Automated Shelter Recognition in Refugee Camps

AUGUST 2018

AUTHOR:

Nathan Lacroix

UNOSAT

SUPERVISORS(S):

Lars Bromley
Taghi Aliyev

CERN openlab

Abstract

In June 2018, more than 68.5 Million people across the globe were reported to be fleeing war or persecution. Within the United Nations, UNOSAT is the organ in charge of collecting demographic information based on satellite images of refugee camps to ensure reliable UN operations providing shelter, food and medicines to refugees and internally displaced people. This work aims at assisting UNOSAT analysts by providing them with an automated shelter detector to increase their efficiency and quality of the analysis.

The tool is developed in two phases. The first one consists of generating polygon masks of shelters using single point location generated by UNOSAT analysts in the past 10 years. The second uses the newly generated dataset to develop a model detecting multi-class shelters in new camp images without any analyst data.

This report describes the model developed for the first phase of the project. It consists of a fine-tuned Mask R-CNN conditioned on analyst point shelter locations. On average, the tool achieves a recall of 88% and a precision of 81% compared to a human annotated shelter. A user-friendly interface allows the usage of the tool with a command line one-liner. Its usage is summarized in the report and detailed in the Github Repository. The latter also contains the entire python code of the developed tool.

The results of all experiments are reported and the weaknesses of the model are discussed. Finally, possible further improvements and next steps required to achieve the development of the automated shelter detector tool are reported.

Contents

List of Figures

List of Tables

1. Introduction

In June 2018, the World Health Organization reported the highest number of refugees in Human History: more than 68.5 Million people across the globe are fleeing war or persecution [4]. Having left their home, a high percentage of refugees and internally displaced people (IDP) live in refugee camps. Due to the urgency of the situation, the lack of financial means, and the important influx of refugees, camps are often chaotic and poorly documented. It is difficult to keep track of (1) where refugees settle, (2) how many they are, (3) in what living situation they are.

In that sense, satellite imagery analysis is a very powerful tool which enables the collection of accurate, regular and up-to-date information about camps in a trustworthy and secure way. UNOSAT, a subdivision of UNITAR located at CERN, is the office responsible for providing this type of information to the United Nations (UN). It consists of a team of highly trained analysts which analyze high resolution satellite images to respond to humanitarian disasters and provide relevant information for the UNs on field operations. UNOSAT is not limited to collecting data about refugee camps, it also responds also to other humanitarian disasters such as floods and damage assessment in conflict zones. In the case of refugee camps, analyst count and mark shelters locations. Such information help evaluate objectively the population distribution of the camp (and its evolution) and can be a usefull proxy to determine how many refugees live in the camp. In turn, such numbers are highly valuable for UN field operations. For instance, to send the right amount of equipment, medications, and sanitary installations to locations which need it most.

Nevertheless, the steady increase of refugees over the last 15 years has led to a challenging situation for UNOSAT. Namely, it has become intractable to keep up with the data requested by the UN using traditional analysis methods. The project described in this report has the pragmatic objective to increase UNOSAT's throughput by alleviating analysts' workload, as well as providing additional valuable analysis data which cannot be collected by traditional methods. The rest of the report is structured as follow: first, the issue with current manual tools for analysis is explained, challenges for automation are reported and the scope of this project is specified. Next, a short litterature review of automated shelter detection is provided as background. Thereafter, the method of the envisioned approach, with its different models and software/hardware requirements, is described at length. The different undertaken experiments are then reported, followed by their results. Fulfilment of the defined goals for the project are additionally discussed. Finally, further research directions are provided. A short conclusion at the end summaries the different findings and core points of this report.

2. Problem Statement

Camps often consist of more than 10 000 shelters and are sometimes analyzed several times to monitor their evolution. Despite providing a very high quality output, manual analysis is time consuming, cumbersome and expensive. Indeed, an analysis can easily take up to several days for a single camp. Using a software called ArcGIS, an analyst inspects each region of a high-resolution satellite image and marks each refugee tent with a dot in the center of the tent. Sometimes, additional information is also recorded, such as the type of shelter or the satellite image it was taken from. This data is then stored in a shapefile vector layer which encodes the geocoordinates of each shelter along with the information associated to it.

Automated shelter detection can help reducing drastically the time required for each analysis. In the following subsections the requirements for such an automated analysis are described, followed by a summary of the difficulties associated with remote sensing for the development of such an automated process. Finally, the proposed pipeline for the development of the tool is presented and the scope of this specific project within this pipeline is defined.

2.1 Objectives

In 2013, Tiede et al. published a list summarizing the requirements for the adoption in practice of an automated shelter recognition system [13]. The key points included : (1) the ability to use the algorithm on different camp images taken with different sensors, (2) the ability to implement and deliver high quality performance in real-world and real-time scenario's, (3) The ability to validate results, especially reflecting end-user's needs. Being a direct collaboration with UNOSAT analysts, the aim of this project is to ensure the above-mentioned criteria are fulfilled such that the algorithm can be used for operational purposes. Hence an expanded and more specific list of requirements and objectives is defined below for this project, based on Tiede's critiria:

- The system can detect tents and refugee shelters with a recall and precision higher than 90% in different camps using different satellite sensors.
- It should be capable of identifying different types of tents: eg. refugee shelter, administration building, etc.
- The system should be able to run with highly flexible input image types. I.e. with images of sizes up to 6-8 GB, with different number of bands (1-8), and using dataformat ranging from uint8 to unit32.
- The tool should be able to provide output in shapefile format (.shp); the format used in the

remote sensing community.

- The software must be embedded into an easy-to-use tool for UNOSAT analysts, which do not possess prior knowledge of command line and software engineering.
- Due to funding limitations, the algorithm should be runnable on desktop computer (possibly with 1 GPU of type Nvidia GTX 1080 or similar graphic card).

2.2 Field Challenges

There are many challenges associated to the field of remote sensing which complicate automation. These difficulties also partially explain why, to the authors best knowledge, no automation tool satisfying the above-mentioned criteria has been developed until now.

2.2.1 Data Scarceness

High resolution satellite images are expensive, submitted to strict licenses and hard to obtain. UNOSAT has access to several databases of satellite images. Nevertheless, the total number of refugee camp images possessed by UNOSAT is estimated to be between 100-150. Each camp image can cover an area of 2-25 km^2 englobing the entire refugee camp and its neighborhood. For many automation techniques such as machine learning and deep learning, 100 to 150 images is very few and hardly enough to obtain the aforementioned performance.

2.2.2 Resolution

Indeed, satellite images come with different resolutions, depending on the satellite sensor they were taken with. For the considered application, resolution of images considered vary between 20cm and 80cm. Refugee tents can be of sizes down to 1.5 x 1.5 m, which thus translates to 2x2 pixels in the worst case. It is understandably difficult to develop a tool which can detect such small objects in images with high precision and recall.

2.2.3 Band Variety

Similarly to ordinary images, satellite images are composed of different spectral bands. Depending on the satellite, the image can be monochromatic (that is, the image has one band and looks like a black and white image), Red Green Blue (RGB), or multispectral. In the latter case, images can have 4 to 8 different spectral bands, such as the Near Infrared (NIR) band, etc.

Just like objects are of different colors in the RGB spectrum, they can different response to other spectral bands. For example, vegetation tends to reflect NIR while a metal structure may absorb

Figure 2.1: Illustration of the shelter variety in terms of shape, size, color, resolution.

it. Thus, in principle, the more bands an image has, the better, since it conveys more information. Nevertheless, not all images have the same number of spectral bands, and there is no standard into which part of the electromagnetic spectrum the band should cover. So even among images with the same number of bands, these bands do not per say correspond to the exact same part of the electromagnetic spectrum. In addition, no neural network has been found to be pretrained for images with more than the 3 usual spectral bands (R,G,B). This makes it challenging to take advantage of the extra information encoded in the bands, or even to develop a tool which can deal with varying number of bands.

2.2.4 Shelter, camp and image variety

As illustrated in Figure 2.1, tents from different camps can be very different in size, color and shape. Some tents can be as small as 2x3 pixels (approx. 1.5x1.5m) while others are in fact buildings which span areas of hundreds of pixels. Colors can vary from bright white (eg. UN disaster relief tents) to dirt brown (shelters made out of dried mud). Shelters can be rectangular, oval shaped, circular, and combinations thereof.

At a camp level, some are very dense and chaotic whereas others are spread over large areas with low shelter density. In addition, depending on the angle at which the satellite was when it took the picture, the time of the day, the season of the year, the exact same camp can look very different on two different images. An illustration of these principles can be found in Figure 2.2.

In addition to atmospheric interference, all these factors contribute to a high level of noise in the images, which further increase the difficulty of the task, even for human beings.

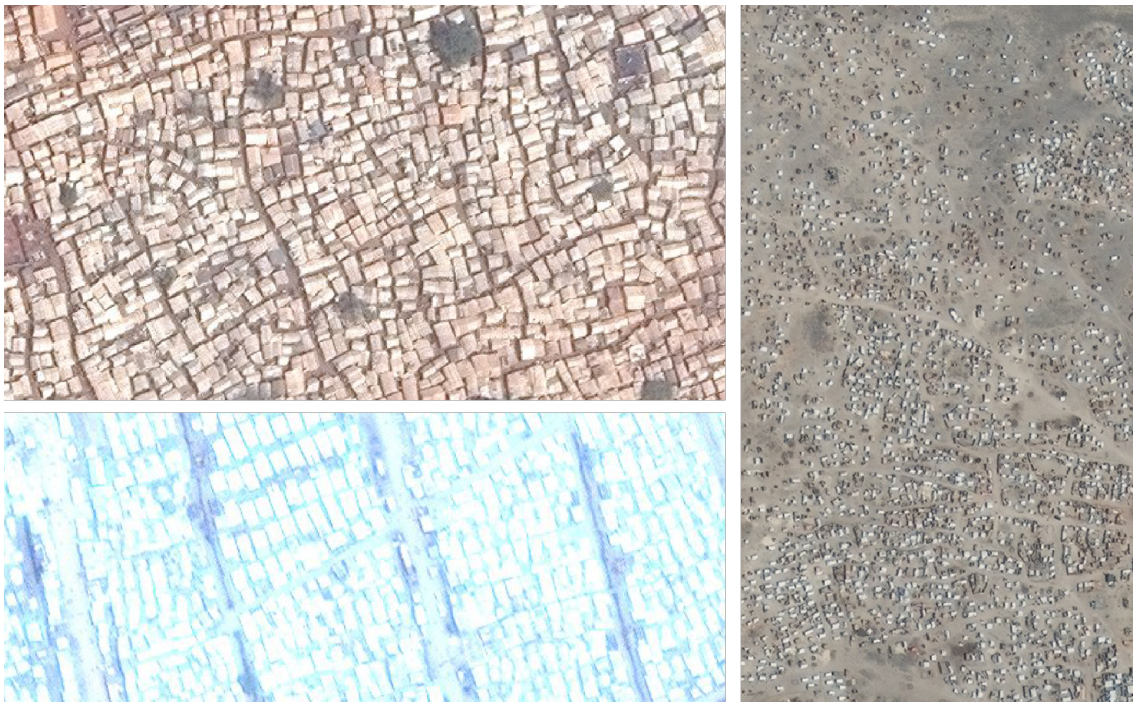

Figure 2.2: Illustration of the camp variety in terms of density, season, time of the day.

2.2.5 Multi temporal point data

Specific camps are mapped on a regular basis, and evolution of the camp is analyzed. This requires the comparison of several images of the same camp. To ease this process, images are sometimes slightly shifted through imaging processing techniques within the ArcGIS software. Nevertheless, the same shapefile is used to save the point location of the shelters. This results in point location which may be shifted from the original image. Hence the point location cannot be used anymore with the other (more recent) images of the camp.

2.3 Proposed Pipeline

To develop an automated shelter recognition software, a two steps pipeline is proposed. The process flow is illustrated in 2.3.

The first step, aka Point-to-Polygon step, consists in generating polygon masks for each shelter in camp images using the point data generated by the analyst. In other words, the algorithm seeks to determine the shape and area of the shelter using the point data as starting point or hint for the location of the shelter. This important intermediate step toward the final goal of a fully automated system is important in several ways. It provides a way to generate a high amount of training masks which can thereafter be used to train the final algorithm to have an end-to-end system. Indeed, the lack of training data is one of the major issues preventing the development

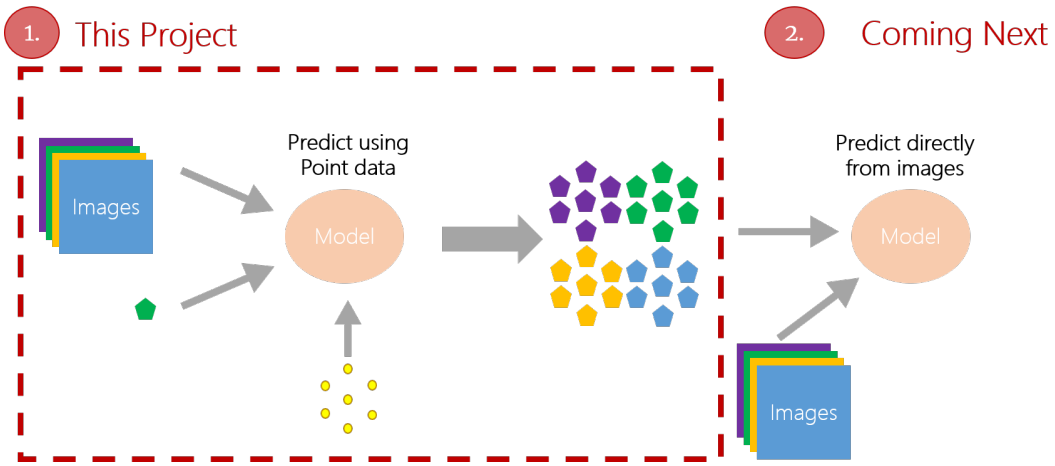

Figure 2.3: Proposed pipeline for the development of an automated shelter recognition system. The first step is the specific scope of this project. It consists in using the available database of point data (yellow points) with its corresponding satellite image and the very few available polygons to create more polygons (from which the name originated: point-to-polygon).

of a fully automated system right away. However, UNOSAT possesses a large number of images for which point data was generated over the last 10 years by analysts. By generating shelter masks for all the images in this database, a large training set can be generated for the second step of the pipeline. Nevertheless, the point-to-polygon algorithm is already very valuable for the analysts, as it provides a more accurate estimation of the shelter area; a valuable metric to the UN to estimate living space per refugee, etc.

The second step, aka end-to-end algorithm, consists in finding tents in new camp images without any point data. To this end, the large shelter masks dataset previously generated using UNOSATs point database is used to train, validate and evaluate the end-to-end algorithm.

2.4 Specific Scope of this Project

One of the most important steps of this pipeline is the first one, as no reliable end-to-end algorithm can be developed without a high-quality training dataset. Hence this summer internship focused exclusively on the development of the point-to-polygon algorithm. The objectives specific to this step are the following:

1. Within approx. 8 weeks, develop an algorithm providing a shapefile with shelters masks using as input point data and satellite image, with a minimum average recall and precision of 80 % with respect to human annotated shelters.

2. Make the algorithm as flexible as possible for different image types (uint8, uint16, etc.), size, number of bands, tiled images, etc.

3. Package the algorithm in an easy to use software for UNOSAT analysts.

Before discussing the method used to achieve these objectives, a brief literature review is provided in the following chapter. A summary of similar projects are reported and the state of the art using both classical machine learning and deep learning approaches are discussed.

3. Related Work

Automated shelter recognition in refugee camps is not a new topic. Nevertheless, methods used for shelter recognition have significantly evolved in the last 15 years. An overview of this evolution is provided in this chapter, along with analogous yet unrelated automation challenges.

In 2003, Giarda et al. published a paper on automated shelter recognition in Lukole refugee camp, Tanzania [7]. Giarda et al. investigated four different classification techniques; supervised classification (maximum likelihood classifier); unsupervised classification (ISODATA clustering); mathematical morphology based on shape and contrast (combination of image processing techniques including opening, closing, filtering, thresholding) and finally multi-resolution segmentation (weighted average of several factors to segment tents in both panchromatic and multispectral images). The latter two techniques achieved impressive results. Namely, spacial accuracy of approx. 85 % (omission and commission error below 15%) in the sampled areas and an overall shelter count in the camp with a statistical accuracy of 97%. However, these techniques all require significant tuning and a priori knowledge about the image (average size of shelter, shape, minimum size of shelter for post processing thresholding, etc.). Moreover, they are highly dependent on contrast and colors, shadow, which makes it almost impossible to use the same parameters for a different refugee camp. This lack of robustness has limited their effectiveness in practice and adoption amongst remote sensing analysts.

Three years later, Laneve et al. published for the first time an automation algorithm for shelter recognition applied on more than one camp image (Goz Amer, Lukole, Mille) [11]. The authors used a mathematical morphology analysis to extract the refugee shelters. Several algorithms were investigated, including using structual elements with defined shape and size; a combination of PCA to extract multispectral information with Watershed floodfilling segmentation. Achieved accuracy in the investigated sample areares is of approximately 80-90 % depending on the algorithm. This paper brought another innovation: namely the authors did not only comment on shelter count, but also reported their findings about the ”shelter mask” to estimate the area of each shelter, and how this can be used to better estimate the refugee population. This concept has been of great importance for one of the developed algorithm in this project and is further expanded upon in the following chapter. Nevertheless, the authors report themselves that the algorithms are very dependent on the fact that the chosen samples are ”well behaved”. I.e., these techniques remain difficultly usable in practice.

A more quantitative investigation of the robustness problem is described in [14, 12]. Their algorithm consists of a sophisticated pipeline including a combination of rules, edge detections, thresholds, filtering, etc. Authors report that although their automated algorithm detected ¿ 90% of the bright tents and ironed-roofed shelters, performance decreased significantly to ¡ 60% on

traditional huts and dust-/sand-covered tents.

In recent years, Neural Networks (NN) have attracted considerable attention due to their generalization capabilities and robustness for object recognition. In particular, Convolution Neural Networks (CNN) and more specifically mask-RCNN have gained wide acceptance for robust object detection and segmentation in every day life images [9]. They thus are ideal candidates to develop robust tools for automatic shelter recognition in satellite images. However, literature about the specific application of NN to shelter recognition is scarce. In fact, no publication has been found on this specific topic at the beginning of this project (June 2018). Nonetheless, several similar issues are illustrated in the literature.

In 2017, Guirado et al. published a comparison between CNNs and Object Based Image Analysis (OBIA) for shrub detection in Google Earth satellite images [8]. OBIA is a set of standard tools using algorithms similar to the previously described algorithms, and additional segmentation algorithms such as k-nearest neighbor, support vector machines, etc. Using an improved ResNet-based (a specific CNN architecture), they were able to achieve F1 scores of 96.5 and 93.4% compared to 92.9 and 77.3% respectively for the best performing OBIA model.

More analogous to this project, crowdAI – a plateform hosting Artifical Intelligence challenges – organized a house mapping challenge from satellite images, which ended at the same time as this project (August 2018) [1]. The winning solution achieved approx. 94% average recall and average precision at IoU $>= 0.5$, using an improved version of a U-Net, along with data augmentation.

Although houses are much bigger and easily identifiable than most refugee tents, it can be hypothetized that NNs can achieve high performance in automated shelter recognition. In addition, their robustness and generalization capability are promising factors which will enhance their potential transfer to different camps and sensors. Hence NNs are the main investigated technique in this project, although a classical unsupervised method has also been explored initially, seeking to take advantage from the point data. In the following chapter, the pipelines and methods used for the development of the point-to-polygon software are describes in length.

4. Methods

The concept behind the point-to-polygon algorithm is summarized in Figure 4.1: Using the point data collected by the analyst to predict the shape (polygon) of each corresponding shelter. The concepts behind the engineered pipeline to achieve this is presented in the following section. Thereafter, software and hardware tools and packages are described.

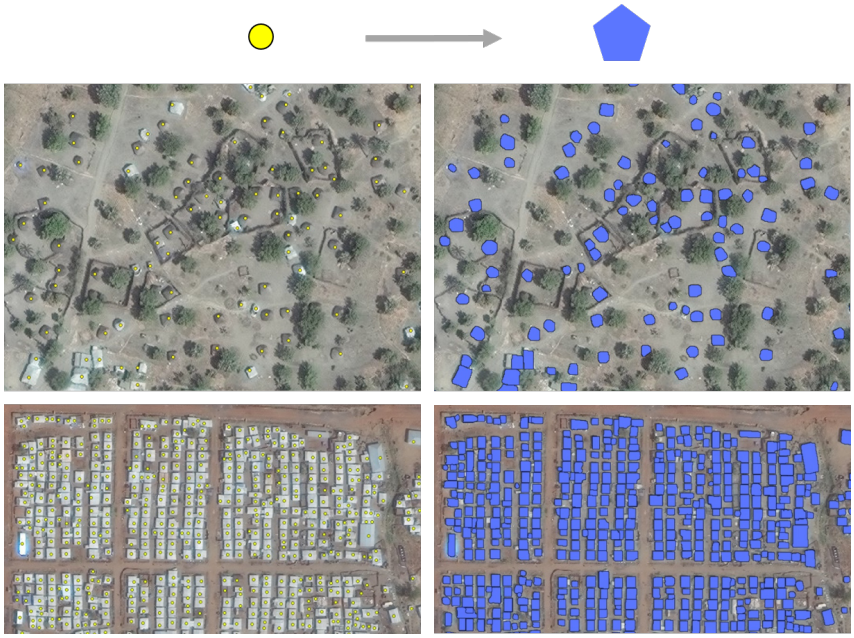

Figure 4.1: Summary of the concept behind the point-to-polygon algorithm and software tool.

4.1 Pipeline: Concepts

A high level schematics of the pipeline is presented in Figure 4.2. It consists of three major steps:

1. **Image preparation** : This step consists in going from raw camp images to processable images for the algorithm.
2. **Model** : The core algorithm chosen to predict the polygons from images.
3. **Predictions post-processing** : This step takes raw predictions and transforms them to usable data for analysts seeminglessly.

Each step is detailed below for a thorough understanding of the pipeline.

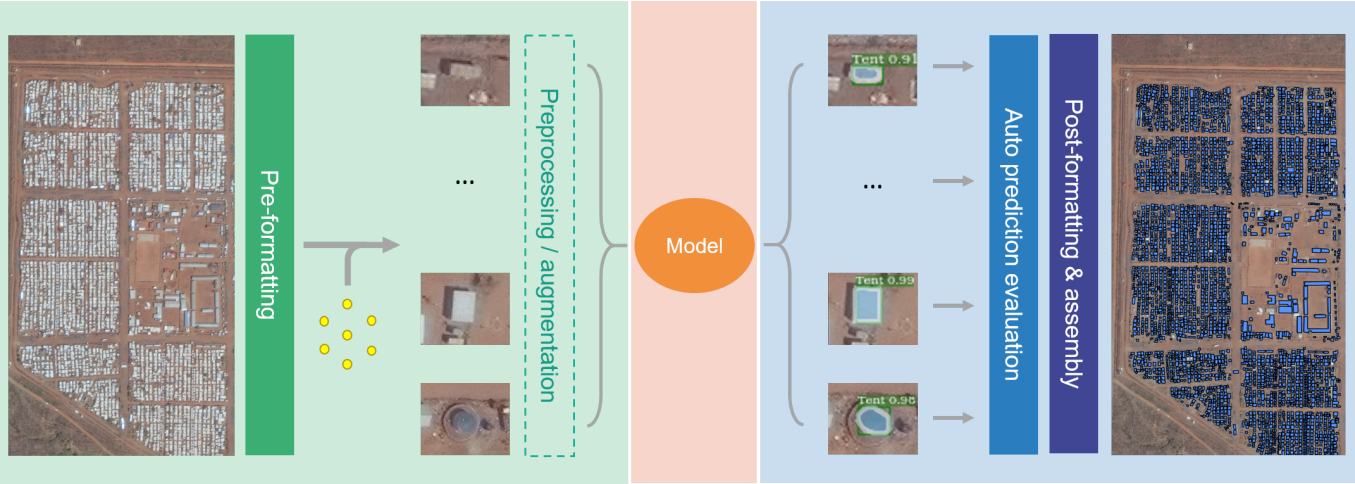

Figure 4.2: Point-to-polygon pipeline. First, images are prepared (green) from raw camp tiles / images and point-data information is encoded to help the algorithm. Next (orange), images are fed one by one in to the core algorithm, which predicts a polygon for each shelter. Finally (blue), predictions are assembled and formatted to a single .shp file. For more details, cf. subsections 4.1.1, 4.1.2, 4.1.3.

4.1.1 Image preparation

The first preparation step is preformatting, i.e. transform the raw satellite image type to a processable image type for the model. High resolution satellite images are usually stored as GeoTIFF files with the file extension .TIFF or .TIF (stands for Tagged Image File). This flexible file type enables lossless compression of high quality rasters along with information about the image (called metadata).

An important attribute of the metadata for GeoTIFF images is the CRS (Coordinate Reference System) along with the GPS coordinates of the image itself. Indeed, many different coordinate systems exist and it is important to know which one is being used when superimposing the image for example with vector layers (eg. point-data). Therefore, CRS have to be checked and images reprojected if different CRS are used for point-data and satellite images.

Additionally, the metadata lists the number of rasters (bands) in the image. This information is crucial to develop a flexible algorithm allowing images with different numbers of bands at the input yet predicting polygons correctly. Consequent band reduction / expansion must thus occur to ensure all images given to the model have the same number of bands.

Finally, unlike every day life images which are usually stored in uint8 data type (the intensities in each band vary between 0 and 255), satellite images can be store in many different datatypes such as uint8, uint16 and uint32. Unfortunately, common python image processing packages such as openCV do not always handle these datatypes. Moreover, computation is much more expensive when using higher resolution datatypes. Adequate datatype handling must thus occur to set all

images to the same data type. In this process, several streching techniques can be used to utilize the entire spectrum of the datatype.

Thereafter, one needs to choose how the point data information can be encoded/ given to the algorithm to ease the prediction process of each tent. This process is of course dependent on the algorithm itself. Therefore, this step is explained further for each of the models reported in subsection 4.1.2.

Next, satellite images are usually very large in size. They can be millions of pixel wide (up to 6-8 GB in file size). Processing these images in one step is simply not possible in terms of RAM memory. An additional tiling process is thus required before feeding images into the model. Great consideration has to be taken in this process as tiling the image can result in tents being devided over 2 to 4 different tiles. Not only can this make the task more difficult for the algorithm, but it also makes the reassembly of the predictions in one single image more difficult (necessity of merging semi/quarter tents, etc.).

A rather intuitive approach is to have a sliding window with an overlap between two subsequent windows, and where predictions are ignored near borders. A simpler approach is to process shelters one by one, by creating one mini-tile centered on each shelter of the image (cf. Figure 4.2) and telling the algorithm to predict the polygon for the shelter on which the mini-tile is centered. This can be done as point data is provided for each shelter in the image. This method has two advantages. On the one hand, it solves the assembly problem: each shelter is predicted individually and they can easily be reassembled in one filed. On the other hand, it is an implicit way of encoding the point information about the location of the shelter for the algorithm ; i.e. there will always be one shelter to predict and it will always be in the center of the tile.

Finally, once images are tiled, they can be preprocessed (eg. contrast enhancement, etc.) and/or used for data augmentation depending on the chosen model (more information is provided below). The images are then ready to be fed to the predicting algorithm.

4.1.2 Model

Two types of algorithms have been investigated.

Firstly, traditional unsupervised imagery algorithms have been applied to find the tents. These algorithms have the advantage of having a low computing complexity, and therefore can be performed on any laptop with low computation capabilities, at a very high processing speed and without any training. Nevertheless, the downside of these algorithms is that they require lots of manual tuning and are poorly generalizable. In fact, this is the issue which led to the focus on another method.

The second type of investigated algorithms are CNNs. These show properties which are opposite

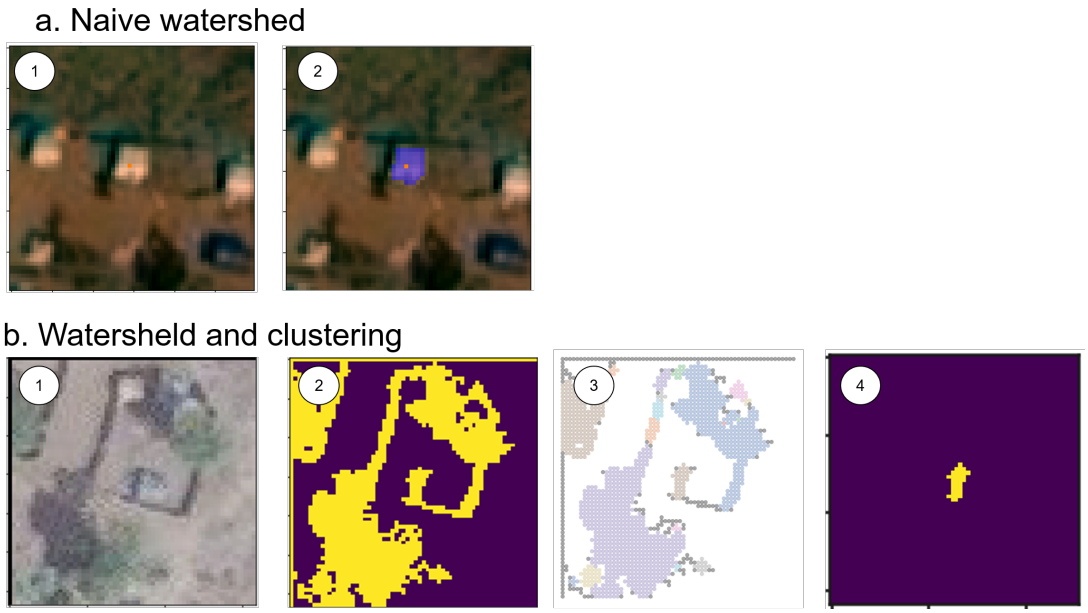

Figure 4.3: Illustration of predictions made using unsupervised segmentation algorithms. (a). Naive watershed prediction of the center shelter on the tile. The prediction is accurate. (b). Combination of watershed and clustering algorithm to improve noisy predictions: the noisy prediction generated in step (2) is passed through a clustering algorithm (3) and only the seed cluster is kept (4). Prediction is still far from perfect yet better than before clustering.

to the aforementioned method: they have high computational cost and require GPU training. However, they can generalize to different intensities, tent shapes and colors much better than their traditional counterpart.

Below, the two types of algorithms and their implementations are discussed in further details.

Unsupervised Algorithms for Shelter Prediction

The simplest unsupervised algorithm for image segmentation is thresholding. Nevertheless, it often requires very fine tuning in satellite images due to the high level of noise, the different colors involved, and the different satellite sensors utilizing different stretches of the electromagnetic spectrum. Hence, more sophisticated algorithms were chosen to tackle the segmentation of the refugee shelters.

First, a watershed segmentation is applied to each tile. Watersheld segmentation is a flood filling algorithm first introduced by Beucher et al. [6]. The main idea behind it is the following: local minimas in the image are taken as starting points (i.e. seeds or sources) for image segments, and flood progressively the entire relief of the image. Barriers are then drawn where the different growing segments meet.

The major difference between the traditional watershed algorithm and this implementation is the choice of the seeds. Instead of using the computed local minimas, which are often subject to high level of noise, or not always present in a small refugee shelter, the point data provided by the analysts are taken as seeds for the shelters. This use-case specific modification of the algorithm:

1. owers drastically the level of noise in the detections,
2. guarantees to have a segment for each identified shelters,
3. enables the separation of difficultly distinguishable neighboring shelters, since 2 or more seeds will be used.

Examples of watershed predicted shelters are provided in Figure 4.3.

Nonetheless, as visualized in Figure 4.3 (b), each individual segment can still be subject to noise. Hence a clustering algorithm (in this case, DBScan) is used as filter to select only the main cluster including the seed, to ensure compactness of the prediction. This method has been prefered to more the simpler ”opening” technique, as the atter made the prediction often disappear when acting on the many very small shelter (3-8 pixels).

Mask R-CNN for Shelter Prediction

The shelter segmentation using neural networks was performed by doing transfer learning from object detectors to learn segmenting refugee shelters. Indeed, transfer learning is adequate for the project as it requires less training and less labelled images than when training a network from scratch.

Nevertheless, transfer learning also adds constrains to the model. For example, it complexifies the architecture changes to the model. For instance, the number of input channels. Most pre-trained networks (including the one used) are trained on standard RGB images, or B/W images which are then transformed to RGB equivalent images. The network thus has 3 input channels. However, as explained above, satellite images can have up to 8 different channels, all of which contain relevant information to the detection of shelters. Navely adding many branches to the network results in unpredictable and often unstable networks, as too many weights are retrained from scratch. Moreover, the number of input channels varies with the image. And while transformations exist to convert B/W single channel to RGB and vice-versa, transformations to higher dimensions is not trivial, if even possible. Therefore, it was chosen to remove any additional channel and keep only the 3 standard RGB channels for each image.

For compatibility issues with the parent project, the Detectron framework was used to this effect. As backbone, a Resnet 50 with FPN has been chosen as it delivered a good tradeoff between performance and required training time.

4.1.3 Post-processing

Once tents have been predicted on each individual tile, they have to be reassembled into one single .shp file to satisfy end user's needs. Before performing this formatting step, the point-to-polygon software performs an evaluation upon its own predictions. Interestingly, although predicting the shelter correctly is not always simple, the assessment of the quality of a prediction can be easier.

For instance, predictions which are too small (eg. fewer than 3 pixels) or too big can be labeled as suspicious predictions. More elaborate tests, such as the Polsby-Popper test can also be performed. Originally developed to assess the degree of gerrymandering of political districts, the Polsby Popper gives a score between 0 and 1 for the compactness of a shape [3]. The score is obtained as follow:

$$PP = \frac{4\pi \cdot A}{p^2} \tag{4.1}$$

where A and p are the area and respectively the perimeter of the shape. Indeed, shelters are expected to be compact, such that shelters with a low Polsby-Popper score can also be considered as suspicious.

Furthermore, as shelters are predicted individually, overlapping areas can arise when merging the predictions into one single shapefile. When the overlapping area is big relatively to the size of the prediction, it is likely that several shelters are included in the overlapping predictions. Although not implemented in the first version of the project, detecting these cases and finding adequate splitting procedures can help minimize overlaps. A simpler approach is just to label these predictions as suspicious.

Finally, all suspiciously labeled predictions can be reviewed by the analyst in the first place, which can then correct the majority of bad predictions more efficiently.

4.2 Hardware

The unsupervised model was run on a simple TL desktop computer with 8GB RAM, 4 CPUs and ubuntu 16.04 as OS. With this device, approx. 430 shelter tiles (64x64 pixels) can be predicted per minute (7.2/second). Hence a camp image like Doro_20171111_PL with approx. 12 750 tents can be processed by the algorithm in under 30 min (for the most sophisticated version of the algorithm). Note that no emphasis was put into optimizing run time, and that the algorithm can surely be optimized to reduce drastically prediction time. One also has to remember that no training is required for unsupervised algorithms.

The mask R-CNN model was run on 2 Nvidia GTX1080 GPUs from the CERN TechLab Grid. The GPUs had respectively 8 and 10 GB RAM Memory and ran under CentOS 7. The model has also been tested on a Desktop Nvidia GTX1070 GPU with 10 GB RAM under CentOS. Inference

rate on single GPU (once the model is trained) is about 3000 tiles / min (50 / second) such that shelters in Doro_20171111_PL can be predicted in under 5 minutes. Note that although some frameworks allow inference using CPU only, it is *not the case of Detectron*. That is, Detectron requires GPU even for inference.

4.3 Software

4.3.1 Requirements

Point-to-polygon is coded in python 2.7. The software can also be called using python 3.6, as long as python 2.7 is installed and accessible under the command python2. It is strongly advised to use a virtualenv for the python packages related to the point-to-polygon software. A list of the packages, as well as a detailed installation procedure can be found in the Github repository under requirements.txt and doc/ installation .md respectively.

For the mask R-CNN, a modified version of the Detectron Framework developed by Facebook AI was used. The modified version is also included in the point-to-polygon repository. Detectron relies on caffe2, an AI library, which has to be installed separately. Good luck in installing it from source: it is a mess!

4.3.2 Usage

All required information about the software can be found in the Github repository. Instructions to use the point-to-polygon software are also provided. In short, as long as the correct packages are installed and the directory structure abode for, polygons can be generated from raw images using the following one liner in the terminal:

```
# launch the following command from the base directory:
python2 polygon_maker.py <base-dir> <output-dir>
```

Where <base-dir> is the main working directory where all the scripts are stored and <output-dir> the directory in which you want to store the predictions. In addition, the images needed to be processed should be specified in the configs/config_dataset.yaml, otherwise all images will be processed. To get information about which additional flags can be used, type python2 polygon_maker .py -h.

5. Experiments

At first, this chapter provides details about the datasets and parameters used for training, validation and testing. Various statistics are provided about satellite sensor type, tent categories, etc. Next, a description of the noticible experiments is provided. Finally, the metrics used for evaluation are explained.

5.1 Dataset

The UNOSAT database regroups circa 100 refugee camp images, collected from approximately 25 different camps. Nevertheless, only 12 of these images have polygonized shelters which can be used for training and evaluation. 8 were used for training, 2 for validation, 1 for testing and the remaining one was just used to control visually as it did not have corresponding point data and hence could not be used in the pipeline. A more detailed description of the training, validation and test sets is provided in Table 5.1. Statistics about the sets are displayed in Table 5.2. In the latter table, structure types gives an idea of the different classes UNOSAT gives to the structures found in refugee camps. The ideal end-to-end shelter detector should thus be able to recognize the different shelter types. Nevertheless, in this project all structures are considered as part of the same class.

A longer list of dataset has been created and can be found in the point-to-polygon archive under `dataset>created_datasets`. There are not available in the github repository. They were generated using the script `create_cropped_dataset.py` and its corresponding config file: `configs>config_dataset.config`. This script also generates coco-style annotations which encode shelter information (such as geocoordinates for remapping to the shapefile), and ground truth of the corresponding polygon for evaluation when available. Note that, since polygons and point data were created independently and by different humans, some points do not have a corresponding polygon, and vice-versa. In fact many shelter lacked polygons. Either because the point was out of the shelter and therefore does not fall within a polygon, or because not all shelters have been polygonized in the image. The script skipped all shelters in this situation. The numbers displayed in Table 5.1 and used thereafter are thus the successfully annotated shelters *only*. More details about the creation of the various datasets can also be found in logs under `logs > create_cropped_dataset.*.log`.

Note that the ground truth itself is sometimes of poor quality. This poor quality is caused by (1) badly annotated shelters due to human laziness, (2) shelters cropped on screens (tiles) of human annotators, (3) shelters difficulty discernable even for humans, and finally (4) intrinsically due to

Table 5.1: Images used in the training dataset. Hyperlinks are given to the locations of the images and corresponding point data.

Image Name	Image Date	Satellite	Vector Point	Shelters
Training Set				**53 028**
Yida_6June2017_w3	June 6th 2017	WorldView 3	Yida	10 305
Doro_20171111_PL	November 11th 2017	Pleiades	Doro 1	8 111
Doro_20131214_WV02	December 14th 2013	WorldView 2	Doro 2	11 831
Nyal_20170108_WV03	08 January 2017	WorldView 3	Nyal	4 191
Wau_20161215_WV03	15 December 2016	WorldView 3	Wau	3 415
AjouonTh_01Apr17_WV1	01 April 2017	WorldView 1	Ajouon	12 501
Muna_20160905_WV3	5 September 2016	WorldView 3	Muna	2 176
Ngala_9nov2016_w2	Nov. 9th 2016	WorldView 2	Self made	498
Validation Set				**12 756**
Ganyel_20170108_WV03	08 January 2017	WorldView 3	Ganyel	2 910
Juba_08Feb2017_WV03	Feb. 8th 2017	WorldView 3	Juba	9 846
Test Set				**2 518**
HTCCamp_20161226	Dec. 26th 2016	WorldView 1	HTCCamp	2 518

the small size of the shelter, a slight difference in pixels can induce a significant difference in metrics. It is thus important to remember performance is negatively biased.

Table 5.2: Detailed description of the characteristics of the training, validation and test sets. Percentages indicate the fraction of the dataset which has the characteristic described in the left column.

Characteristic	Training	Validation	Test
# shelters	53 028	12 756	2 518
# camps	8	2	1
Satellites			
WV1	23.6 %	-	100 %
WV2	21.2 %	-	-
WV3	39.9 %	100 %	-
PL	15.3 %	-	-
Structure Types			
Tent Shelter	90.7 %	78.5 %	100 %
Improvised Shelter	0.6 %	-	-
Semi-Permanent Structure	6.1 %	17.3 %	-
Admin Building	2.0 %	3.2 %	-
Metal Structure	0.6 %	1.0 %	-

5.2 Important parameters

All parameters used in experiments can be found in the config files and/or in corresponding log files. Important parameters which stayed constant in most experiences are the following:

- `crop_size`: 32. This is the number of pixels which are taken left, right, above and under the point data to delimit the size of the tile. For almost all experiments, tiles of 64x64 pixels were used. Note that this resulted in bigger buildings being sometimes cropped.

- `crs`: `EPSG:4326`. This is the Coordinate Reference System. It is used at several occasions, such as when reprojecting the predictions to a shapefile. The current version of the project does not fully support other Coordinate Reference Systems.

- `bands`: 1, 2, 3. This corresponds to the band selection when the image has more than 3 bands. The standard R, G, B channels are used. When the image had only 1 band, a transform to a RGB equivalent was performed.

5.3 List & Description of Experiments

The following list summarizes the different experiments and their purpose. Watershed experiments were run at smaller scale while the neural network experiments were run on the entire datasets.

1. **Baselines**:
   (a) **Square Tent Model**: The first baseline consists of assuming square tents centered at point data providen by analysts. The side length (15 px) was chosen by evaluating all possible lengths between 5 and 20 px and retaining the length which yielded the best scores.
2. **Watershed**
   (a) **Naive watershed**: Contrast enhancement as preprocessing followed by naive watershed algorithm with location point data provided by the analysts as seeds.
   (b) **Improved seeds & custering**: The watershed algorithm is very sensitive to the exact location of the seed. Nevertheless, humans do not always place the seed within the exact boundaries of the shelter, especially when it is small (4-20 pixels)! This results in completely wrong predictions since the seed of the segmentation is then located in the background instead of the shelter. In this version of the algorithm, a seed improvement algorithm is added. Starting from the point data, the seed is iteratively moved up the gradient of neighboring intensities to find the local maxima of the nearby shelter. This assumes tents are brighter than the background. Note that this assumption is taken throughout the watershed experiments. A clustering step is added at the end to diminish the noise in the mask output and thereby boost precision of the model.
3. **Neural Network**:
   (a) **Labeled All Instances**: Most straight forward application of transfer learning. The pretrained network is fine-tuned on refugee camp images where all shelters are labeled.
   (b) **Center Focused ResNet50-FPN**: Only the center shelter is annotated, such that the network learns to focus on the central part of the image. This also eases the fusion of the prediction in the final shapefile, avoiding having to fuse shelters separated on different tiles.
   (c) **Network Conditioning**: Investigating how the network can be conditioned by the data provided by the analyst to increase the precision and recall of the predictions. A relatively explicit way of encoding the information is simply to color each pixel corresponding to a point in the vector layer with a color that is very unlikely to be in the image (eg. bright red). The hypothesis is that it should help the network distinguish two neighboring shelters located near the center of the image. Another possibility is to feed an extra raster to each image at decoding time. Nevertheless, this introduces new weights which have to be trained from scratch and has thus not been performed here.

5.4 Metrics for evaluation

The use of the point data provided by analyst ensures an omission and commission error of 0% relative to human analysts. Therefore, these metrics are not appropriate for the investigated task. Rather, a pixel-level metric is required to evaluate the quality of the predictions, with respect to human polygon annotations.

To UNOSAT, the most important aspect is that no shelters / shelter-parts are left undetected. Indeed, shelters/ shelter areas are often used as proxy to evaluate the number of refugees in a camp. Therefore missing out on shelters leads to underestimating the number of camp inhabitants, which can yield devastating consequences. **Recall** is a good metric to measure whether predictions cover the entire area of corresponing shelters. It is defined for each shelter as followed:

$$R = \frac{prediction \cap truth}{truth} \tag{5.1}$$

In words, it corresponds to the intersection area between the prediction and the true shelter, divided by the area of the true shelter. Hence if the prediction covers the entire true shelter, intersection will be equal to the area of the true shelter and recall will be 1. Respectively, if no prediction is made the intersection will be 0 and thus recall will be 0. Average Recall (AR) corresponds to the recall averaged over all predictions.

Nevertheless, AR cannot be used as sole metric. Indeed, if a sufficiently large square is used as prediction for all shelters, the intersection with the ground truth will always be 1 and thus average recall will be 1 although the total area of the predictions is completely wrong! **Precision** addresses the latter problem; it corresponds to:

$$P = \frac{prediction \cap truth}{prediction} \tag{5.2}$$

Namely, the intersection area between the prediction and the true shelter, divided by the area of the prediction. Therefore precision is lower than 1 as soon as the prediction covers too much area with respect to the true shelter. Analagously to AR, Average Precision (AP) corresponds to the precision averaged over all predictions.

The goal is obviously to achieve both high precision and high recall for each shelter. To this effect, recall and precision can be combined in different ways:

- **F1-score**: this is the geometric mean of precision and recall. It is computed based on the following formula:

  $$F_1 = \left(\frac{P^{-1} + R^{-1}}{2} \right)^{-1} \tag{5.3}$$

  where R and P correspond to recall and precision respectively. Again, it can be averaged over all predictions to obtain the Average F1-score. The latter provides a good insight into the average performance of the model, where precision and recall are weighted equaly. An average F1-score of 1 corresponds to a perfect match between predicition and truth for all shelters in the dataset.

- **Intersection over Union (IoU)**: IoU corresponds to a more complex relationship between precision and recall but can also simply be seen as:

  $$IoU = \frac{prediction \cap truth}{prediction \cup truth} \tag{5.4}$$

  IoU is also qualitatively illustrated in Figure **??**. IoU is often used as metric in object detection. The difference between F1 and IoU is well explained here [2]. The main point is that

the average F1 score describes well the average similarity between predictions and ground truths wereas IoU is a better measure for the worst case similarity between predictions and ground truths.

6. Results & Discussion

The results of the aforementioned experiments are reported and discussed in this chapter. A summary of the results is to be found in Table 6.1. Visual examples of predictions using the best model are presented for Ganyel and Juba in Figure 6.1 and 6.2 respectively.

Table 6.1: Results of all experiments. Best scores for each metric are in bold. AP and AR stand for Average Precision and Average Recall respectively.

Experiment	AP	AR	F1	IoU
Baseline				
Square Tent Model (15x15 px)	65.1 ± 20.5	80.8 ± 18.0	67.8 ± 12.6	52.6 ± 14.0
Watershed				
Vanilla watershed	62.9 ± 33.5	68.8 ± 30.0	47.8 ± 25.1	38.3 ± 20.0
Improved seed & clustering	66.1 ± 30.5	68.9 ± 28.6	53.6 ± 24.2	44.0 ± 25.8
Mask R-CNN				
All Instances Labeled	$78.5 \pm 20..2$	89.1 ± 10.8	81.4 ± 15.2	70.9 ± 18.0
Center Focused R-CNN	78.2 ± 19.3	$\mathbf{89.7 \pm 10.4}$	81.4 ± 14.4	70.7 ± 17.3
Conditioned	$\mathbf{80.7 \pm 17.6}$	88.0 ± 10.8	$\mathbf{82.4 \pm 12.7}$	$\mathbf{71.7 \pm 15.5}$

The reported baseline already achieves a relatively high recall, meaning the square predictions cover usually most of the ground truth shelter. To obtain better F1 and IoU scores (representative for the average and bad performances of the model respectively), a higher precision has to be achieved.

The vanilla Watershed Model does not achieve better performance than the best baseline. In an attempt to increase its precision, a seed improvement mechanism and a clustering algorithm are added to the model. The former is meant to decrease wrong initialization of the watershed algorithm while the latter restricts the detection to the cluster in which the seed is located to limit the prediction to a relatively compact shape. The improvements are shown visually in Figure 6.3. Although the Improved Seed + Clustering Model performs slightly better than the Vanilla watershed Model, it still does not perform as well as the best baseline (Square tent model) on the validation set. Looking at performance separately for each camp of the dataset allows to conclude it is due to the poor generalization capability of the model. Indeed, the Watershed Model achieves +5%, +2% compared to the baseline in AP and IoU respectively in the Juba Camp. However, it collapses completely on the Ganyel shelters (AP: 54%, AR: 32%, IoU: 9.5%). Ganyel shelters are very difficult to detect for the Watershed model as the shelters are of the same color (material)

Figure 6.1: Conditioned Mask R-CNN predictions on camp Ganyel. Shelter density is relatively low, but shelter diversity is high: shelters are of different colors and shapes, some of which are very similar to the background color. The network can identify well sheltersin this environment.

Figure 6.2: Conditioned Mask R-CNN predictions on camp Juba. Shelter diversity is low, but shelter density is high. The network can identify well shelters in this environment, but sometimes fuses neighboring shelters together.

Figure 6.3: Clustering and seed improvement allow better performance for the Watershed Models. From left to right, camp image, vanilla watershed, improved seed & clustering watershed. One shall notice previously fused predictions are now separated in the improved watershed algorithm.

as the background (or even darker), while the model assumes shelters are brighter than their surroundings. By contrast, this brightness assumption is verified in camp Juba, where the model performs well. An example shelter prediction is provided for of each champ in Figure 6.4.

Scores of Mask-RCNN experiments are higher than watershed experiments. This illustrates the higher generalization capabilities of the RCNN compared to the Watershed Models. The first Mask R-CNN experiment (with all shelter instances labeled) already yields a significant improvement with respect to the baseline.

By reducing the attention span of the network to the center shelter, a slightly higher average recall (AR) is achieved (cf. Center Focused Mask R-CNN). Indeed, the network has then less trouble dealing with partial shelters at the sides of the image and learns better how to recognize the center shelter.

Average precision (AP) is not as high as AR. Indeed, most common mistakes of the network consist of fusing several shelters in the same prediction in densely populated areas. This principaly happens because shelters can be very close together and the network does not see the boundary between them. To mitigate this effect, hints are given to the network by providing the point data of neighboring shelters to the network. This type of conditioning (illustrated in Figure 6.5) indeed increases the AP, as well as the F1 and IoU scores, albeit reducing slightly the AR.

A complementary visual inspection of the predictions reveals the Mask R-CNN is sometimes even better than human annotators. This is illustrated in Figure 6.6. Hence the obtained results should be considered as ”worst case scenario”, as the human annotations themselves probably contain approx. 5-10% variations from real ground truth.

In high shelter density areas, or in areas where tents have an unexpected shape / arrangement, the Mask R-CNN has a tendency to fuse neighboring tents together. Human annotators can better distinguish the different shelters. Note that the overall estimated area by the Marsk R-CNN

Figure 6.4: Illustration of the poor generalization capability of Watershed Models. From left to right, columns represent the bare image, watershed predictions and mask R-CNN predictions. The first row camp Juba where tents are bright and easily discernable, and the second row is camp Ganyel where shelters are darker than their surroundings. While the watershed model works relatively well in Juba, it collapses in Ganyel. The most occuring scenarios are (1) too dark tents which yield no valley for the floods filling algorithm to fill (hence just the seed pixel is marked as tent) and (2) algorithm mistaking background as part of the shelter as they are of similar colors. By contrast, the mask R-CNN model performs well in both camps.

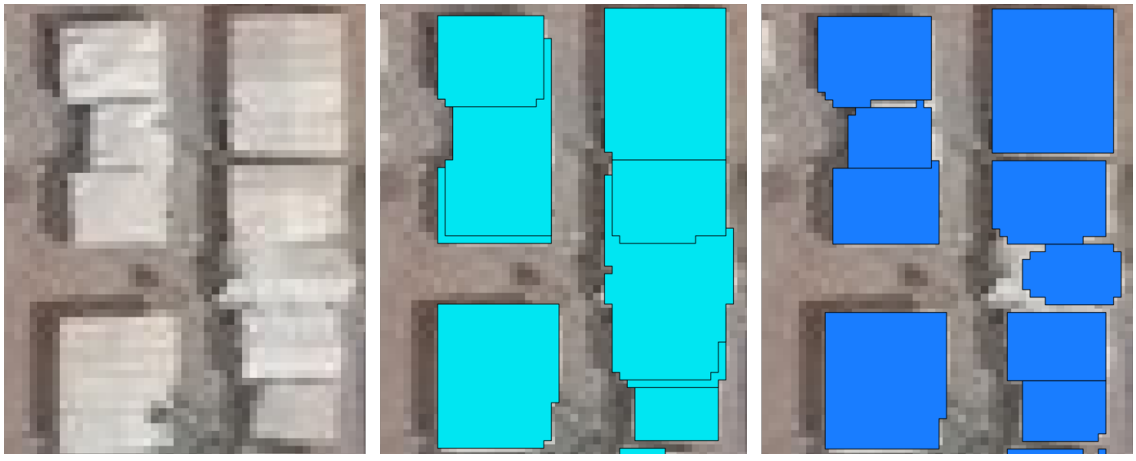

Figure 6.5: Effect of Network conditioning. From left to right, camp image, mask R-CNN without conditioning, mask R-CNN with conditioning. Indeed, the second network identifies better the boundary between the shelters, although the first one already approximately covers the total shelter area.

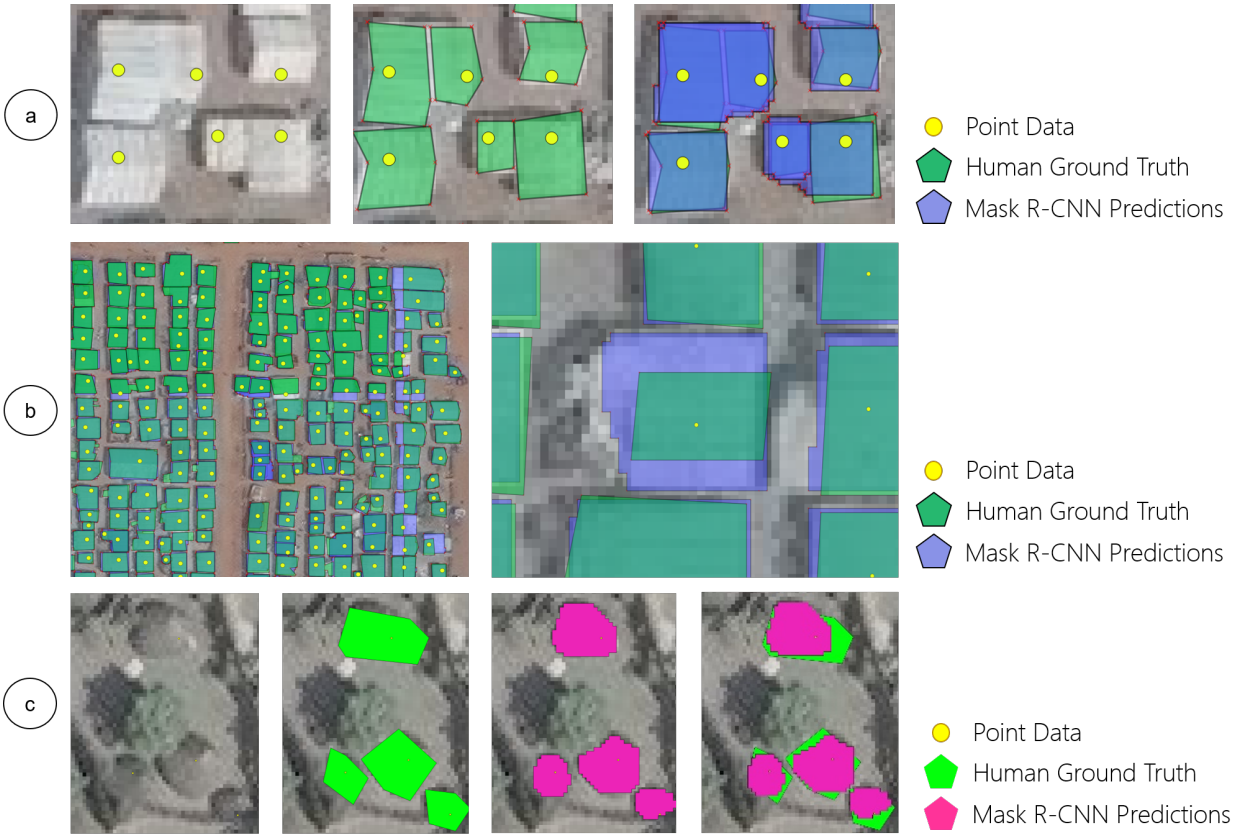

Figure 6.6: Comparison between human annotators and Conditioned Mask R-CNN Model. (a) Analysts are better at distinguishing close shelters. The Network sees only one shelter. (b) and (c) Visual inspection reveals predictions can be better than human annotated ground truth. Reasons include: human laziness (eg. (b), right), human annotation resolution (less vertices as predictions) (eg. (c)), cropped shelters on annotation tool (eg. (b), left.: both horizontal and vertical cropping lines are visible in ground truth).

remains very close to the sum of the shelters identified by human annotators. Again, this lead to a penality in scoring but does not cause major issues for real case scenarios where what matters most is the area of the shelters and not the number of shelters.

Practically speaking, if a first prediction round is made by the mask R-CNN and the UNOSAT analyst spends approximately 30 minutes per camp reviewing densely populated areas, starting with suspiciously labeled shelters, the UNOSAT ultimate target accuracy of 90-95% with respect to ”true ground truth” (not human annotations) can be achieved. The target of the tool itself (80% AP and AR) are in essence achieved.

Nonetheless, several improvements can be made to the model to increase further its performance and diminish the required review time of analysts. The following are detailed in the next chapter.

7. Further Improvements and Next Steps

This chapter lists envisioned further improvements to the UNOSAT Automatic Shelter Detection Project. First, improvements for the model itself are discussed. Next, several pipeline improvements are proposed. Finally, the concrete next important steps of the project are described.

7.1 Improvements

7.1.1 Model

- **Architecture**: A simple Mask R-CNN has been used so far. Nevertheless, other architectures should be tested. For instance, one could use a U-Net, which often performs very well in vision tasks when only scarse data is available. It is also the architecture of the winning solution of the Mapping Challenge of CrowdAI [10], which has a similar goal (building segmentation in satellite images).

- **Customized Loss Function**: The loss function could be tailored to the application. For instance, this application-specific loss should penalize more undetected small structures and penalize the fusing of neighboring shelters.

- **Band-flexible Input**: As explained above, satellite images often have more than the standard RGB channels (called bands in satellite imagery). These additional bands offer additional information which could be used by the network to detect the shelters better. In the long run, it would also help identify the different types of shelters (different materials with different scattering to different wavelengths reflect specific bands differently). Clever architecture designs could be envisioned to allow the usage and training on more than 3 bands if available. Indeed, UNOSAT has access to many satellite images with up to 8 spectral bands.

- **Data Augmentation**: At training time, it could help alleviate the scarceness of data. Simple augmentation technique such as sheer, brightness, saturation, flip and rotation could be used to generate more data on the fly. More sophisticated approaches, such as copy pasting shelters into another camp background image, could potentially help the network identify a shelter independently of the season, time of the day, inclination of the satellite, etc. At test time, flipping and rotating images followed by taking the mean of predictions often helps achieve better results, despite being a non-elegant technique.

- **Other Conditioning Methods**: Other methods using the point data could be used to con-

dition the network. As previously mentioned, one could add an additional channel with the point data. This could also be done in a separate branch at decoding time: one could provide the point data at each upsampling step of the decoder and force the network to predict the masks of the shelters on this branch. This method has the advantage of being interpretable as one would be able to observe the expansion of the shelters at each upsampling stage of the decoder. One could also use other methods, such as providing the number of tents to be detected (as a scalar) in the image (using the number of shelters detected in the point data), thereby forcing the right number of detections. This could help reducing the fusion of shelters.

- **Bootstrapping**: As observed during evaluation, prediction sometimes become more accurate than the human annotated ground truth. It could possibly improve performance to gradually allow confident predictions of the network to become the ground truth, and train further the network on those, in a bootstrapping scheme.

- **Self-supervised and Weakly Supervised Models**: Instead of using a fully supervised model where labeled data is the major bottleneck, one could use self-supervised or weakly supervised models. For instance, one could implement and adapt the MIST Framework. In this framework, a modified autoencoder is used for both detection and classification in an image [5]. A weak supervision is introduced by providing the network with the number of instances to detect, which is precisely what can be achieved using the point data. The advantage of this approach is that virtually all point data-annotated images can be used for this task, without need for ground truth polygons. This significantly increases the size of the available dataset. More over, images which previously could not be used because of the offset between point data and image could be used in this case as the precise localization of the points is not used. The encoder of this framework could then be fine-tuned on the ground truth annotated polygons.

7.1.2 Pipeline

- **ArcGIS Tool Development**: Transform the Automatic Shelter Detection into an ArcGIS extension would be an important step to ease the integration in the workflow of UNOSAT analysts.

- **CPU Version**: Up to date, the Detectron Framework used by the model requires a GPU, even in inference mode. Rewriting the model in a framework which can infer on a CPU would add a lot of flexibility to the tool. Another option is to have the tool run on a remote GPU. In that case, the appropriate interface should be developed.

- **Dynamic Cropping**: Up to date, the model evaluates shelters individually and the mini-tile size is constant for all shelters. This leads to an issue when the shelter is in fact larger/longer than the mini-tile. To alleviate this issue, a dynamic crop size could be established. Either

using the ground truth polygon if given, or if by writing the requested tile size in the coco-style annotation.

- **Additional Post Processing**: Several post processing steps could significantly improve the performance of the tool:

  1. Tag suspicious shelters: as mentioned in Section 4.1.3, there are techniques to evaluate whether a prediction seems plausible or suspicious (eg. shelters which do not have a compact shape are more likely to be suspicious). By developping further this tagging system, the analyst could go straight to suspicious predictions and thereby diminish the time required to review a camp.

  2. Shelter splitting: the most recurring error of the network is to fuse shelters together. Since each shelter is predicted separately, this leads to overlapping prediction. A simple post processing algorithm could split the fused tents into several when detecting overlapping predictions (meaning neighbor tents where detected as one).

7.2 Next Steps

The goal of this section is to highlight the concrete next steps to be taken for the project. They are ordered in a chronological order.

7.2.1 Improve Point-to-Polygon Model

Improve performance of the developed model using recommendations listed above. Not all improvements have to be implemented or tested, but several quick fixes will significantly increase performance. This includes ”Post Processing” and ”Dynamic cropping”.

7.2.2 Generate Multi-class Polygon Dataset

Generate a set of valid polygons which are labeled with the different classes UNOSAT cares about (using information transfer from the point data shapefile). The set would include the largest possible set of images listed in Appendix 1. The goal is thereafter to train the network with these high quality generated polygons to predict the different classes directly from the images.

The desired output is a list of coco annotation-style .JSON file(s) with (a selection of) high quality predicted polygons and their corresponding labels. Separate the satellite images into expanded training, validating and testing images. Vary as much as possible in terms of satellite, camps, shelter types (i.e. similarly to the statistics colleced in Table 5.2).

7.2.3 Develop End-to-End Shelter Detector

Once the large multi-class polygon dataset is generated, the second part of the project should be implemented, as explained in Figure 4.2. This will enable the detection of shelters without any point data required. Most recommendations listed above also apply for this second network. It is important to integrate the developed model into the ArcGIS software to ensure easy usage for UNOSAT analysts. Once this step is achieved, all initial requirements of the project will have been fulfilled.

8. Conclusion

The United Nations is a key player to support the millions refugees and internally displaced people (IDP) by providing shelter, food, medicines and other basic needs. Reliable census and geographical data is required to ensure high impact of the UN's operations. Within the UN, UNOSAT is the organ in charge of collecting such data based on satellite images of the refugee camps. However, this task has become increasingly difficult, as the world is currently undergoing the largest refugee crisis in modern human history, with more than 68 millions refugees and IDP concerned worldwide.

The goal of this project is thus to provide a user-friendly automated refugee shelter detection and polygonization tool for UNOSAT analysts. This tool increases the efficiency of their analysis (camps can be surveyed faster) and the value of the collected data (area of shelters is a better proxy for inhabitants and living space), thereby allowing analysts to collect better data, faster.

A two step pipeline is proposed to develop this tool. First, a model is developed to generate polygon masks of shelters using shelter location point data generated by UNOSAT analysts in the past 10 years. Second, a model is developed to detect and polygonize multi-class shelters in new camp images without any point data. The previously generated dataset is used to train the second model.

This report focuses exclusively on the first model. The three requirements have been reached:

1. The model is capable of polygonizing refugee shelters with an average precision of 81% and average recall of 88%, thus above the 80% AP – 80% AR target. Indeed, higher targets are not meaningful as the ground truth itself is noisy (noise level estimated to 5-10%), and fused shelters are penalized although the overall area of the prediction is similar to the sum of the shelters annotated separately by humans.
   A conditioned Mask R-CNN with Resnet 50 is used to achieve these results, trained for 2 days (180000 shelters seen - approx. 3 epoch) on two Nvidia GTX 1080 GPUs.

2. The model is flexible to different input size, image types and number of bands (currently, only the first three are used for inference).

3. The model is packaged and can be used with a one-liner in the command line after installation, thereby integrating nicely in the workflow of UNOSAT analysts:

```
python2 polygon_maker.py <base-directory> <output-directory>
```

The main caveat of the tool is that it sometimes fuses neighboring shelters human annotators

considered separate. Nevertheless, the overall predicted shelter by the tool corresponds well to the sum of the two manually annotated shelters. In addition, such fusions can easily be mitigated using a simple post processing step.

The intermediate tool can be used not only to generate the required dataset for the second model, but also to polygonize any new camp analyzed by UNOSAT analysts in the meanwhile, to obtain shelter area data.

This tool has the potential to multiply by a factor 10 the number of ground truth polygons available in the dataset, if run on all the usable satellite images of the UNOSAT database. Moreover, it devides by 200 the time required to create new shelter polygons (compared to human annotators).

In a nutshell, the obtained results not only fulfill the initial requirements but also also establishes a first benchmark on the UNOSAT mini-validation dataset. Further research and development will decuplate the impact of the current tool both by improving performance and user friendliness. In essence, the proof of concept of a reliable, truly automated shelter detection has clearly been demonstrated and this work paves the way to such a tool for UNOSAT in a near future.

Bibliography

[1] crowdAI - Mapping Challenge, Oct 2018. [Online; accessed 27. Oct. 2018]. 16

[2] F1/Dice-Score vs IoU, Nov 2018. [Online; accessed 25. Nov. 2018]. 28

[3] Polsby-Popper Test - Wikipedia, Nov 2018. [Online; accessed 17. Oct. 2018]. 22

[4] UNHCR Global Trends 2017. Technical report, United Nations High Commissioner for Refugees (UNHCR), June 2018. [Online; accessed 13. Dec. 2018]. 8

[5] Baptiste Angles, Shahram Izadi, Andrea Tagliasacchi, and Kwang Moo Yi. MIST: Multiple Instance Spatial Transformer Network. *ArXiv e-prints*, Nov 2018. 37

[6] S. Beucher. The watershed transformation applied to image segmentation. *SCANNING MICROSCOPY-SUPPLEMENT*, pages 299–299, 1992. 20

[7] S. Giada, T. De Groeve, D. Ehrlich, and P. Soille. Information extraction from very high resolution satellite imagery over Lukole refugee camp, Tanzania. *International Journal of Remote Sensing*, 24(22):4251–4266, Jan 2003. 15

[8] E. Guirado, S. Tabik, D. Alcaraz-Segura, J. Cabello, and F. Herrera. Deep-learning Versus OBIA for Scattered Shrub Detection with Google Earth Imagery: Ziziphus lotus as Case Study. *Remote Sensing*, 9(12):1220, Nov 2017. 16

[9] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 16

[10] Kamil Kaczmarek. Mapping Challenge winning solution. *Towards Data Science*, Sep 2018. 36

[11] G. Laneve, G. Santilli, and I. Lingenfelder. Development of Automatic Techniques for Refugee Camps Monitoring using Very High Spatial Resolution (VHSR) Satellite Imagery. *2006 IEEE International Symposium on Geoscience and Remote Sensing*, Jul 2006. 15

[12] S. Lang, D. Tiede, D. Hölbling, P. Füreder, and P. Zeil. Earth observation (EO)-based ex post assessment of internally displaced person (IDP) camp evolution and population dynamics in Zam Zam, Darfur. *International Journal of Remote Sensing*, 31(21):5709–5731, Nov 2010. 15

[13] D. Tiede, P. Füreder, S. Lang, D. Hölbling, and P. Zeil. Automated analysis of satellite im-

agery to provide information products for humanitarian relief operations in refugee camps - from scientific development towards operational services. *PFG Photogrammetrie, Fernerkundung, Geoinformation*, 2013(3):185–195, 06 2013. 9

[14] D. Tiede, S. Lang, D. Hölbling, and P. Füreder. Transferability of Obia Rulesets for Idp Camp Analysis in Darfur. 2010. [Online; accessed 27. Oct. 2018]. 15

Table 1: Images for expanded polygonized dataset

Image Name	Status	Remark	Fixable ?
doro_01711110830479	Usable		
Wau_0161215_V03	Usable		
doro_3dec14084932	Usable		
Muna_0160905_V03	Usable		
Nyal_0170108_V03	Usable		
Yida_June2017_3	Usable		
AjouonTh_1Apr17_V1	Usable		
Khaldiyah_0170108_V1	Partially Usable	Point data generated from polygon centrods. Not all tents were polygonized	Yes, by completing corresponding shapefile
Juba_8Feb2017_V03	Usable		
Ganyel_0170108_V03	Usable		
HTCcamp_0161226_V1	Usable		
Ngala_nov2016_2	Mitigated	Idem as Khaldiyah. Ground Truth polygons are of very bad quality.	maybe by using NN prediction as ground truth else repolygonize image.
K18_0170228_V1	Usable		
Azraq_0jun2016_3	Usable		
Batil_0120805_E01	Maybe Usable (?)	Point data not in right CRS, pansharpening of rather bad quality. This affects the shape of the tents	Reproject .shp file or algorithm upgrade.
Batil_0131125_V02	Not Usable	Point data shifted from image. Shelter status is given only for very few points.	Probably not with this shapefile. The image is ok.
babalsalame_0150605_L	Mitigated	Image of very bad quality. Shelter Status code 0 for open instead of 1.	Use a unified standard to indicate closed shelters (either 'open', 'closed' or '0', '1' or '1', '0' but not a different one for each image !) + algorithm upgrade
CoxBasar	Not Usable	No point data. Cannot generate from polygons since polygons are of very bad quality (shifter). Moreover, picture taken from drone is too different	Probably not.
Delthoma_0151129_V1	Probably Usable	Pb because of combination of band usage and datatype (uint16)	Algorithm upgrade

Table 1 continued from previous page

Image Name	**Status**	Remark	Fixable ?
Gendrasa_Apr2015_V01	Maybe Usable	Offset of point data	correct offset and hope it is uniform
hadalat_0170915_l	Mitigated	Poor quality image. Conflict in shelter Status standard (see babalsalame)	Decide of the shelter status standard + algorithm upgrade
jamatwo_0120928_v02	Probably Usable	Pb because of combination of band usage and datatype (uint16)	Algorithm upgrade
Leitchour_0140505_V01	Probably Usable	Pb because of combination of band usage and datatype (uint16)	Algorithm upgrade
Malakal_0feb2016_2	Mitigated	*Image of very bad quality*	Possibly with an algorithm improvement this could be captured but very challenging.
Melut_0141202_V02	Probably Usable	Wrong CRS and datatype	Algorithm upgrade
MPokoBangui_oct2015_v03	Probably Usable	Pb because of combination of band usage and datatype (uint16)	Algorithm upgrade
Oncunipar_0150605_L	Mitigated	parts of image are bad quality	Algorithm improvement
sujjo_0150605_l	Usable		
YousefBatil_0171031_L	Usable