures, Rotation angles, Cameras, ScarfTextures
foreach scene in Scenes do
    foreach envTexture in EnvironmentTextures do
        for rotation ← 0 to 360 step 90 do
            foreach camera in scene.Cameras do
                foreach scarfTexture in ScarfTextures do
                    if renderWithMaterial then
                        renderWithMaterial(scene, envTexture, rotation, camera,
                         scarfTexture);
                    end
                    else // replace in compositing editor
                        renderWithCompositing(scene, envTexture, rotation,
                         camera, scarfTexture);
                    end
                end
            end
        end
    end
end
```

Algorithm 4.3: Replace scarf texture

```
# Checks if the material is using nodes
if material and material.use_nodes:
    # Save all nodes from this material
    nodes = material.node_tree.nodes
    for node in nodes:
        # Checks if node names matches identifier
        if node.type == 'TEX_IMAGE' and node.name == "SCARF_TEX":
            # Replaces the texture
            node.image = new_img
            break
```

Algorithm 4.4: Set environment mapping rotation

```
# Checks if the scene environment is using nodes
if world and world.use_nodes:
    # Save all nodes from the scene environment
    nodes = world.node_tree.nodes
    for node in nodes:
       # Checks if node name matches identifier
        if node.name == "ENV_MAPPING":
            # Setting new rotation angle around the upwards axis
            node.inputs["Rotation"].default_value[2]
                                                  =  math.radians(angle)
            break
```

CHAPTER 5

Results

This chapter gives the results of the work. All models have been trained on a RTX 3090 and an AMD Ryzen 9 3900X.

5.1 Flat Scarf

The results for the traditional machine learning approach are presented in the following. There is a distinction made between the dataset with and without a background. As SVMs work with different kernels, one model is trained with a linear kernel and the other model with an RBF (Radial Basis Function) kernel. Table 5.1 shows that the model with an RBF kernel does not perform well, regardless of the dataset. For the first dataset both subsets, with and without background, do not deliver any good results. For the second dataset both subsets perform similar, but still with an too low performance. The linear kernel on the other hand performs much better. The first dataset, similar to the RBF kernel, does not perform well. The second dataset on the hand performs better. There is also a difference between the performance on the subdatasets as can be seen in Table 5.2. The dataset without the backgrounds reaches almost an accuracy of 40 % whereas the one with backgrounds only classifies 24 % of the data samples correctly.

The results for the deep learning approach can be seen in the following. In comparison to the traditional Machine Learning approach this method performs better overall as can be seen in Table 5.3. By fine-tuning the inner layers during the training process the accuracy is improved to 97 % compared to 57 % in the approach without fine-tuning. Freezing the inner layers had no effect on the training time. The minimal difference in the training can be explained as it stems from the hardware utilization.

Table 5.1: RBF Kernel results

Model	Accuracy	Precision	Recall	F1-Score	Time
SVM Dataset1 with Background	0.0	0.0	0.0	0.0	0.0 s
SVM Dataset1 without Background	0.0	0.0	0.0	0.0	0.0 s
SVM Dataset2 with Background	0.16	0.07	0.16	0.09	0.3 s
SVM Dataset2 without Background	0.15	0.07	0.17	0.09	0.2 s

Table 5.2: Linear Kernel results

Model	Accuracy	Precision	Recall	F1-Score	Time
SVM Dataset1 with Background	0.0	0.0	0.0	0.0	0.6 s
SVM Dataset1 without Background	0.06	0.00	0.03	0.06	1.8 s
SVM with Background	0.33	0.27	0.33	0.24	0.8 s
SVM without Background	0.39	0.41	0.39	0.38	0.3 s

Table 5.3: Neural network performance with different training strategies

Model	Accuracy	Precision	Recall	F1-Score	Time
NN with inner layers frozen	0.57	0.54	0.51	0.50	154m 10s
NN with inner layers fine-tuned	0.97	0.98	0.97	0.97	153m 28s

5.2 Rolled Scarf

The segmentation network was only trained on one class to detect scarves in general. In Table 5.4 the performance of this one class can be seen. Compared to the deep learning method for flat scarves, the training time only took 5m 5s. Compared to the flat scarf detection there are two additional metrics. The metric mAP50 (mean Average Precision 50) measures the precision where the Intersection over Union (IoU) is above 50 % between the prediction bounding box and the predicted bounding box. The metric mAP50-95 measures the average precision for the IoU steps 50 %, 55 %, .. 95 % and the computes the average of all these values. For a good model it is considered a mAP50 value above 0.5 and a mAP50-90 value above 0.3. The model does neither perform very well nor does it perform badly looking at the metrics. In Fig 5.1 and Fig 5.2 some results of the detection and segmentation model can be seen. In Fig 5.2a can be seen two bounding boxes that are not scarf. A possible adjustment to boost the performance could be to

broaden and diversify the dataset. One way to achieve this is to expand the dataset.

Table 5.4: Detection and segmentation metrics

Box-P	**Box-R**	**Box-F1**	**mAP50**	**mAP50-95**	**Mask-P**	**Mask-R**	**Mask-F1**
0.66	0.50	0.57	0.48	0.27	0.66	0.48	0.55

Compared to the flat scarf classification, the rolled up scarf classification performed worse than expected (Table 5.5). With an accuracy of 23 % the model is not stable enough to provide accurate results. Fine-tuning by adjusting the learning rate for different layers and training the model further did not result in improvements. The most probable cause for this low performance is the dataset. As it is synthetically generated, changing the scene and scarf to make it look more realistic could improve the performance.

Table 5.5: Classifier model performance

Accuracy	**Precision**	**Recall**	**F1-Score**
0.23	0.36	0.19	0.23

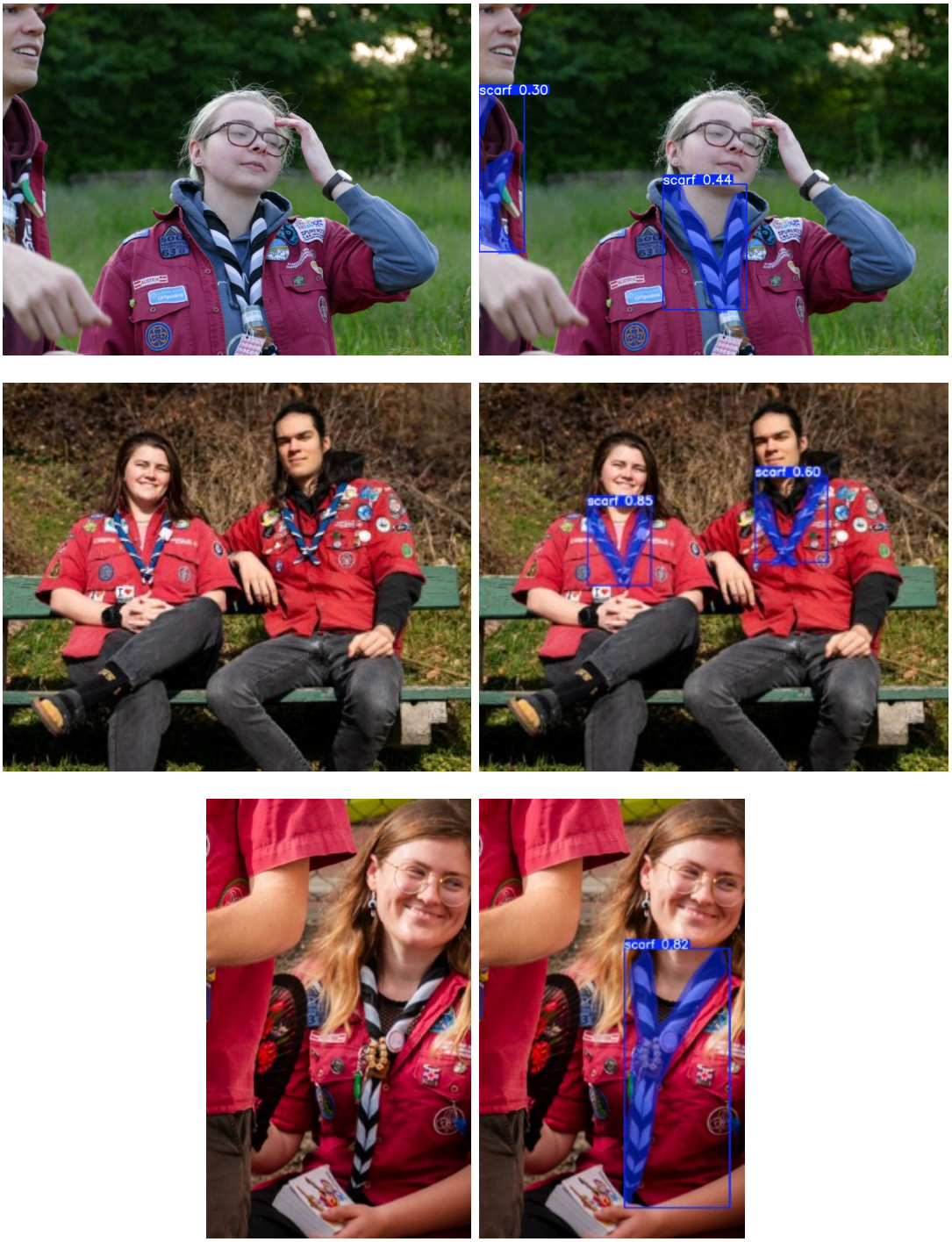


Figure 5.1: Image comparison before and after detected and segmented scarves. (Part 1/2)

(a) Example of detections that are not a scarf

Figure 5.2: Image comparison before and after detected and segmented scarves. (Part 2/2)

CHAPTER 6

Conclusion & Discussion

In this work the detection, segmentation, and classification of scarves in two forms were discussed. A triangle is a triangular piece of cloth that can be rolled up. There are a lot of different scarves out there. In Chapter 1 the scarf is described in more detail. The first form a scarf can appear flat which then is taking the shape of a triangle. For the classification, two approaches are explored that are described in more detail in Section 3.1. The first approach is using a traditional Machine Learning methods in this case a SupportVectorMachine (SVM). The training and testing data are obtained from real-life images by extracting hand-crafted features based on the images' colors. Although this method produces promising results that exceed initial expectations, it is outperformed by the second approach where pre-trained networks that were subsequently fine-tuned are used. This fine-tuning is done by not freezing the inner layers but adapting them during training, resulting in a network that is better suited to the task at hand compared to the first approach. In addition, the dataset is diversified with Data Augmentation. Even with a smaller dataset the network achieves desirable results. The implementation of both approaches are discussed in Section 4.1. The second form of scarves is when they are rolled up and worn around the neck. For this case (Section 3.2), two different models are trained. The first model is responsible for detection and segmentation of scarves in general. Sometimes, there are some artifacts found by the model that do not contain a scarf. The creation of the dataset for this model is very time-consuming, approximately 5 min per image in the dataset, as it has to be done manually. The second model task is to classify the scarves, that are detected by the first model. The architecture of this model is similar to the second approach in Section 3.1. In this case, the model delivered insufficient results as its accuracy did not even exceed 24 %. The most likely cause for this is the synthetically generated dataset (Section 3.3). Reworking the creation of the dataset (Section 4.3) and aiming for more realism could solve the problem.

Looking at the work that has been done and the results some of them yielded satisfactory results and others did not. In the flat scarf classification the traditional Machine Learning

approach performed worse and the pre-trained network approach performed better than expected. The performance of the traditional Machine Learning could be improved by adapting the feature extraction methods. Exploring different feature combinations could also be done in the future. The pre-trained network approach is a state-of-the-art approach as-is and performs accordingly. Trying different pre-trained networks to reduce the training time and dataset size are additional possible directions for future work. The detection and segmentation of the rolled up scarves performed better than expected. As it is a more complex task and the dataset is rather small it could have performed worse. The classification of the rolled up scarves was rather disappointing. Using a more current pre-trained model could help to improve the performance. It is likely that the synthetic dataset that is used as the base of the fine-tuning for the pre-trained network did not meet the expectations. In hindsight, instead of a digitally created scarf graphic taking a photo in which the flat scarf is fully visible could have improved performance as well. Other ways to improve performance could be by setting up the scene with more details in the area around the scarf. An option for this would be 3D photoscanning objects and people to enhance the realism aspect. It should be kept in mind that as the scenes get more complex the render time for each image increases.

Overview of Generative AI Tools Used

ChatGPT was used to research current technologies, to arrange images in the figures, create the table arrangements and to convert the python code into pseudo code.

Übersicht verwendeter Hilfsmittel

ChatGPT wurde verwendet, um aktuelle Technologien zu recherchieren, Bilder in den Abbildungen anzuordnen, die Tabellenanordnung zu erstellen und den Python-Code in Pseudocode umzuwandeln.

List of Figures

List of Tables

List of Algorithms

Bibliography

[Ado25] Adobe. Lightroom classic. https://www.adobe.com/at/products/photoshop-lightroom-classic.html, 2025. Accessed: 2025-08-18.

[Ble25] BlenderFoundation. Python package bpy. https://pypi.org/project/bpy/, 2025. Accessed: 2025-08-08.

[Coe25] Luis Pedro Coelho. Mahotas. https://pypi.org/project/mahotas/, 2025. Accessed: 2025-08-09.

[ea25] Travis E. Oliphant et al. Numpy. https://pypi.org/project/numpy/, 2025. Accessed: 2025-08-09.

[fBBPoG08] Robert Baden-Powell (first Baron Badon-Powell of Gilwell). *Scouting for Boys - A Handbook for Instruction in Good Citizenship*. C. Arthur Pearson LTD, London, Tower House, Hampton Street, first edition edition, 1908.

[Fou25] Blender Foundation. Blender. https://blender.org/, 2025. Accessed: 2025-08-10.

[GHQ18] Ming Gu, Kun Hao, and Zhiyi Qu. Flag detection with convolutional network. In *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*, pages 258–262, 2018.

[HSD07] Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 2007.

[Igl25] Vladimir Iglovikov. Albumentations. https://pypi.org/project/albumentations/, 2025. Accessed: 2025-08-09.

[JQ24] Glenn Jocher and Jing Qiu. Ultralytics yolo11. https://github.com/ultralytics/ultralytics, 2024.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[Man18] Vukašin Manojlović. Flagnet. https://github.com/iamvukasin/flagnet, 2018. Accessed: 2025-08-07.

[Sca24] Scarves. The scouts scarves directory. https://scouts-scarves.eu/, 2024. Accessed: 2025-08-07.

[sld25] scikit-learn developers. scikit-learn. https://pypi.org/project/scikit-learn/, 2025. Accessed: 2025-08-09.

[Tea25a] ImageNet Research Team. Imagenet. https://image-net.org/index.php, 2025. Accessed: 2025-08-18.

[Tea25b] OpenCV Team. Opencv for python. https://pypi.org/project/opencv-python/, 2025. Accessed: 2025-08-09.

[Tea25c] PyTorch Team. Pytorch. https://pypi.org/project/torch/, 2025. Accessed: 2025-08-09.

[Tea25d] PyTorch Core Team. Torchvision. https://pypi.org/project/torchvision/, 2025. Accessed: 2025-08-09.

[TL19] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[Ult25] Ultralytics. Ultralytics. https://pypi.org/project/ultralytics/, 2025. Accessed: 2025-08-10.

[wos25] World scout movement organization commitee members. https://www.scout.org/world-scout-committee-members, 2025. Accessed: 2025-08-09.

[wpp25] Wiener pfadfinder*innen. https://www.wpp.at/, 2025. Accessed: 2025-08-09.