SAPIENZA
UNIVERSITÀ DI ROMA

Smart Environment Monitoring through micro Unmanned Aerial Vehicles

Faculty of Information Engineering, Informatics, and Statistics

Doctor of Philosophy in Computer Science – XXXI Cycle

Candidate

Daniele Pannone

ID number 1261377

Thesis Advisor

Prof. Luigi Cinque

Co-Advisors

Prof. Gian Luca Foresti

Prof. Danilo Avola

Smart Environment Monitoring through micro Unmanned Aerial Vehicles
Ph.D. thesis. Sapienza University

"I've seen things you people wouldn't believe. Attack ships on fire off the shoulder of Orion. I watched C-beams glitter in the dark near the Tannhäuser Gate. All those moments will be lost in time, like tears in rain. Time to die."

Roy Batty - Blade Runner

Abstract

In recent years, the improvements of small-scale Unmanned Aerial Vehicles (UAVs) in terms of flight time, automatic control, and remote transmission are promoting the development of a wide range of practical applications. In aerial video surveillance, the monitoring of broad areas still has many challenges due to the achievement of different tasks in real-time, including mosaicking, change detection, and object detection. In this thesis work, a small-scale UAV based vision system to maintain regular surveillance over target areas is proposed. The system works in two modes. The first mode allows to monitor an area of interest by performing several flights. During the first flight, it creates an incremental geo-referenced mosaic of an area of interest and classifies all the known elements (e.g., persons) found on the ground by an improved Faster R-CNN architecture previously trained. In subsequent reconnaissance flights, the system searches for any changes (e.g., disappearance of persons) that may occur in the mosaic by a histogram equalization and RGB-Local Binary Pattern (RGB-LBP) based algorithm. If present, the mosaic is updated. The second mode, allows to perform a real-time classification by using, again, our improved Faster R-CNN model, useful for time-critical operations. Thanks to different design features, the system works in real-time and performs mosaicking and change detection tasks at low-altitude, thus allowing the classification even of small objects. The proposed system was tested by using the whole set of challenging video sequences contained in the UAV Mosaicking and Change Detection (UMCD) dataset and other public datasets. The evaluation of the system by well-known performance metrics has shown remarkable results in terms of mosaic creation and updating, as well as in terms of change detection and object detection.

Contents

List of Symbols

C_m	Camera matrix
f_x, f_y	Focal length parameters
c_x, c_y	Camera optical center
k_1, k_2, k_3	Radial distortion coefficients
p_1, p_2	Tangential distortion coefficients
x_d, y_d	Actual coordinates
r	Distance from optical center
UAV_{path}	Set of the area of interest GPS coordinates
F_{TRS}	Set of transmitted frames
H	Homography matrix
ST	Similarity transformation matrix
AoV	Angle of View
W_{CCD}, H_{CCD}	Width and height of the CCD sensor
h	UAV Flight height
f	Sensor focal length
W_p, L_p	Camera spatial resolution
h_{max}	Maximum flight height for a specific task
SRJ	Minimum spatial resolution needed for a specific task
fr	Framerate
fr_{min}	Minimum framerate
s	Overlap parameter
$\phi(x_k, y_k)$	Interpolated GPS coordinate
T, Q	Images extracted from the mosaic, and the frame received from the UAV
O	Change detection output image
L	Change detection output bounding boxes list
T_{diff}	Difference threshold
$D_{i,j}$	Pixel resulting from the grayscale difference
S_R, S_G, S_B	Binary strings computed on each image channel with LBP method
T_H	Hamming distance threshold
M_{diff}	Binary mask obtained in the change detection image difference step
$N_{i,j}$	Binary mask with isolations removed
T_{iso}	Isolations threshold
T_{area}	Blob removal threshold
B	Bhattacharyya distance

Chapter 1

Introduction

In the last decade, the use of small-scale UAV based vision systems is gaining more and more importance to successfully deal with a wide range of application contexts. This is due to the fact that these systems make possible to achieve certain aims more advantageously than similar systems based on other types of aircraft. Additionally, in some contexts, the small-scale UAV based vision systems have led to accomplish tasks otherwise unattainable. A recent example is reported in [5], where this kind of UAVs is used to implement a change detection vision system for constant monitoring of wide areas at low-altitude. The authors highlight that such a type of tasks can only be achieved by using small-scale UAVs since they can be used several times over a day. Moreover, these UAVs allow to reach target areas otherwise unreachable by larger aircrafts. Another interesting example is presented in [61], where a small-scale UAV is used to address trail detection and tracking, as well as autonomous scene understanding problems to support a wide range of missions in unstructured outdoor environments, such as isolated disaster sites. Also in this case, the authors point out how the real-time processing and the limited size of the adopted UAV play a key role. A final example is discussed in [59], where an automatic small-scale UAV based inspection vision system for asset assessment and defect detection of large-scale photovoltaic power plants is detailed. The authors underline how their system is more efficient in comparison with conventional methods.

In aerial video surveillance, small-scale UAVs have a set of features that, jointly to those previously exposed, make them particularly suitable for reaching different missions. First of all, thanks to their small size and being they always more silent, these UAVs can pass unnoticed and can hide out easily. Furthermore, for take-off and landing, they do not require any structure or platform. Finally, these UAVs can fly at very low-altitude and very reduced speed, thus facilitating complex tasks that require high spatial resolution, such as person re-identification and small object detection.

This thesis work presents a small-scale UAV based vision system to monitor broad target areas by detecting and classifying changes on the ground. The system works in two modes. The first mode allows the monitoring of an area of interest, and it is composed by two steps. In the first step, the UAV performs one or more flights and creates an incremental geo-referenced mosaic of the area. Each flight is driven by a set of GPS coordinates that specifies the area to be mosaicked. During the

mosaicking, all the known elements found on the ground are classified by a previously trained Faster R-CNN [79]. In this thesis, an improvement for the Region Proposal Network (RPN) of the Faster R-CNN has been proposed. After being classified, the object is localized within the mosaic through the GPS coordinates. In detail, three classes of elements were defined: person, car, and small object. In particular, with the label car a class composed of some types of economy cars is defined, while with the label small object a class composed of selected small objects is defined. In aerial video surveillance, the last class is of great importance since dangerous gadgets, e.g., Improvised Explosive Devices (IEDs), can be hosted inside seemingly harmless items like boxes or little suitcases. In the second step, regular reconnaissance flights to update the mosaic are performed. If the new video sequences have GPS coordinates outside to those of the created mosaic, then new areas are added to it and any known element is once again classified and localized. On the contrary, if the GPS coordinates are inside, then a comparison between the current frames and the associated portions of the mosaic is performed. To make the comparison robust to the illumination changes that afflict the outdoor environments, a histogram equalization preprocessing stage is carried out. Subsequently, an RGB-LBP based algorithm is used to detect the possible changes [5], e.g., a person appears or a car disappears, which if present drive the mosaic updating.

The second mode, uses again the proposed modified Faster R-CNN to classify, in real-time, the elements defined above. In this mode, no mosaic is created nor is needed, since this mode is designed for time-critical operations such Search and Rescue (SAR).

1.1 State of the Art

In this section, the state-of-the-art regarding mosaicking, change detection and object classification is discussed. In each section, firstly are discussed the general works in the field, then works focused on UAV application are exposed.

1.1.1 Mosaicking

Image stitching, or mosaicking, has been widely investigated in literature. Usually, the main reason for merging together two or more images is to create a panorama image. The basic problem of the image stitching process is well known, and consist in estimating a transformation matrix. This matrix, namely the homography, is computed for each pair of image, and it is used to match the images. The first step in estimating the homography consists in the initialization, or an input provided by the user to facilitate the image alignment. A typical initialization provided is, for example, the order in which the images are acquired. In the state-of-the-art, there are mainly two categories of image alignment and stitching. The first is the direct method [93, 44, 83, 94], which have the advantage of using all the available image data. This leads to a very accurate image registration, but a very close initialization is required. The second method, in which is the majority of the works at the state-of-the-art fall, is the feature based method [15, 110, 119, 21, 65]. This method does not require initialization, allowing real-time performances. The mosaicking

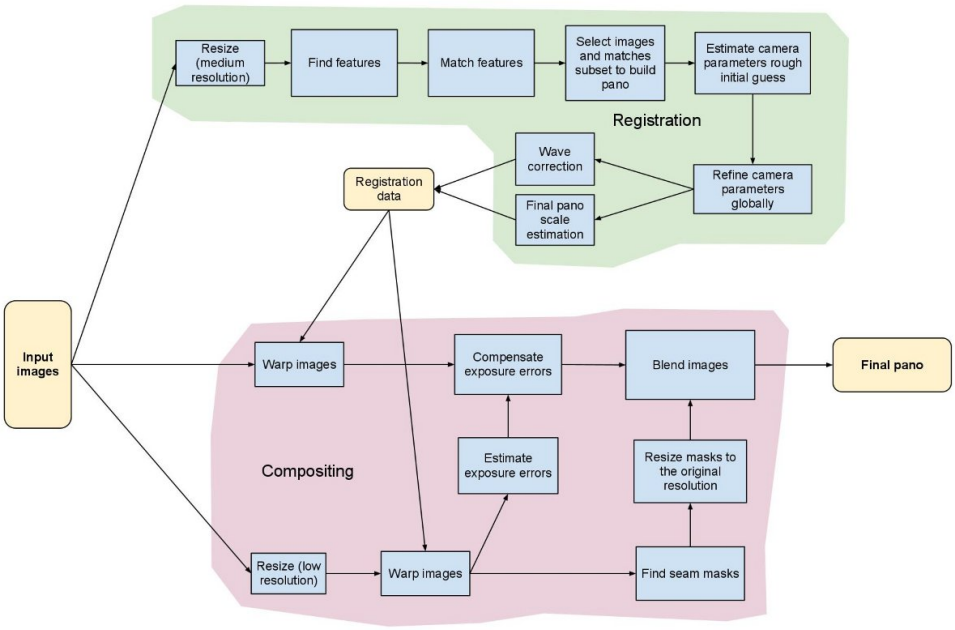


Figure 1.1. Example of the OpenCV stitching pipeline. As it is possible to see, the Image Registration and the Image Composing modules are composed of several sub-modules exploited to optimize the stitching process. [73]

process performed with the feature based method has a well-known pipeline, which consists of the following main steps:

- **Frame Correction**: This step consists in removing any type of distortions, e.g., radial, barrel, and tangent, introduced by the sensor lens.
- **Image Registration** : This is the most important step of the pipeline. It consists in extracting salient points from the overlapping parts of all the images that will be stitched together. Then, these points are matched to find the transformation matrix that allows to align the images.
- **Image Composing**: In this step, the aligned images are blended together to create the final mosaic.

Despite the entire process may seems trivial, there are many sub-processes that are exploited to generate the best mosaic possible. Depending on the framework or on the used software, these sub-processes may vary. An example of pipeline comprising all the modules is shown in Figure 1.1, which is the one contained in the OpenCV framework.

This pipeline follows the steps presented by Brown and Lowe in [15], that is one of the first works that described the entire pipeline. In this work, an invariant feature based approach to create panoramic images automatically is presented, bringing several advantages with respect to the other approaches. A first advantage was

Figure 1.2. Example of mosaic created with the steps described in [15].

given by using Scale Invariant Features Transform (SIFT)[63] feature extractor and Random Sample Consensus (RANSAC)[32], which allow to match images despite rotation, zoom and illumination change in the input images. The second advantage is that, by using a multi-band blender technique [19], it is possible to generate seamless high quality panoramic images. In Figure 1.2, an example of mosaic that obtained with the Brown and Lowe pipeline.

The subsequent works addressed the several problems that may arise in generating a mosaic with UAV, or aimed to improve one or more of the steps proposed in [15]. For example, authors in [110] proposed a method based on SIFT and histogram matching to perform the image stitching. Since image stitching among images that have significant illumination changes will lead to unnatural mosaic image, the authors first use the histogram match to perform image registration so that the images to be stitched are at the same level of illumination. Then, SIFT algorithm is used to extract the keypoints for a rough matching process, followed by RANSAC for the matching process. Finally, a simple weighted average algorithm is used for the image blending step. Authors in [27], proposed a stitching method that reduces the artifacts caused by different parallaxes. This is achieved by selecting an optimal seam pair by comparing the cross correlations from multiple seams detected by variable cost weights. Then, the homography matrix is refined by using matching among Histogram of Oriented Gradients (HOG) [28] features, and the remaining misalignment is eliminated using the propagation of deformation vectors calculated from matching points. In the blending step, the regions that are overlapped are determined by using a distance between the matching points to remove overlapping artifacts. In [105], instead, the authors used multiple UAVs to perform mosaic of disaster scenario. Together with the mosaicking algorithm, a prioritized image transfer protocol is provided in order to transmit efficiently the images to the ground station by using the limited UAVs hardware and network resources.

Due to the high number of steps in the mosaicking process, it may require a large amount of time to produce the final image. Some works at the state-of-the-art aim to optimize some steps to speed up the process. Example of works in this direction is the method proposed in [20]. In the first work, an improved method based on Speed up Robust Features (SURF)[8] is proposed. In detail, Support Vector Machine (SVM) [25] are used to predict primary threshold of determinant of Hessian matrix to reduce detected feature points and simplify the process of features matching. Then, an optimized method of image preprocessing-cylindrical projection and image interpolation is used to weigh the final quality of stitching image and stitching time.

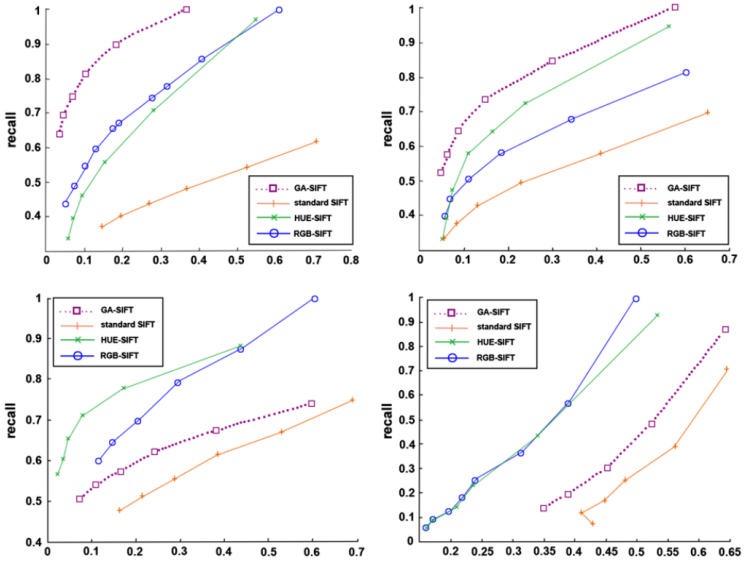


Figure 1.3. Comparison between GA-SIFT and other feature extraction methods.

Figure 1.4. Example mosaic obtained with GA-SIFT and O-RANSAC.

The fields that have benefited most from the mosaicking are the ones in which UAVs are used. This is due to the fact that once the mosaic of an area of interest has been generated, it can be analyzed with several approaches, i.e. by using change detection or object detection/classification algorithms. There are several recent works in image mosaicking from UAV. In [113], the authors proposed an improved Geometric Algebra SIFT (GA-SIFT) to perform fast features extraction and matching. The GA-SIFT can detect more feature points with greater accuracy than the traditional SIFT method. Moreover, to perform the image alignment they presented the optimal modified random sample consensus (O-RANSAC), that has full affine-invariance while maintaining the rapidity advantage. In Figure 1.3, the comparison between GA-SIFT and other feature extractor is shown, while in Figure 1.4 the resultant mosaic is depicted.

Authors in [57] presented a mosaicking algorithm for UAV based on dynamic programming and stereo images. The algorithm aims to solve the dislocation problem that is caused by the seam line in building-intensive areas. The idea is to analyze the

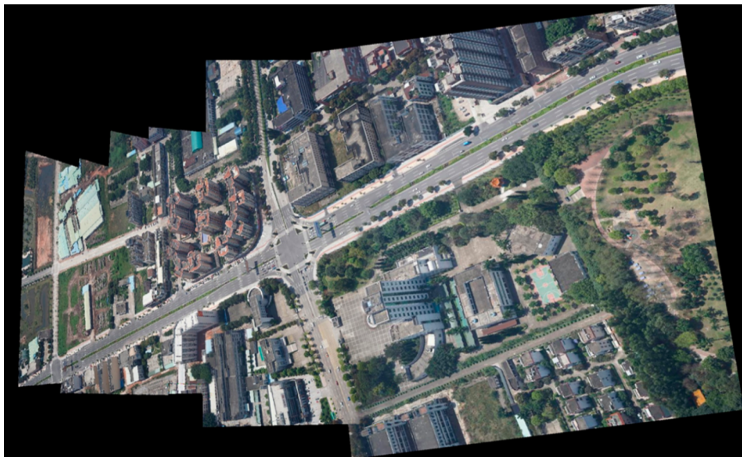


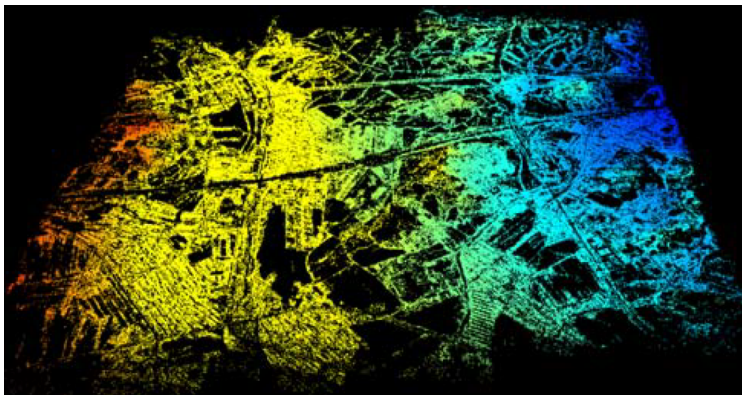
Figure 1.5. Mosaic generated with the dynamc programming algorithm presented in [57].

mapping relationships between the left and right images in order to determine the geometric errors introduced by perspective errors, camera distortions, and radiation errors. Since it is an optimization problem, there is no unique solution for this image seam line searching problem. The authors consider the close optimum solution the one that meets the requirements of image stitching. In Figure 1.5 the resultant mosaic is shown.

By considering Structure From Motion (SfM)[120] technique, authors in [102] presented the standard mosaicking procedure based on SfM. It differs from the standard mosaicking approach on two points. The first is that the features are computed on a 3D space, while the second is that an ortho-rectification step is performed. As it is possible to see in Figure 1.6, with the SfM approach it is possible to obtain both 3D data and 2D mosaic. Another method based on SfM is the one presented in [117]. The authors proposed a novel and robust method to register multiple images captured by UAV with modified camera matrices. In detail, images are registered using camera matrix modified by vertical vector of fitted plane, and the reference plane (i.e., the plane parallel to the generated mosaic) is determined by performing view selection. Finally, like the other methods the blending is performed by using the multi-band blending.

The problem with SfM approach is that it has a very high computational cost, which makes impossible to use it for real-time computation. A solution has been proposed in [17]. Instead of using SfM, the authors used a Simultaneous Localization and Mapping (SLAM)[95] approach, which allowed to build a mosaic incrementally and in real-time. Despite the good performances, the method may introduce some artifacts in the final mosaic.

Finally, there are works that exploit the classical mosaicking pipeline with the aim to optimize it for embedded hardware platforms such as DSP and FPGA. An example is the work presented in [101], in which the authors proposed an ORB based stitching method for aerial images optimized for embedded platforms. In detail, the authors designed a parallel approach in order to perform the stitching on all the cores of the DSP platform TMS320C6678, showing near real time performances and good final mosaic, as shown in Figure 1.7.



(a)

(b)

Figure 1.6. Example of a) 3D model computed with SfM and b) the corresponding 2D mosaic.[102]

Figure 1.7. Mosaic generated by using the DSP platform TMS320C6678.

1.1.2 Change Detection

With the term Change detection (CD) is considered that set of algorithms that aim to identify, precisely, the changes that occur within the observed image. Important applications of change detection include video surveillance [24, 89, 106], remote sensing [16, 23, 43], medical diagnosis and treatment [14, 80], civil infrastructure [53, 69], and underwater sensing [31, 54]. Usually, the approach to perform CD is the background subtraction technique [46]. It consists in considering one image as ground truth, and to subtract it from the subsequently images. As result, all the pixels changed between the images involved in the subtraction are obtained. As it is possible to notice, the simple subtraction is not a robust method since some noise sources such as shadows, trees and water moved by the wind, illumination changes, and other can negatively affect the result of the subtraction. There are several recent state-of-the-art works that aim to enhance the CD process. Authors in [96] proposed a colour consistency and a local contrast enhancement for mobile image-based CD. The method aim to compensate lightness and colour inconsistencies depending on different illumination conditions, due to shadows or to the different time instants at which the area is acquired (i.e., morning and afternoon). The proposed approach combines the centre/surround Retinex model [49] and the Gray World hypothesis [18] using a non-linear colour processing function previously used for colour restoration [48]. In [84], a light field camera has been introduced to avoid false detections, as the one shown in Figure 1.8. The light field camera is used to configure an active surveillance field, within which the focusness, based on spatial consistency of the light rays, and foregroundness, based on background modelling to detect temporal changes in the light rays, are measured.

Another field in which performing change detection is a difficult task is underwater image analysis. Authors in [81] proposed a local change detection method for detecting the position of the moving objects. To achieve this, the Mixture of Gaussian (MoG) process has been integrated in a Wronskian framework. The latter takes care of the spatio-contextual modes whereas MoG models the temporal modes arising due to inter-dependency of a pixel in a video. In [77], instead, used a Flux

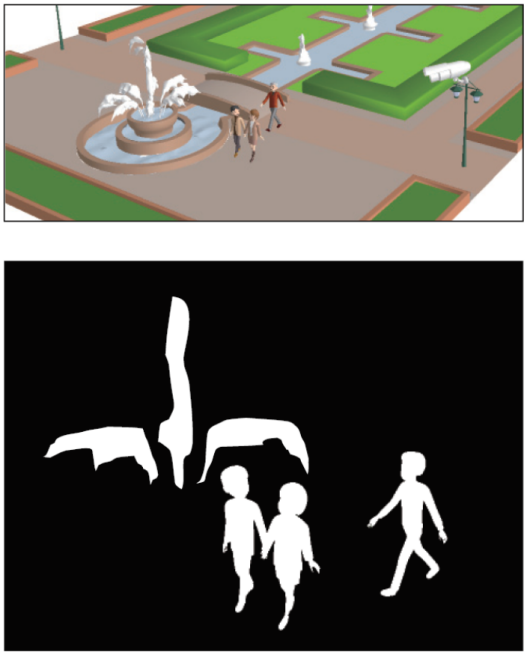


Figure 1.8. Example of false positive in change detection algorithms. The fountain is part of the background, but since it is moving is detected as a change.

Tensor-based approach as pre-segmentations to exclude moving parts of the image from the background updating process. Then, the Gaussian Switch Model [78] has been enhanced with the Mixture of Gaussian idea, a foreground model and an intelligent updating scheme to make the method robust to difficult scenarios.

Other recent approaches tried to use Artificial Intelligence and Deep Learning to perform CD. In [12], the authors proposed a genetic programming method to combine automatically change detection algorithms. In detail, the method can: a) automatically select the algorithms that give the best overall results relative to a set of predefined algorithms; b) automatically deduce which kind of post-processing of the original or intermediate masks to be applied in order to improve the results, is automatically built from the unary, binary and n-ary functions fed as input to the algorithm. In [37] authors proposed a generative adversarial networks (GANs) method able to recover the training data distribution from noise input. In detail, the prior knowledge for sampling the training data is given by the joint distribution of the two images to be detected is taken as input. Then, through the adversarial learning the shared mapping function between the training data and their corresponding image patches is built. The last step consists in using an unsupervised clustering algorithm is used to analyze the difference image to obtain the desired binary change map.

Taking in consideration the works concerning aerial images, authors in [107] proposed a novel binarization model based on the Weibull mixture model. The method models the changed and unchanged region classes using non-normal Weibull distributions and further estimates parameters using a genetic algorithm. In Figure 1.9, the result of the CD by applying the improved binarization is shown. In [39], a variant of a local binary similarity pattern (LBSP) descriptor is proposed. The LBSP

Figure 1.9. Result of the improved binarization proposed in [107].

Figure 1.10. Change Detection performed through multiscale uncertainty analysis [112].

has been chosen due to its good resistance to illumination variation. The presented algorithm consists in two steps: binary feature vector creation and generation of binary change map. The first is obtained through LBSP, while the latter in obtained by using the Hamming distance as a similarity metric. Authors in [22] proposed an unsupervised 2-D and 3-D urban change detection scheme based on Quad-PolSAR data. The changes are extracted by segmenting the images into superpixels, and the positive and negative changes in both direction (i.e., horizontal and vertical) are computed by using a multivariate Gaussian mixed model. The latter is applied to a subset of polarimetric parameters at the superpixel level. Considering the scale information, authors in [112] proposed an object-based change detection (OBCD) algorithm for very high resolution images that uses multiscale uncertainty analysis. The images taken at different time instants are stacked and segmented by using "from coarse to fine, refine layer by layer" strategy. Then, a first CD is performed by fusing the pixel-based CD result and OBCD result based on Dempter–Shafer (DS) evidence theory. Subsequently, the multiscale uncertainty is implemented by SVM classification. Finally, the resultant CD map is obtained by fusing the information available in all the scales, as shown in Figure 1.10. By considering the deep learning approach, authors in [74] presented and algorithm to perform CD for damage assessment caused by fires. The novelty introduced by the work is the feature extraction within tunable Q discrete wavelet transform (TQWT) using higher order log cumulants of fractional Fourier transform (FrFT). The, these features are given as input to a stacked autoencoder to distinguish changed and unchanged areas. After the stacked autoencoder training, the decoding layer is replaced by a logistic regression layer which performs fine-tuning and classification.

1.1.3 Object Detection

The object classification task is one of the hardest tasks to perform in Computer Vision. It consists, as it suggests, to give a class to an object present in the observed scene. Usually, the detection is performed by using a model trained on the features

Figure 1.11. Pedestrian detected through a CNN.

extracted from the samples of the classes. In the last years, the research moved from the recognition and classification of the objects through features [62] to deep learning [56] approaches. This is due to the fact that, despite the high amount of time needed to train a deep model, it can achieve better results than humans [42]. There are several deep neural network model for object detection and classification: Convolutional Neural Network (CNN) [55], Single Shot Multibox Detector (SSD) [60], Faster Region Convolutional Neural Network (Faster R-CNN) [79], Inception [92], and others. These models find usage in different field and tasks. For example, in [115] a Faster R-CNN is used to perform pedestrian detection, as shown in Figure 1.11. Instead of using a K-object classifier, a pedestrian specific softmax is used, and the ConvNet parameters are tuned with features from the Caltech Pedestrian Dataset. Considering 3-D data, in [13], a deep learning model for classifying objects in point cloud is proposed. In detail, the authors designed a novel 4-D convolutional layer that takes as input a 4-D descriptor. To feed proper data to the layer, a point pair descriptor that is robust to noise and occlusion optimized for point cloud data is also provided. Authors in [71] proposed an improvement of the SSD network for object detection. Despite the SSD is one of the fastest algorithm, it has a big gap in Mean Average Precision (mAP) with respect to other approaches. The improvement consists in replacing the extra layers of the SSD network with the Inception block, allowing to catch more information without increasing the complexity. Concerning classification works on UAV field that use deep learning technique, in [109] the problem of multilabelling in UAV imagery has been addressed. In detail, a pretrained GoogLeNet CNN is used to extract the features from the images after they have been subdivided in equally spaced grid. Since GoogLeNet is not directly adapted to multilabel classification tasks, the authors substituted the softmax classifier with a radial basis function neural network (RBFNN), which is a classifier that can fit the multilabeling requirement. The methodology of dividing images into regions has been used also in [2] for detecting cars in UAV imagery. In

Figure 1.12. Detection of cars in UAV imagery by using CNN and SVM.

Figure 1.13. Example of avalanche victims detected through CNN.

this case, a window is extracted around each region, and also in this case a CNN is used for feature extraction. Finally, the classification is performed by using a SVM to simply dividing regions in ”car” and ”no-car”, and in Figure 1.12 the classification result is shown. Other works that use CNN in UAV imagery are [3] and [9]. In the first work, they are used to discriminate between different levels of damage during damage assessment and monitoring operations. In the second work, which consists in detecting victims after avalanches, are used in the same way of [2]. In Figure 1.13, the result of victim detection is shown.

In [7], instead of CNN the SSD has been used to detect concurrent human action detection. In detail, the SSD is used for the pedestrian detection task, and once the pedestrian has been detected they used the spatio-temporal model presented in [87] to classify the action performed. In Figure 1.14, the result of the action recognition is shown.

1.1.4 Datasets

The role of the datasets is a key factor in computer vision. This section reports some popular aerial-based datasets to highlight their importance in some application

Figure 1.14. Pedestrian action detected by a UAV through a SSD network.

fields. Moreover, the section also shows the need to have available a public dataset focused on testing mosaicking and change detection algorithms with aerial video sequences acquired at low-altitude. In [72] one of the most popular and appreciated large-scale dataset, named VIRAT, is presented. It consists of both ground and aerial video sequences whose aim is to assess the performance of diverse visual event recognition algorithms in outdoor areas with wide coverage. Focused only on aerial images is, instead, the dataset reported in [108] by which the authors present a benchmarking study for aerial image segmentation. The aerial videos contained in these datasets have not been acquired to support the purposes described in the present paper. In fact, on one side, they do not have the suitable altitude and length to stress the mosaicking algorithms. On the other hand, they do not contain same paths, with and without objects, to test the change detection algorithms. Finally, these sequences are not aligned with GPS coordinates to check the working of the on-line algorithms. Regarding the video acquisitions by using small-scale UAVs, in [85] a dataset to support aerial video analysis is provided. In particular, the authors propose a repository for detecting 12 events (e.g., exchanging box, sit on table), 18 human roles (e.g., deliverer, receiver) and 12 object categories (e.g., box, car). Also in this case, the videos cannot be used to support the purposes of the present paper for the same reasons observed above. Concerning the change detection, in [104] a benchmark dataset, named CDnet, is reported. It contains challenging videos for testing several tasks, including dynamic background, camera jitter, and shadows. Finally, in [17] the NPU dataset is presented. This dataset contains videos for performing mosaicking at high-altitudes (i.e., from 65 to 376 meters) and it is used to test the algorithm proposed by the authors. To the best of our knowledge, currently, there are no datasets containing video sequences acquired at low-altitude for mosaicking and change detection purposes. However, the public availability of this kind of datasets would provide a support to different open issues, including the management of the parallax error, the testing of the on-line algorithms, and the detection of small objects on the ground. Moreover, also other types of application domains could take advantages from this kind of datasets, some examples include UAV camera automation [51], search and rescue [99, 29], routing and delivery [30],

target searching and localization [47, 67], target following [58, 97], and monitoring of malicious UAVs [68].

1.2 Contribution

This section reports in brief the contribution of this thesis work with respect to the state-of-the-art. Concerning the mosaicking process, the first improvement is that it is performed at very low altitudes. As mentioned previously, this has been possible thanks to the usage of affine transformation instead of homography, which limits the transformations applicable to the image. Another contribution is the implementation of a ROI technique, allowing to perform the mosaicking on-line and in near real-time, limiting the time needed to extract the features and the hardware utilization. The last contribution is the usage of A-KAZE feature extraction, allowing a fast extraction of robust features.

As for mosaicking, also in CD state-of-the-art works the used images are acquired from satellites or from very high altitudes, thus making impossible to detect small objects, persons or vehicles. In this thesis work, the proposed CD algorithm is designed to work with images acquired at very low altitude, by implementing a novel robust pipeline .

Regarding object classification, the majority of the works at the state-of-the-art uses well-known deep architecture to perform classification task. The modification most performed in a deep network is at classification level, by replacing the fully connected layer with softmax or a SVM. In this thesis work, a new architecture based on Faster R-CNN is proposed. The proposed network aim to maximize the precision in object detection, by replacing the RPN of the classical Faster R-CNN with a new network specially designed.

Finally, to test exhaustively the methods and the algorithms proposed in this work, the UAV Mosaicking and Change Detection (UMCD) dataset is proposed. The latter consists in 50 challenging videos, specially acquired for mosaicking and change detection tasks at very low altitude.

This thesis is structured as follows. In Chapter 2, the UMCD dataset is presented. In Chapter 3, the Mosaicking, Change Detection, and Object Detection modules are described in detail. In Chapter 4, the performed experiments and the results obtained are discussed. Finally, Chapter 5 concludes this work.

Chapter 2

The Dataset

In this chapter, the UAV mosaicking and change detection dataset (UMCD) is described. The dataset has been built to test the proposed mosaicking, change detection, and object detection algorithms, due to the lack of aerial datasets containing images acquired at very low flight altitude.

In Figure 2.1, the main structure of the UMCD dataset[1] is shown. It consists of 50 challenging video sequences acquired by small-scale UAVs at low-altitude flights. In order to stress both mosaicking and change detection algorithms, each acquisition has been performed by introducing a certain level of variation of parameters, including spatial resolution, frames per second (fps), altitude, and average speed. Moreover, the acquisitions have been taken on three different environments: urban, dirt, and countryside (Figure 2.2). Each video sequence has been acquired by using a path established by a set of GPS coordinates. This last aspect is useful to perform both geo-referenced mosaicking and real-time change detection operations. In fact, the larger the area to be explored in search of a change, the greater the usefulness of adopting GPS coordinates is, to check the current position of the UAV with respect to the geo-referenced mosaic previously acquired. A brief summary of the main characteristics of the video sequences is provided below:

- Two different small-scale UAVs have been used. The first is a DJI Phantom 3 Advanced used with its built-in camera, while the second is a custom home-made exacopter. The latter has allowed us to change cameras to acquire video sequences with different spatial resolutions ranging from 720x540 (4:3, Standard Definition) up to 1920x1080 (16:9, High Definition) pixels per frame;
- The different cameras have also allowed us to acquire video sequences with different frame rates ranging from 24 up to 50 fps. Moreover, the length of the videos is quite variable ranging from 250 up to 3600 frames;
- The different video sequences have been acquired with various altitudes and speeds. In particular, the altitudes ranging from 6 up to 15 meters, while the speeds ranging from 2 up to 12 meters per second.

The use of two different small-scale UAVs allowed to build a dataset containing heterogeneous video sequences. In fact, the two drones differ in some technical

[1] http://www.umcd-dataset.net/

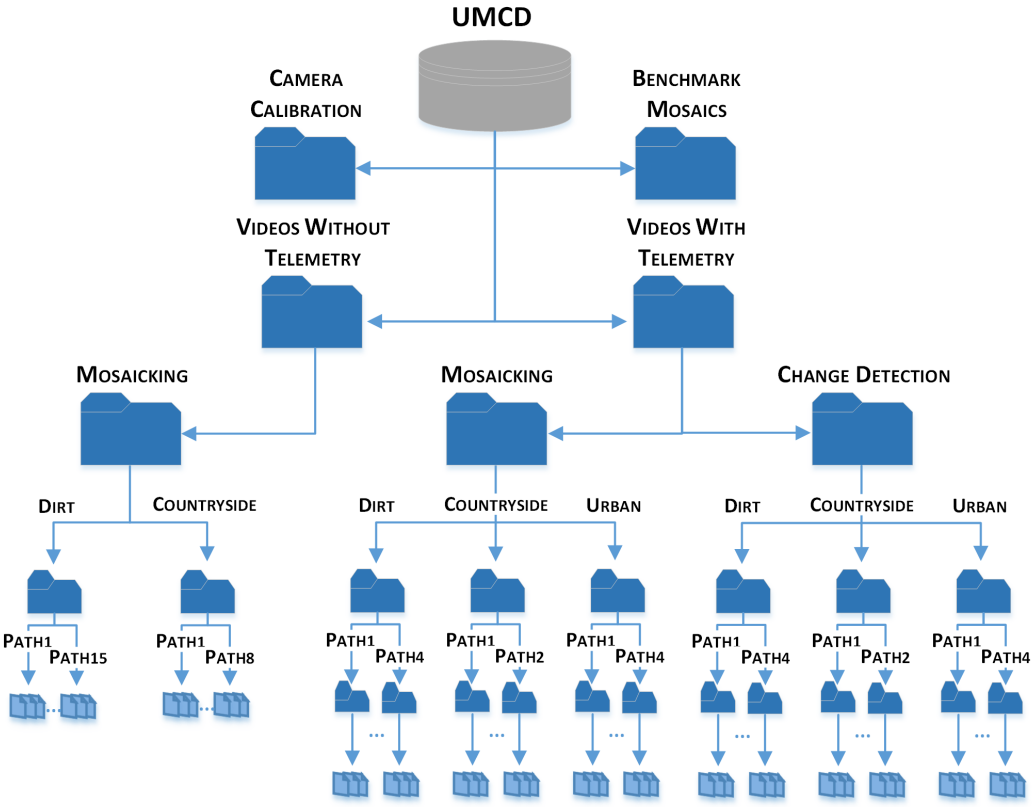


Figure 2.1. Main structure of the UMCD dataset.

features, including camera quality, resolution, and gimbal hardware, thus avoiding an excessive bias of the dataset over a specific system. Anyway, not all the cameras mounted on the home-made exacopter allowed the acquisition of geo-referenced data aligned to the video streams, as a consequence not all the video sequences have been stored with the GPS information. In addition, the cameras have different hue bias due to the specific features of the embedded white balancing algorithm. For this reason, in some videos, the global changing of the brightness can be more or less noticeable. In any case, the purpose of the UMCD dataset is to provide video sequences to test the mosaicking and change detection algorithms in real environments regardless of common illumination changes. All the videos are stored in MP4 format.

The dataset is organized in four main folders (see Figure 2.1): *Videos Without Telemetry*, *Videos With Telemetry*, *Camera Calibration* and *Benchmark Mosaics*. The first one contains the videos acquired to test on-line or off-line mosaicking algorithms without using telemetry parameters but only relying on visual features of the frames. Conversely, the second one contains, for each video, a synchronized text file (e.g., CSV, LOG) whose information can be used to develop on-line (and real-time) or off-line mosaicking and change detection algorithms. The third folder contains the camera calibration parameters file, which allows to unwarp acquired images affected by radial distortion. The last folder contains mosaics obtained with

Table 2.1. Main characteristics of the videos contained in the folder: Videos Without Telemetry → Mosaicking.

Filename	Environment	Length [s]	Frequency [fps]	Total Frames [#]	Altitude [m]	Speed [m/s]	Light Conditions
path01.mp4	Dirt	18	25	450	8	3	morning
path02.mp4	Dirt	15	25	375	9	3	morning
path03.mp4	Dirt	16	25	400	9	5	morning
path04.mp4	Dirt	11	25	275	9	3	morning
path05.mp4	Dirt	19	25	475	8	3	morning
path06.mp4	Dirt	8	25	200	8	4	morning
path07.mp4	Dirt	9	25	225	8	4	morning
path08.mp4	Dirt	12	25	300	10	2	morning
path09.mp4	Dirt	9	25	225	9	3	morning
path10.mp4	Dirt	13	25	325	10	3	morning
path11.mp4	Dirt	12	25	300	9	2	morning
path12.mp4	Dirt	8	25	200	8	3	morning
path13.mp4	Dirt	13	50	650	15	3	afternoon
path14.mp4	Dirt	16	50	800	14	3	afternoon
path15.mp4	Dirt	15	50	750	15	2	afternoon
path16.mp4	Dirt	22	50	1100	13	2	afternoon
path17.mp4	Dirt	16	50	800	10	2	afternoon
path18.mp4	Dirt	8	50	400	6	4	afternoon
path19.mp4	Dirt	10	50	500	6	3	afternoon
path20.mp4	Dirt	20	50	1000	6	2	afternoon
path01.mp4	Countryside	11	30	330	12	5	morning
path02.mp4	Countryside	7	50	350	12	4	morning
path03.mp4	Countryside	6	50	300	11	4	morning
path04.mp4	Countryside	9	50	450	10	2	morning
path05.mp4	Countryside	8	50	400	10	6	afternoon
path06.mp4	Countryside	14	50	700	10	6	afternoon
path07.mp4	Countryside	8	50	400	8	7	afternoon
path08.mp4	Countryside	12	50	600	8	5	afternoon
path09.mp4	Countryside	8	50	400	9	4	afternoon
path10.mp4	Countryside	13	25	325	8	2	afternoon

the approaches presented in this thesis. In the next subsections, details about the above introduced folders are reported.

2.1 Videos Without Telemetry

This folder contains several videos divided mainly in two sub-folders depending on the type of the environment. In particular, 20 and 10 video sequences have been acquired overflying dirt and countryside environments, respectively. Each sequence in this folder presents many challenging features, including small variations of pitch, roll, and the altitude of the small-scale UAV. These features make the mosaicking process a hard task. In addition, the videos have been acquired in different hours of the day (i.e., morning and afternoon) to obtain sequences with different illumination changes and shadows. As mentioned above, these videos are not associated with any telemetry and can be used only for mosaicking purposes. In Table 2.1, some relevant characteristics of the acquired video sequences are reported.

(a) (b) (c)

Figure 2.2. Examples of different environments in the dataset sequences: (a) urban, (b) dirt, and (c) countryside.

Table 2.2. Telemetry data available in the GPS files.

ID	Telemetry Data	ID	Telemetry Data	ID	Telemetry Data	ID	Telemetry Data	ID	Telemetry Data	ID	Telemetry Data
1	Tick#	12	accelY(M/S2)	23	velD(M/S)	34	errorZ	45	batteryCycleCount	56	MotorSpeed:RBack
2	offsetTime	13	accelZ(M/S2)	24	vel(M/S)	35	error	46	batteryLifePercentage	57	MotorLoad:RFront
3	flightTime(msec)	14	gyroX(degrees/s)	25	quatW	36	accel(M/S2)	47	Gimbal:roll	58	MotorLoad:LFront
4	Longitude	15	gyroY(degrees/s)	26	quatX	37	gyro(degrees/s)	48	Gimbal:pitch	59	MotorLoad:LBack
5	Latitude	16	gyroZ(degrees/s)	27	quatY	38	distancHP(M)	49	Gimbal:yaw	60	MotorLoad:RBack
6	numSats	17	magX	28	quatZ	39	distanceTravelled(M)	50	Gimbal:Xroll	61	NMEA GGA/RMC Strings
7	gpsAltitude(meters)	18	magY	29	Roll	40	directionOfTravel	51	Gimbal:Xpitch		
8	baroAlt(meters)	19	magZ	30	Pitch	41	IMUTemp(C)	52	Gimbal:Xyaw		
9	vpsHeight(M)	20	magMod	31	Yaw	42	batteryTemp(C)	53	MotorSpeed:RFront		
10	relativeHeight	21	velN(M/S)	32	errorX	43	ratedCapacity	54	MotorSpeed:LFront		
11	accelX(M/S2)	22	velE(M/S)	33	errorY	44	remaingCapacity	55	MotorSpeed:LBack		

2.2 Videos With Telemetry

This folder contains videos divided mainly in two sub-folders: mosaicking and change detection. Each of these folders is in turn divided in three sub-folders depending on the type of the environment. In particular, the mosaicking sub-folder contains 4, 4, and 2 video sequences acquired on urban, dirt, and countryside environments, respectively. Each of these videos has been acquired without the presence of any vehicle, person or object. The change detection sub-folder contains videos that have been acquired on the same paths of the previous ones, where some elements above mentioned have been introduced. Usually, dirt and countryside environments represent typical areas surrounding tactical bases or given paths, while urban environments represent common areas in cities or highways. These environments are often more challenging if compared to the first ones due to the presence of some structures (e.g., buildings, trees) that can introduce different issues including parallax errors and shadows. Each video of this folder is linked to a text file containing the telemetry of the UAV flight. In Table 2.2, the available telemetry data is reported.

The different kinds of UAV-based systems currently available provide telemetry information in standard or proprietary format. In any case, these formats are text files that require a very simple parsing process to be used. For this reason, we have chosen one of the most popular standard format, i.e., the NMEA[2]. In addition, we have also extracted information from the small-scale UAV to build a simple proprietary format. In order to give the opportunity for developers to test the GPS coordinate interpolation algorithms during mosaicking or change detection processes, the proprietary format is provided with 10 GPS coordinates linked to each second of the video sequence (i.e., 10 coordinates per second). Differently, the NMEA format is provided with only a GPS coordinate linked to each second of the video sequence

[2]http://www.gpsinformation.org/dale/nmea.htm

Table 2.3. Main characteristics of the videos contained in the folder: Videos With Telemetry → Mosaicking.

Filename	Environment	Length [s]	Frequency [fps]	Total Frames [#]	Telemetry Data (Table 2.2)
path1.mp4	Urban	47	25	1175	From 1 to 60
path2.mp4	Urban	22	30	660	61
path3.mp4	Urban	16	30	480	61
path4.mp4	Urban	100	30	3000	61
path1.mp4	Dirt	144	25	3600	From 1 to 60
path2.mp4	Dirt	138	25	3450	From 1 to 60
path3.mp4	Dirt	75	30	2250	61
path4.mp4	Dirt	67	30	2010	61
path1.mp4	Countryside	39	25	975	From 1 to 60
path2.mp4	Countryside	61	25	1525	From 1 to 60

(i.e., 1 coordinate per second). Concerning the GPS accuracy, the home-made UAV has the standard accuracy, which is typically in the range of ± 4.9 meters, while the DJI Phantom 3 Advanced reports a vertical accuracy of ± 0.5 meters and a horizontal accuracy of ± 1.5 meters.

One of the main novelties of the proposed dataset regards the detection of small objects, on the ground, in videos acquired at low-altitude. In order to make this task comparable with real situations, during the flights performed to acquire the sequences useful to test the change detection algorithms, we have used both static/dynamic vehicles and persons as well as static objects. In particular, we have used objects with different properties, including size, shape, and color.

To test the change detection algorithms, it is possible to use, as ground truth, the part of geo-referenced mosaic (built previously) whose GPS coordinates are linked to those of the current frame within the change detection video sequence. In addition to the telemetry text file, each video is also linked to a metadata text file in which a set of information about the properties of the objects is reported, in detail the following data are available for each object:

- Type, e.g., car, person, box;
- Shape, e.g., rectangular, square;
- Dimension, measured in centimeters (cm);
- Color (with respect to the background), e.g., same range of color, complementary range;
- Movement, e.g., static person or moving car.

Table 2.3 and Table 2.4 report some relevant characteristics of the videos contained in the mosaicking and change detection sub-folders, respectively. Note that, the path4 (urban environment) within the mosaicking folder has two paths, i.e., path4_1 and path4_2, within the change detection folder. This indicates that, on the same path, two different sets of objects have been placed in two different time instants.

Table 2.4. Main characteristics of the videos contained in the folder: Videos With Telemetry → Change Detection.

Filename	Environment	Length [s]	Frequency [fps]	Total Frames [#]	N° of Objects [#]	Type of Object	Object Appearance [s] to [s]
path1.mp4	Urban	48	25	1200	3	• Object 1: Tire; • Object 2: Gas Bottle; • Object 3: Person.	• Object 1: 09 to 18; • Object 2: 12 to 20; • Object 3: 16 to 24.
path2.mp4	Urban	25	30	750	2	• Object 1: Person; • Object 2: Person.	• Object 1: 13 to 16; • Object 2: 14 to 16.
path3.mp4	Urban	25	30	750	2	• Object 1: Person; • Object 2: Person.	• Object 1: 08 to 18; • Object 2: 09 to 19.
path4_1.mp4	Urban	12	30	360	1	• Object 1: Car.	• Object 1: 06 to 09.
path4_2.mp4	Urban	14	30	420	1	• Object 1: Car.	• Object 1: 10 to 11.
path1.mp4	Dirt	125	25	3125	4	• Object 1: Small Box; • Object 2: Small Box; • Object 3: Big Box; • Object 4: Person.	• Object 1: 32 to 42; • Object 2: 69 to 81; • Object 3: 80 to 91; • Object 4: 101 to 111.
path2.mp4	Dirt	123	25	3075	4	• Object 1: Small Box; • Object 2: Big Box; • Object 3: Small Box; • Object 4: Person.	• Object 1: 30 to 40; • Object 2: 83 to 93; • Object 3: 103 to 113; • Object 4: 113 to 122.
path3.mp4	Dirt	66	30	1980	1	• Object 1: Metal Suitcase.	• Object 1: 28 to 32.
path4.mp4	Dirt	60	30	1800	1	• Object 1: Metal Suitcase.	• Object 1: 27 to 37.
path1.mp4	Countryside	40	25	1000	2	• Object 1: Suitcase; • Object 2: Person.	• Object 1: 00 to 09; • Object 2: 11 to 21.
path2.mp4	Countryside	47	25	1175	2	• Object 1: Suitcase; • Object 2: Person; • Object 3: Bag.	• Object 1: 03 to 13; • Object 2: 19 to 26; • Object 3: 36 to 46.

2.3 Camera Calibration

In this folder, the file containing the calibration parameters is provided. The file is stored in eXtensible Markup Language (XML) format, and it contains both the intrinsic camera matrix and the distortion coefficients. The camera matrix C_m is defined as follows:

$$C_m = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

where f_x and f_y are the focal length parameters, and (c_x, c_y) is the optical center. All these parameters are computed by using the method presented in [114]. The distortion coefficients are defined as $\{k_1, k_2, k_3, p_1, p_2\}$, where k_1, k_2, and k_3 are the radial distortion coefficients, and p_1, p_2 are the tangential distortion coefficients. The adopted distortion model, mapping ideal points with coordinates (x, y) to actual coordinates (x_d, y_d), is defined as follows:

$$\begin{aligned} x_d =& x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + \\ & 2p_1 xy + p_2(r^2 + 2x^2) \end{aligned} \tag{2.2}$$

$$\begin{aligned} y_d =& y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + \\ & p_1(r^2 + 2y^2) + 2p_2 xy \end{aligned} \tag{2.3}$$

where r is the distance of (x, y) from the optical center:

$$r = \sqrt{(x - c_x)^2 + (y - c_y)^2}. \tag{2.4}$$

2.4 Benchmark Mosaics

In this folder, the mosaics of all the dataset videos are provided. The motivation is to give a set of reference mosaics created with the baseline testing algorithm, in order to provide a comparative basis for more advanced algorithms. Notice that, since we used a baseline algorithm, some generated mosaics have visual artifacts and, in the worst case, the mosaic contains only a part of the path. This is due to some challenging factors, such as pitch and roll, which are not properly handled by the adopted algorithm.

Chapter 3

The Proposed System

In this section, the architecture of the system is described. In detail, we first provide a description of the system and its logical architecture. Then, the Mosaicking, Change Detection, and Object Detection modules are described in depth.

The proposed system is designed to work in two different modes, namely *Mode 1* and *Mode 2*. When it is used in Mode 1, the UAV performs two flights. During the first flight, the UAV acquires the area of interest and generates a mosaic of it. Moreover, during the acquisition it classifies the objects present in the scene, and associate them with the GPS coordinate flown at the moment of the acquisition. During the second flight, which can occur after minutes, hours, days, or even weeks and years, the UAV acquires again the area of interest, and by comparing the image just acquired with the mosaic previously generated (i.e., the reference mosaic), it can find the changes occurred within the scene. Since during the mosaic generation the UAV performed a classification of the objects, in the change detection step it is possible to know precisely

When used in Mode 2, the UAV performs a flight above the area of interest, and searches for the targets for which the neural network has been trained. In Figure 3.1, the logical architecture of the proposed system is shown. In the depicted architecture, the object surrounded by a red square is the result of the change detection module, while the objects surrounded by a blue square is the result of the object detection module.

3.1 Mosaicking

This section presents the module used for real-time creation of incremental and geo-referenced mosaics of areas of interest acquired at low-altitude. The only input required by the system is a set of GPS coordinates that specifies one or more areas that have to be mosaicked. The proposed mosaicking algorithm presents several innovative contributions compared to the current state-of-the-art. First, to speed-up the feature extraction and matching processes, it adopts the A-KAZE extractor [1]. The recent literature [1, 6, 4] has shown that A-KAZE features are faster to compute than SIFT [63] and SURF [8]. Moreover, these works exhibit much better performance in detection and description than ORB [82]. Second, the mosaicking algorithm implements an automatic method to optimize the acquisition

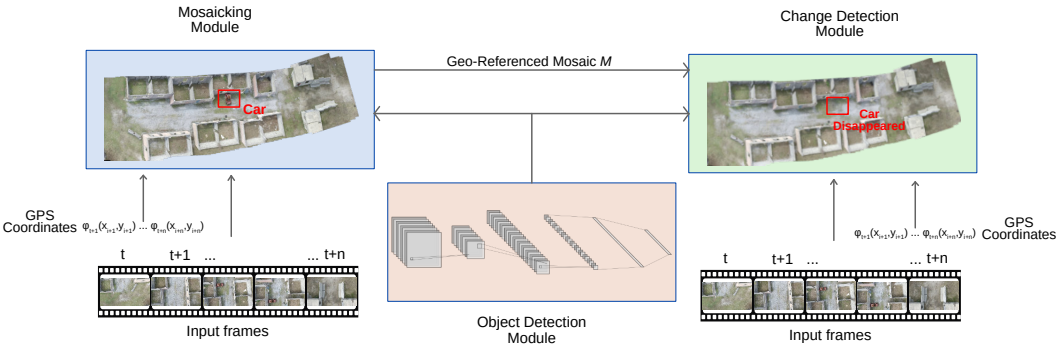


Figure 3.1. Architecture of the proposed system. When used in mode 1, the UAV performs a first flight needed to generate the mosaic of the area of interest (blue square). Then, a second flight is performed by the UAV, and it checks in real-time the changes occurred within the scene (green square). When used in mode 2, instead, the UAV performs a flight and detects the objects of interest within the acquired scene without the need of a mosaic (red squares). The module implemented in mode 2, is also used in mode 1 to classify and geo-referencing the objects in the mosaic or the changes occurred.

rate of the RGB camera based on the telemetry (i.e., speed and height). Third, to speed-up all steps involved in the stitching process, the mosaicking algorithm implements a ROI through which the computation required for the stitching of each new frame on the mosaic is reduced. Fourth, unlike the majority of the mosaicking algorithms known in literature that use RANSAC [32] to perform the geometric transformation stage, the proposed algorithm adopts the rigid transformation [70] that allows the building of mosaics at low-altitude mitigating in part the artifacts due to the parallax error [40]. For low altitudes, the UMCD dataset has been used. Instead, to test the algorithm at high-altitudes we have used the NPU Drone-Map dataset[1].

3.1.1 The Mosaicking Algorithm

The logical architecture of the small-scale UAV based system and the pipeline of the proposed mosaicking algorithm are shown in Figure3.2. The algorithm consists of four main stages each of which is discussed below. Despite the system is designed to work with standalone and client-server architectures, the latter is used to explain properly how the system works.

Background

In the following, let:

$$UAV_{path} = \{\phi_{t+i}(x_{t+i}, y_{t+i}) \mid t \in \mathbb{N} \wedge i \in [1, \ldots, n] \subset \mathbb{N}\} \tag{3.1}$$

be the set of GPS coordinates that defines the area of interest that needs to be mosaicked, where, t is the amount of seconds required by the UAV to reach the area, and n is the second of flight duration within the area. Besides, for each $i \in [1, \ldots, n]$, $\phi_{t+i}(x_{t+i}, y_{t+i})$ is the $t + i^{th}$ coordinate and (x_{t+i}, y_{t+i}) is the pair

[1] http://zhaoyong.adv-ci.com/downloads/npu-dronemap-dataset/

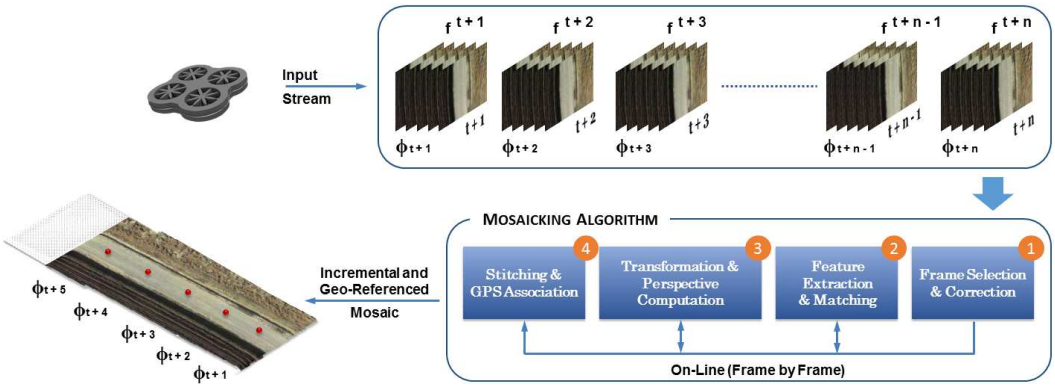


Figure 3.2. The proposed mosaicking algorithm. f^t and ϕ_t are the frames and the linked GPS coordinates provided to the algorithm at each second t, respectively.

$(latitude, longitude)$. Without loss of generality, we can define ϕ_{start} and ϕ_{end} when $i = t + 1$ and $i = t + n$, respectively. In addition, let:

$$F_{TRS} = \{f^{t+i} \mid t \in \mathbb{N} \wedge i \in [1, \dots, n] \subset \mathbb{N}\} \tag{3.2}$$

be the set of frames transmitted from the UAV to the processing unit (local or remote) within the UAV_{path}, where t and n are defined as above. For each $i \in [1, \dots, n]$, $f^{t+i} = \{f_1^{t+i}, f_2^{t+i}, \dots, f_{FPS}^{t+i}\}$ is the set of frames transmitted by the UAV at the second i. The set depends on frame per second (FPS) of the RGB camera. The UAV starts the transmission to the processing unit from the take-off up to the landing. In general, each second $k \in \mathbb{N}$ of transmission is composed of a GPS coordinate, $\phi_k(x_k, y_k)$, and a set of frames, $f^k = \{f_1^k, f_2^k, \dots, f_{FPS}^k\}$.

Frame Selection and Correction

Since the aim of the algorithm is to build the mosaic of the area of interest defined by the UAV_{path}, all the frames transmitted outside of this path (i.e., $f^k \notin F_{TRS}$ for each $k \in \mathbb{N}$) are discarded by the processing unit. The rest of the frames transmitted by the UAV (i.e., $f^k \in F_{TRS}$ for each $k \in [t + 1, \dots, t + n] \subset \mathbb{N}$) are used in part to create the mosaic, while the remaining are discarded again. This is due to the fact that at each second the UAV tends to transmit more frames than ones necessary to create a proper mosaic.

In the proposed algorithm, both flight height and framerate are automatically estimated by exploiting both the UAV telemetry and the characteristics of the CCD of the used acquisition sensor. The automatic estimation of the parameters, especially the flight height, is useful when there is the need of a specific spatial resolution to accomplish a task.

Flight Height Estimation The application of the homography to an image is performed by multiplying the coordinates of the pixels composing the image by the matrix H estimated with the RANSAC method and defined as follows:

$$H = \begin{bmatrix} R_a & R_b & T_x \\ R_c & R_d & T_y \\ W_a & W_b & 1 \end{bmatrix} \tag{3.3}$$

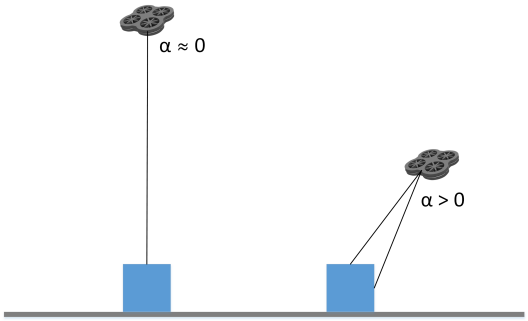


Figure 3.3. Different views of an object placed on the ground with $\alpha \approx 0$ (left) and $\alpha > 0$ (right).

where, $H_R = \begin{bmatrix} R_a & R_b \\ R_c & R_d \end{bmatrix}$ is the *Rotation Matrix*, $M_T = \begin{bmatrix} T_x & T_y \end{bmatrix}^T$ is the *Translation Vector*, and $M_W = \begin{bmatrix} W_a & W_b \end{bmatrix}$ is the *Warping Vector*. This type of transformation allows 8 DoFs and usually it is applied in case of rotation on one or more space axes (i.e., X, Y or Z), such as in panorama applications [15]. Since the UAV movements can be approximated mainly to translations on X and Y axis, it is possible to limit the DoFs of the image transformation by using a similarity transformation, ST, defined as:

$$ST = \begin{bmatrix} R_a & R_b & T_x \\ R_c & R_d & T_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

With respect to the homography, the similarity transformation allows only combinations of translation, rotation, and uniform scaling (4 DoFs), preserving the angles between lines. In this way, the number of transformations applied to an image can be limited, mitigating the errors introduced in case of wrong estimation of the matrix values.

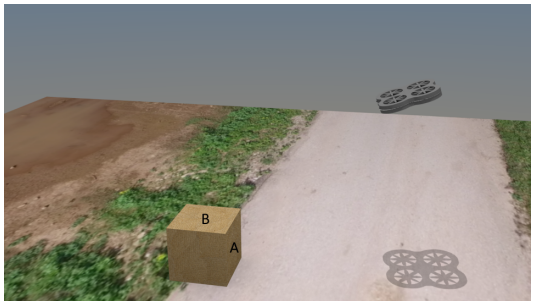
Regarding the perspective, the Angle of View (AoV) of the sensor, at the acquisition instant, has to be considered. Assuming that the lens of the sensor projects rectilinearly the image (i.e., sensor that does not introduce a radial distortion), $AoV = [\alpha_W, \alpha_H]$ can be computed as follows:

$$\alpha_W = 2 \arctan \frac{W_{CCD}}{2f} \tag{3.5}$$

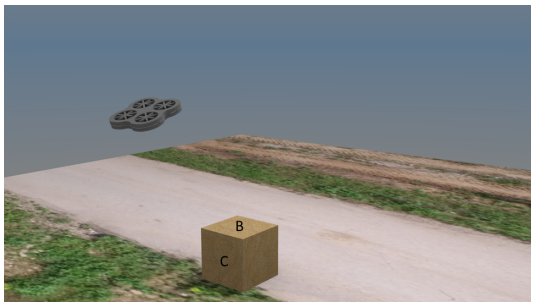
$$\alpha_H = 2 \arctan \frac{H_{CCD}}{2f} \tag{3.6}$$

where, W_{CCD} and H_{CCD} are the width and the height of the CCD sensor, respectively. While, f is the focal length of the sensor.

When the images are orthorectified, α angles between the UAV sensor and an object on the ground are approximately equal to 0. This occurs, as shown in Figure 3.3 (left), because all the projection lines are orthogonal to the projection plane. In our context, where images are not orthorectified, Figure 3.3 (right), we have that the perspective strongly influences the view of the objects in the scene. In particular, we have that the more is the height of the object (or the less is the flying height of the UAV), the more the perspective influences the mosaic construction.



(a)



(b)

Figure 3.4. An example of how a UAV sees an object at a low flight altitude: (a) while it is approaching to the object the faces A and B are acquired; (b) while it flies away from it the faces B and C are acquired.

An example of how these factors influence the construction of the mosaic is represented in Figure 3.4. While approaching to the box (Figure 3.4(a)), the UAV acquires the faces A and B and when leaving the box it acquires the faces B and C (Figure 3.4(b)). As reported highlights two critical aspects: a) the features extracted from faces A and C, despite they are extracted from parts of the same object, cannot be matched; b) three different views of the same object appear, which make impossible to stitch correctly two consecutive frames. Notice that, these issues happen only at low altitude flights since at a high altitude the AoVs described in eq. (3.5) and (3.6) are approximated to 0.

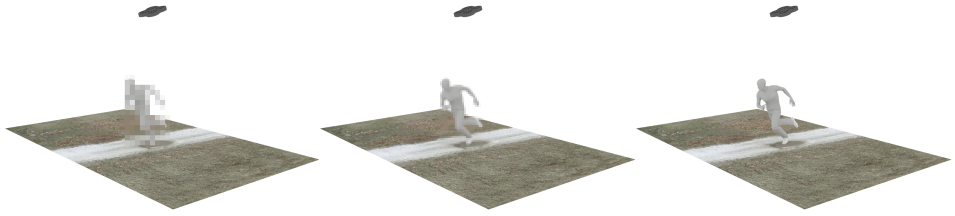
The features extracted from the different faces of an object can be mismatched, and applying subsequently a projective transformation (i.e., homography), some features could be projected to infinity. This introduces a high level of distortion in the image, making impossible to stitch together two consecutive frames. Using the similarity transformation, instead, it is possible to exclude the features mapping to infinity and then to better accomplish the mosaicking task.

The quantity of information captured by the camera mounted on a UAV depends mainly on two factors: the spatial resolution of the sensor and the flight altitude. By using the Johnson's criteria [50], we can determine the minimum flight altitude to accomplish detection, recognition, and identification (DRI) tasks. These criteria state that by acquiring the object with a camera front-facing to it, the minimum number of pixels needed for DRI tasks are 1.5, 6, and 12, respectively. In order to perform one of the DRI tasks, the quantity of pixels per meter must be determined.

An example is introduced to better clarify this concept. Assuming that the critical dimensions for people acquired with a frontal-facing camera is 0.75 meters, the minimum number of pixels per meter required by the DRI tasks can be obtained by computing the ratio between 1.5, 6, 12 pixels and 0.75 meters, respectively. In the second column of Table 3.1, the results of this operation are shown.

Table 3.1. Summary on Johnson's Criteria applied to the DRI tasks for a human being

Discrimination Level (Description)	**Front-Facing Camera** [px/m]	**Bottom-Facing Camera** [px/m]
Detection: An object is present	2	3.02
Recognition: The object type is discerned (person vs car)	8	12.11
Identification: A specific object is discerned (man vs woman)	16	24.23



(a) **(b)** **(c)**

Figure 3.5. Visual representation of (a) detection, (b) recognition, and (c) identification on the basis of Johnson's criteria.

In case that the images are acquired from a UAV and the camera is bottom-facing (i.e., pointing to the ground), the critical dimensions of a subject are smaller. Assuming an average bust depth of 25 cm and an average forearm-forearm breadth of 55.1 cm^2, it is possible to consider as a subject bounding box the rectangle generated by using these two measures. As critical dimension for a person, we can consider the diagonal of the aforementioned bounding box, that in this case is 60.5 cm. As for the frontal-facing camera, it is possible to obtain the minimum number of pixels per meter by performing the ratio between 1.5, 6, 12 pixels, and 0.605 meters, respectively. In the third column of Table 3.1, the results of these operations are shown. In Figure 3.5, a visual representation of the DRI tasks is depicted, while in Figure 3.6 the critical dimensions of a subject acquired by a UAV are shown.

By assuming the pinhole camera model, it is possible to find a relation between the width W and the length L of the observed scene, and the width W_{CCD} and the

[2] https://msis.jsc.nasa.gov/sections/section03.htm

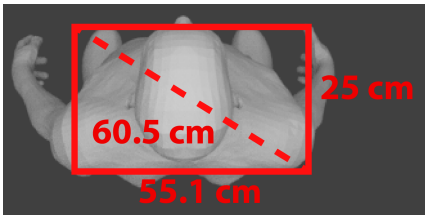


Figure 3.6. Representation of critical dimensions of a subject acquired by a UAV.

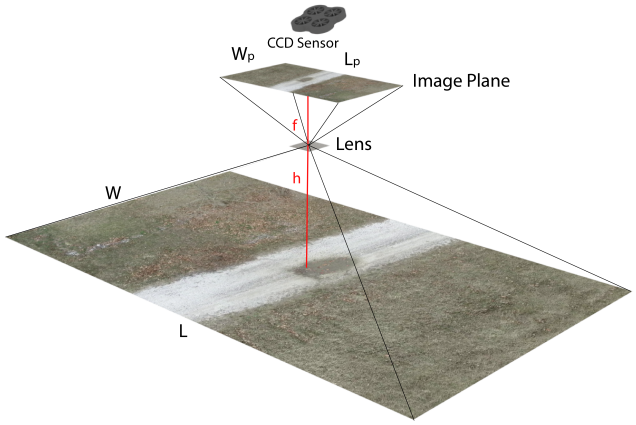


Figure 3.7. Representation of the parameters needed to correlate the Johnson's criteria with the flight height.

height H_{CCD} of the CCD sensor, as follows:

$$W = \frac{hW_{CCD}}{f}[mm], \qquad L = \frac{hL_{CCD}}{f}[mm] \tag{3.7}$$

where, h is the flying height and f is the focal length of the camera. By considering the spatial resolution of the camera, i.e., $W_p/W[pixel/mm]$ and $L_p/L[pixel/mm]$, where W_p and L_p are the number of pixels on the image plane corresponding to W and L millimetres in the observed scene, and knowing the CCD pixel dimensions k_w, k_h $[\mu m]$, the following equation can be expressed:

$$\frac{W_p}{W} = \frac{fW_p}{hW_{CCD}} = \frac{fW_p}{hk_wW_p} = \frac{f}{hk_w} \tag{3.8}$$

The same relation holds for L_p. In Figure 3.7, the parameters used for correlating the Johnson's criteria with the flight height are depicted. Assuming further a squared CCD sensor, i.e., $k_w = k_h = k$, it is possible to use the Johnson's criteria to determine the maximum flight height h_{max} to perform a specific task:

$$h_{max}(task) = \frac{f}{k\ SRJ(task)} \tag{3.9}$$

where, $SRJ(task)$ is the minimum spatial resolution required for a task, and $task \in \{Detection, Recognition, Identification\}$.

To explain in detail the eq. 3.9, an example for each of the DRI tasks is provided. Suppose that we want to determine the maximum height needed to detect, recognize,

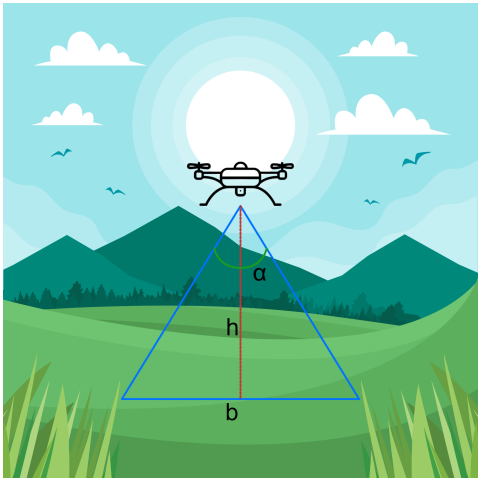


Figure 3.8. Parameters used for computing fr_{min}. The parameter h is the flight height, α is the FoV and b is the length of the observed scene.

and identify a person lying on the ground, such as in rescue operations. This case is amenable to a front facing acquisition, so we can use the DRI values provided by the second column of Table 3.1. Assuming a video sensor with a focal length $f = 1.7mm$, and a CCD pixel size of $k = 5.97\mu m$, the DRI heights computed by means of the eq. 3.9 can be defined as:

$$h_{max}(D) = \frac{1.7\ [mm]}{5.97\ [\mu m] \cdot 2\ [pixel/m]} = 121.95\ meters \tag{3.10}$$

$$h_{max}(R) = \frac{1.7\ [mm]}{5.97\ [\mu m] \cdot 8\ [pixel/m]} = 30.48\ meters \tag{3.11}$$

$$h_{max}(I) = \frac{1.7\ [mm]}{5.97\ [\mu m] \cdot 16\ [pixel/m]} = 15.24\ meters \tag{3.12}$$

Instead, to perform DRI tasks with UAV of a standing person (i.e., walking, running) the values provided in the third column of the Table 3.1 must be used:

$$h_{max}(D) = \frac{1.7\ [mm]}{5.97\ [\mu m] \cdot 3.02\ [pixel/m]} = 80.76\ meters \tag{3.13}$$

$$h_{max}(R) = \frac{1.7\ [mm]}{5.97\ [\mu m] \cdot 12.11\ [pixel/m]} = 20.14\ meters \tag{3.14}$$

$$h_{max}(I) = \frac{1.7\ [mm]}{5.97\ [\mu m] \cdot 24.23\ [pixel/m]} = 10.06\ meters \tag{3.15}$$

Frame Rate Estimation As it is necessary to always ensure an overlapping area between pairs of consecutive frames, an appropriate frame rate fr, depending on the UAV speed v and flight height h must be computed and guaranteed. Some considerations regarding the flight speed can be done:

(a) The higher is the UAV speed v, the higher must be the minimum frame rate fr_{min};

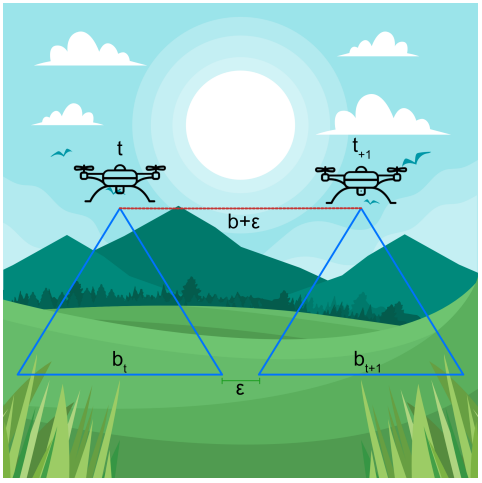


Figure 3.9. Example in which there is no overlap between two consecutive frames.

(b) If the flight height h increases, the minimum frame rate fr_{min} must decrease as a larger area is acquired;

(c) If the Field of View (FoV) of the sensor increases, the minimum frame rate fr_{min} must increase (for the same reason of point b).

Let us assume a UAV having the camera pointed orthogonally towards the ground, as shown in Figure 3.8. Since the steps we are going to illustrate are the same for both sensor dimension, the calculation are performed only on one dimension.

As it is possible to see, we have that

$$b = 2\left(h \cdot tanh\left(\frac{\alpha}{2}\right)\right) \tag{3.16}$$

Let us define b_t and b_{t+1} as the area acquired at time instants t and $t+1$, respectively. If the UAV performs a movement of at least b meters between t and t_{t+1}, as shown in Figure 3.9, there is no overlap between the two acquired frames.

Since the time needed to move by b meters is b/v, we have that $fr = v/b$. Consider the overlap parameter $s \in [0, 1]$, where 0 means that the frames are totally disjointed and 1 means that the frames are totally overlapped. In order to assure a minimum overlap between two consecutive frames, the UAV must flight at most $b(1 - s)$ meters, as shown in Figure 3.10.
So, the minimum frame rate can be defined as:

$$fr_{min} = \frac{v}{b(1-s)} \tag{3.17}$$

Let us consider as running example a camera with $fr = 60fps$ and $FoV = 60°$. By varying the s value, the minimum frame rate fr_{min} can assume the values reported in Figure 3.11.

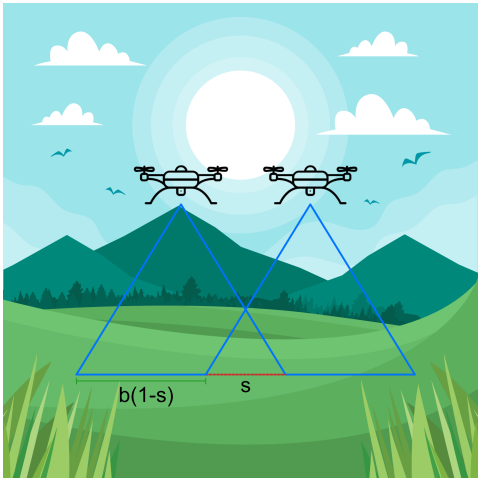


Figure 3.10. Example of frames usable for mosaic generation. As it is possible to see, the minimum overlapping part is assured.

Feature Extraction and Matching

Let M_j be the mosaic built up to the second j and let $\hat{f}_s^{j+1}$, with $j+1 \in [t+1, \ldots, t+n] \subset \mathbb{N}$ and $s \in [1, \ldots, FPS] \subset \mathbb{N}$, the current selected frame, at the second $j+1$, to be added to the mosaic. The main steps to built the new mosaic, $M_j \cup \hat{f}_s^{j+1}$, are the feature extraction and matching processes. In general, the features extracted from each current frame should be compared with those extracted from the whole mosaic to establish where the current frame has to be placed. Since the size of the mosaic grows over time, the comparison stage tends to become unmanageable after a certain period of time. With the aim to avoid such a issue, the proposed system uses a ROI to extract the features from the mosaic. The ROI tracks the last frame added to the mosaic and delimits, to a region surrounding it, the feature extraction process. A ROI centred on the last frame and sized three times than the size of a frame is sufficient to ensure the proper execution of the mosaicking algorithm. By the ROI the adding of a new frame takes a constant-time, no more dependent on the increasing size of the mosaic. Notice that the ROI concept is not new, but it is worth describing it due to the its effectiveness in increasing the system performance. The proposed algorithm uses A-KAZE, instead of the most popular extractors, such as SIFT, SURF or ORB. This is due to the fact that A-KAZE adopts both the Fast Explicit Diffusion (FED) embedded in a pyramidal framework and the Modified-Local Difference Binary (M-LDB) descriptor in order to speed-up feature detection in non-linear scale space and to exploit gradient information from the non-linear scale space, respectively. These aspects make A-KAZE an optimal compromise between speed and performance with respect to the current literature [1].

The keypoints extracted from M_j and $\hat{f}_s^{j+1}$ are used to detect the overlapping region between them. Let $\mathcal{X}_{M_j} = \{\alpha_1, \ldots, \alpha_h\}$ and $\mathcal{X}_{\hat{f}_s^{j+1}} = \{\beta_1, \ldots, \beta_t\}$ be the set of keypoints extracted by A-KAZE from M_j and $\hat{f}_s^{j+1}$, respectively. With the aim of finding the correspondence between the keypoints in $\mathcal{X}_{M_j}$ with those in $\mathcal{X}_{\hat{f}_s^{j+1}}$ a simple Brute Force Matcher (BFM) algorithm is used [88]. This algorithm performs an exhaustive search between the two sets of keypoints and matches only those keypoints

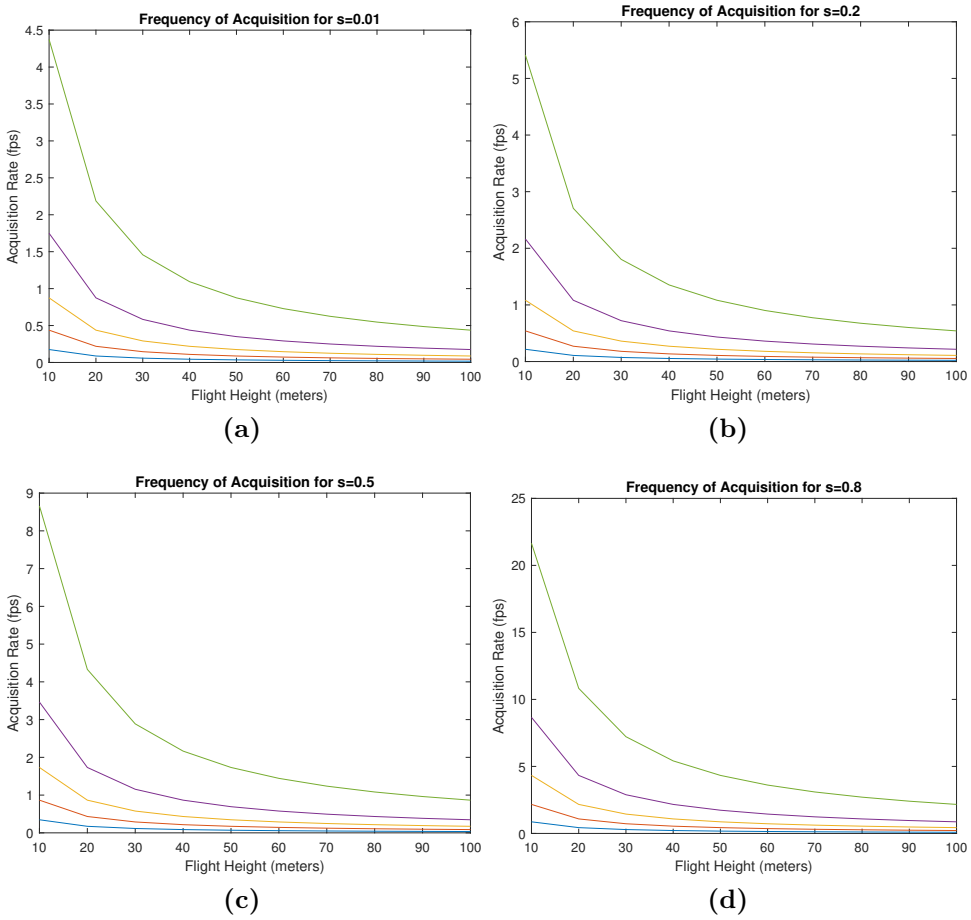
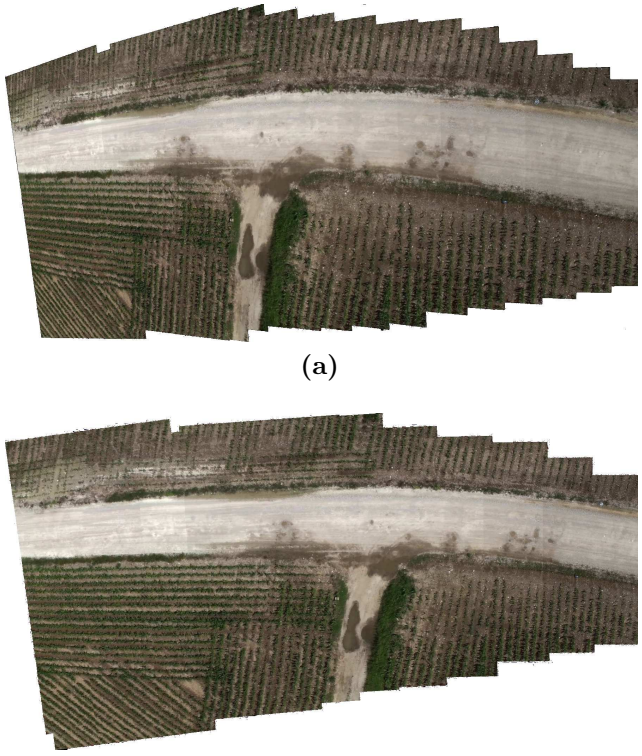


Figure 3.11. Frame rate obtained with an s value of a) 0.01, b) 0.2, c) 0.5 and d) 0.8.

that have an identical pattern (i.e., local structure of the pixels). Formally, at the end of the process, the algorithm generates two sub-sets $\hat{\mathcal{X}}_{M_j} = \{\alpha_{h_1}, \ldots, \alpha_{h_m}\} \subseteq \mathcal{X}_{M_j}$ and $\hat{\mathcal{X}}_{\hat{f}_s^{j+1}} = \{\beta_{t_1}, \ldots, \beta_{t_m}\} \subseteq \mathcal{X}_{\hat{f}_s^{j+1}}$ where for each $k \in \{h_1, \ldots, h_m\}$ exists a single $j \in \{t_1, \ldots, t_m\}$ such that $\alpha_k \equiv \beta_j$. As well-known, the two sub-sets have the same cardinality.

Transformation and Perspective Computation

Once obtained the corresponding keypoints (i.e., $\hat{\mathcal{X}}_{M_j}$ and $\hat{\mathcal{X}}_{\hat{f}_s^{j+1}}$) between the two frames, the system must compute the geometrical transformation by which the keypoints of the current frame, $\hat{f}_s^{j+1}$, are collimated with ones of the mosaic, $\mathcal{X}_{M_j}$, within the reference system of the latter. This transformation is subsequently used on each pixel of the frame to stitch it over the mosaic. In literature, the RANSAC algorithm to calculate the homography transformation is considered the reference approach. It consists in using the corresponding keypoints to iteratively estimate the parameters of a mathematical model by which to perform the geometric projection of each pixel between the two images. Despite this, as shown in Figure 3.12a, the homography transformation can produce a high level of distortions especially when it



(a)

(b)

Figure 3.12. Geometric transformation: (a) homography transformation by RANSAC algorithm, (b) rigid transformation.

is applied on images acquired a low-altitude. In particular, the mosaic can present an unreal curvature. This is due to the fact that the homography transformation matrix has 8 degrees of freedom, hence at least 4 corrected correspondences are required to build a proper mosaic. In the proposed mosaicking algorithm, the acquired images can be considered as a linear scanning of the ground surface, therefore a transformation with less degrees of freedom can be adopted. For this reason, the rigid transformation matrix that has only 4 degrees of freedom is implemented [70]. The reference example reported in Figure 3.12b shows the goodness of the obtained results. The majority of the UAV based systems treat video sequences acquired at high altitude, or propose a orthorectification pre-processing step at the expense of the real-time processing [116] thus avoiding this type of issue. The last step of the module is to merge the pixels of the mosaic, $\mathcal{X}_{M_j}$, with the transformed pixels of the frame, $\Gamma(\hat{f}_1^{j+s})$, to obtain a new pixel matrix, $\mathcal{X}_{M_j} \cup \Gamma(\hat{f}_s^{j+1})$.

Stitching and GPS Association

The acquisition of the GPS coordinates is performed following the NMEA[3] format, one of the most widespread standards for the transmission of position data. Current commercial GPS transmitters provide one or more position data per second, however in the latter case a good practice is to derive a single information per second to

[3] http://www.nmea.org/

reduce the intrinsic error due to the acquisition process. Since the construction of a mosaic can require more frames per second, this means that only the first of the n frames for second acquired by the RGB camera is associated to a GPS coordinate, the rest of the $n-1$ frames, if added to the mosaic, has to be associated to coordinates inferred by ones previously acquired. Actually, once obtained two coordinates of the first frame of two consecutive seconds, then the coordinates of the remaining frames of the first second can be derived by adopting a simple linear interpolation. Let $\phi_j(x_j, y_j)$ and $\phi_{j+1}(x_{j+1}, y_{j+1})$ be the GPS coordinates acquired and associated with the frames $\hat{f}_1^j$ and $\hat{f}_1^{j+1}$, respectively ($s = 1$ in both cases since they are the first frames of each second). In addition, considering $\hat{f}_1^j$ belonging to the mosaic M_j and $\hat{f}_1^{j+1}$ the current frame. Then, the coordinate of any frame added to the mosaic between them can be derived as follows:

$$x_k = x_j + \frac{k}{FPS}(x_{j+1} - x_j) \ , \ y_k = y_j + \frac{k}{FPS}(y_{j+1} - y_j) \tag{3.18}$$

where, x_k and y_k are the interpolated *latitude* and *longitude*, respectively, of the new GPS coordinate $\phi_k(x_k, y_k)$ associated to the frame $\hat{f}_k^j$. Moreover, k specifies the coordinate of which frame needs to be computed, finally, FPS is the frames per second of the sensor. The current version of the system performs the mosaicking algorithm in on-line mode. This means that when the system acquires a new GPS coordinate, it also considers the previous acquired one, computes the interpolation process and associates the interpolated coordinates to the linked frames within the mosaic. Each GPS coordinate (acquired or interpolated) is anchored to the barycentre of the linked frame. This last is a main aspect to enable the system with a wide range of tasks. Once that the GPS coordinate has been linked to the new frame, the gain compensation between this latter and the mosaic is performed by using the multi-band blending [15]. This assures that there will be no seams when the new frame is added to the current mosaic.

3.2 Change Detection

In this section, a novel robust and real-time change detection system for low-altitude flights is proposed. The pipeline, takes as input a geo-referenced mosaic and a video stream with its associate GPS stream sent by a small-scale UAVs during the second reconnaissance flight of the area of interest. Then, the change detection is performed between the frames of the video stream and the corresponding part of the geo-referenced mosaic, extracted by comparing the GPS coordinates. The proposed system uses a novel pipeline comprising sliding window techniques, RGB-LBP operator and histograms similarity. Differently from other change detection algorithms, whose aim is to extract the exact silhouette of the found novelties, in this thesis work the scope is to identify the change (e.g., people, vehicles) among the highest number of frames. This choice is due to the fact that once the change has been found, advanced algorithms can be used for its classification (e.g., [64]). In addition, the proposed system has to deal with some well-known problems. Firstly, it does not use orthorectified images. This means that high objects (e.g., trees, buildings) may introduce a perspective error that influences the detection of changes. Secondly, high-altitudes mitigate several factors, such as noise and alignment errors.

At low-altitudes, a misalignment between two images can irreversibly compromise the detection due to the generated image artifacts. Moreover, at high-altitudes tasks like surveillance [66, 75], search and rescue [10, 90, 76], and tracking [111, 33, 34] cannot be performed, while the proposed system is designed to handle such situations.

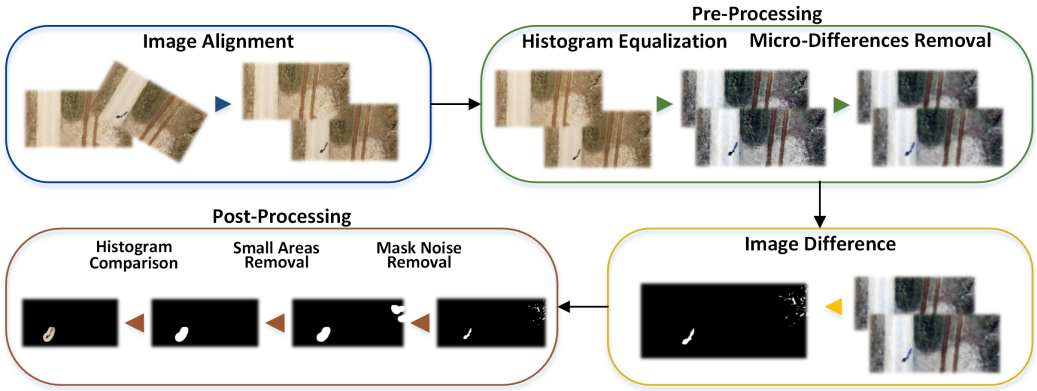


Figure 3.13. Logical architecture of the proposed system.

In this section, the pipeline of the system shown in Figure 3.13 is described through a running example. As input, a geo-referenced mosaic of the interest area is required. For clarity, we will call T and Q, respectively, the train image, or the reference image extracted from the mosaic and the query image, or the image in which we want to detect changes. The pipeline will output an image O, in which there are present the changes and a list $L = R_1, \ldots, R_n$ of bounding boxes. For each R_i, in the sub-image $Q(R_i)$ will be present at least one change with respect to the same sub-image $T(R_i)$.

Image Alignment The first step of the proposed pipeline is the alignment between T and Q. A first gross alignment is performed by using the GPS coordinates sent by the UAV in order to extract the corresponding part of the georeferenced mosaic. This allows to avoid the comparison between the frame received in real-time from the UAV and the whole mosaic. The second step of the alignment consists in using features robust to rotations, scale changes and translations to further align T and Q. In the pipeline, we chose A-KAZE [1] features since they are computed faster than SURF [8] and SIFT [63] and they are also detected and described with much better performance than these methods, including ORB [82].

Pre-Processing The pre-processing is performed in order to reduce the number of false positives/negatives during the difference between images.

The first preprocessing operation is the Histogram Equalization, which is performed to reduce the illumination differences between T and Q. Figures 3.14(a)(b) show the result of this operation.

The second pre-processing step is the micro-differences removal. This process removes all the small differences between T and Q such as rippling water and grass moved by the wind. In order to remove these false changes, a local operator has been implemented. This operator corrects the color of a pixel relying on the average

color of the neighborhood pixels. This is performed by using a sliding window W_a and for each pixel p of T and Q, W_a is translated so that its center corresponds to p coordinates. Hence, each time W_a is moved, the averages of the RGB channels within the window are computed and stored in the RGB channels of p. Figures 3.14(c)(d) depict the results obtained.

Difference between Images At this point, the difference between T and Q is performed. In detail, the difference is performed by using a threshold difference and the RGB-LBP operator. Before applying the threshold difference, both T and Q are converted to grayscale, and we call these images T_{gray} and Q_{gray}. Then, the threshold difference is performed by respecting the following condition:

$$D_{i,j} = \begin{cases} 0 & \text{if } |T_{gray_{i,j}} - Q_{gray_{i,j}}| < T_{diff} \\ 1 & \text{otherwise} \end{cases} \tag{3.19}$$

where T_{diff} is the threshold value used to consider the pixels T_{gray} and Q_{gray} different, and $D_{i,j}$ is the pixel resulting from this difference. The value T_{diff} is chosen according to the difference of illumination between T and Q. The more the illumination is different, the more T_{diff} will be a high value. This is due to the fact that a difference between two values having a high distance will produce a high value as result, easily overcoming a low threshold.

If $D_{i,j}$ is 0, no further actions are performed. Otherwise, the RGB-LBP is used to check if the pixel *(i,j)* is a real change. Given an image I three binary strings, one for each colour channel, are computed. Let S_R, S_G, and S_B be these strings, they are computed in the following way. For each pixel pc in I:

$$S_{Channel_{pc}} = \begin{cases} 0 & \text{if } I(W_l)_{j,k}[Channel] < p_c[Channel] \\ 1 & \text{otherwise} \end{cases} \tag{3.20}$$

where $k = ((i-1) \bmod Width_{W_l}) + 1$, $j = \lfloor (i-1)/Width_{W_l} \rfloor + 1$ and W_l are the neighborhood pixels of pc. The three binary string are computed for both T and Q, and then they are compared by using the Hamming Distance. First, we assure that the strings of T and Q are of the same length. In the case they are not, the strings are considered different. On the contrary, we proceed with the Hamming Distance computation. Also in this case a threshold is used, since it is nearly impossible to have two identical binary strings for two different images. The value of this threshold has been chosen with the same criteria of T_{diff}.

Let c be the number of changes between two compared strings. If $c > T_H * |s|$, where T_H is the Hamming Distance threshold and $|s|$ is the length of a binary string, then the strings are considered different. If the pixels are considered changed with both threshold difference and RGB-LBP methods, it is supposed that a real change has occurred between T and Q. As a result of these steps, we obtain a binary mask M_{diff}, as shown in Figure 3.14(e).

Post Processing The last part of the pipeline regards post processing operations, which are used to remove any false positive (if present). The first post processing step consists in removing the noise from M_{diff}. In M_{diff} there could be white pixels

surrounded by a big amount of black pixels and vice-versa. We call these pixels *isolations*. Usually, isolations are false positive due to small differences not removed during the pre-processing step. To remove isolations, a sliding window technique is used. In detail, a new binary mask N is used, and each pixel $N_{i,j}$ is modified in the following way:

$$N_{i,j} = \begin{cases} 0 & \text{if } n_{black} > T_{iso} \cdot (Width_{W_i} \cdot Height_{W_i}) \\ 255 & \text{otherwise} \end{cases} \tag{3.21}$$

where n_{black} is the number of black pixels surrounding $N_{i,j}$ and $T_{iso} \in [0, 1]$ is a threshold value. In Figure 3.14(f), the result of this operation is shown.

The second post processing step consists in removing small areas. Once isolations are removed, the minimal bounding boxes are computed. A minimal bounding box contains a set of contiguous white pixels in N, and they form the set $L = R_0, \ldots, R_n$. Let T_{area} be a threshold in the interval $[0, 1]$. The value T_{area} is chosen with respect to the size of the acquired frame. The more is the spatial resolution of the image, the more this value should be. All the bounding boxes $R_i \in L$ having a ratio between the area of R_i and the area of N less than T_{area} are removed from L. Moreover, all the pixels of $N(R_i)$ are set to black. This parametrization is due to the fact that small areas in big images have a different weight than small areas in small images.

The last post processing step consists in calculating the histogram similarity. This last step is performed to remove further false positives. First, the part of image contained in the bounding boxes R_i is extracted from both T and Q, respectively $T[R_i]$ and $Q[R_i]$. Then, $T[R_i]$ and $Q[R_i]$ are converted to grayscale and their histograms, H_{T_i} and H_{Q_i} is computed. Finally, the two histograms are compared by using the Bhattacharyya distance [11] B, defined as follows:

$$B(h_1, h_2) = \sqrt{1 - \frac{1}{\overline{h_1} \cdot \overline{h_2} \cdot \beta^2} \sum_i \sqrt{h_1(i) \cdot h_2(i)}} \tag{3.22}$$

where β is the number of bins and $\overline{h_k} = \frac{\sum_i (h_k(i))}{N}$. The Bhattacharyya distance provide as result a similarity value $B \in [0, 1]$, and the most the similarity value is closer to 0, the most the histograms are similar. If B is less than a threshold T_B, all the pixel in $N[R_i]$ are set to 0. The best threshold value T_B for B has been empirically found during the experiments.

In Figure 3.14(g) the result of this last post processing operation is shown.

The last step of the pipeline consists in the application of the mask N to the image Q, highlighting the detected changes. In Figure 3.14(h), the final result is depicted.

3.3 Object Detection

In the last decade, several Deep Learning models have been proposed for object detection. The majority of these models are extension of CNNs, and since the object detection consists in classification and localization of the object within the image, the deep object detection models consist in two parts. The first part is, precisely, a CNN used for classifying the object present in the image. The second part, instead,

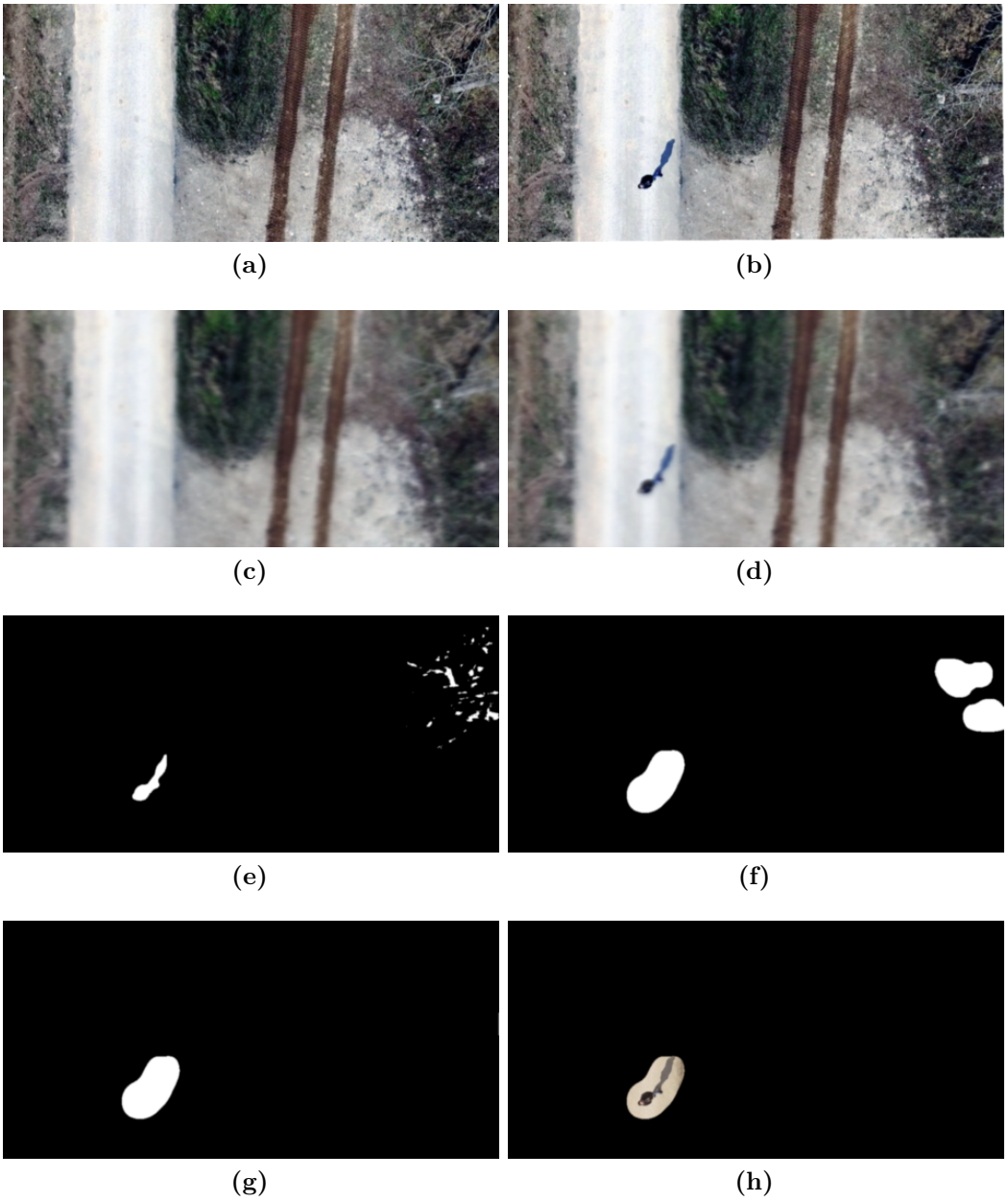


Figure 3.14. Running example on T and Q of the proposed system. Figures (a) and (b) show the histogram equalization. Figures (c) and (d) show the micro-differences removal. Figure (e) and (f) show, respectively, the binary mask obtained from the image difference and isolation removal. Finally, images (g) and (h) show the final mask and its application on Q.

is the algorithm that proposes the bounding boxes containing the objects. Now, the concepts behind the CNN such as convolution and pooling are briefly described to provide background knowledge for better understanding the deep object detection models.

Convolutional Neural Network

The CNNs [26] are a specialized kind of neural network designed for processing data that has a grid-like topology. Typical grid-like data are the time series, which can be seen as a 1D grid taking samples at regular time intervals, and images, which can be though as a 2D grid of pixels. The name of this kind of network indicates that it uses the mathematical operation named *convolution*. So, convolutional networks are neural networks that use the convolution operation in at least one of their layers [38]. A typical layer of a CNN consists of three steps: convolution, activation, and pooling. Now we are going to briefly describe these components.

Convolution In general, the convolution operation is a mathematical operation on two functions of a variable, i.e. x and w, and consists in integrating the product between x and w translated by some value over time. Formally:

$$s(t) = \int x(a)w(t-a)da \tag{3.23}$$

where t is the time instant. Typically, the convolution is denoted in the following way:

$$s(t))(x * w)(t) \tag{3.24}$$

In CNNs terminology, the first argument of the convolution (i.e., x) is referred as the *input* of the layer, while the second argument (i.e., w) is the *kernel*. The output of the convolution, or the output of the layer, is referred as *feature map*. Since the data processed by a computer, it will be discretized so it is possible to define the discrete convolution operation:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{3.25}$$

Trivially, it is possible to use the discrete convolution on more than one axis at time. For example, if we use a two-dimensional image I as input for the layer, we probably also want to use a two-dimensional kernel K. Formally:

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n) \tag{3.26}$$

In Figure 3.15, an example of application of 2D convolution is shown.

By setting the kernel size smaller than the input, two main advantages are obtained. The first advantage is that, with respect to traditional neural network, less computation is performed. Suppose that we have 5×5 input features and a 3×3 output features. In a standard fully connected network, we will have a matrix of weights of $25 \times 9 = 225$ parameters. With the convolution operation, and considering

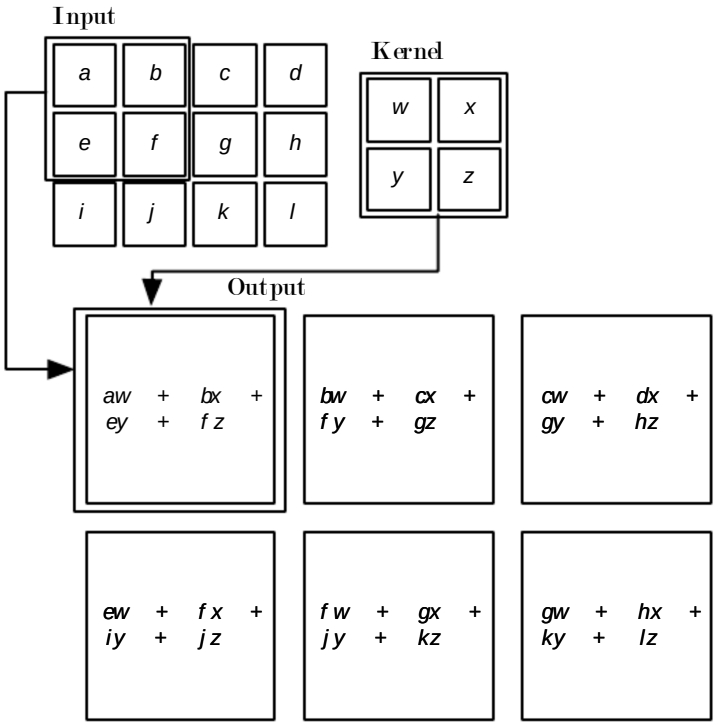


Figure 3.15. An Example of 2D convolution. The output is restricted to only positions where the kernel is completely within the image.

only 'valid' kernels (i.e, the kernel lie entirely within the image), only 9 parameters are computed. In Figure 3.16 and Figure 3.17, respectively, examples of weights computation in a standard fully connected network and with convolution are shown. The second advantage is that since fewer parameters are stored, we have a reduction in memory requirements of the model and an improvement in its statistical efficiency. Moreover, by setting the kernel size smaller than the input, when an image is processed it is possible to detect small but meaningful features, such as edges, lines, circles, etc.

Pooling As result of the first stage, we obtain a set of linear activations. In the second step, these linear activation are used as input for a non-linear activation

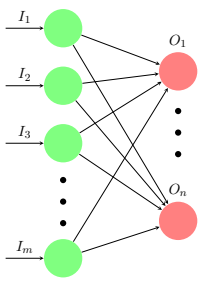


Figure 3.16. Example of input/output connection in a standard fully connected network.

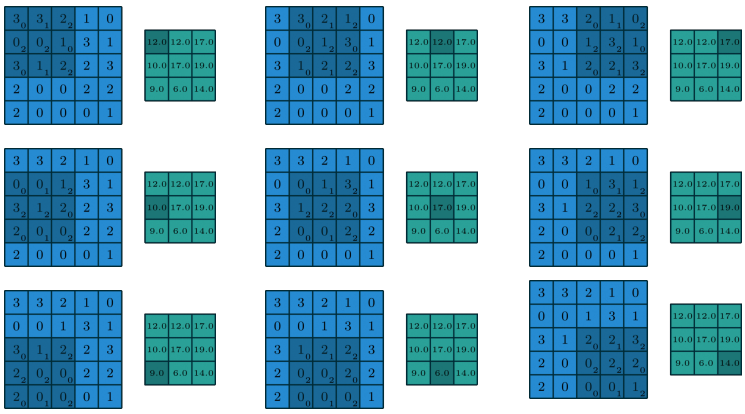


Figure 3.17. Example of matrix weight obtained with the convolution operation.

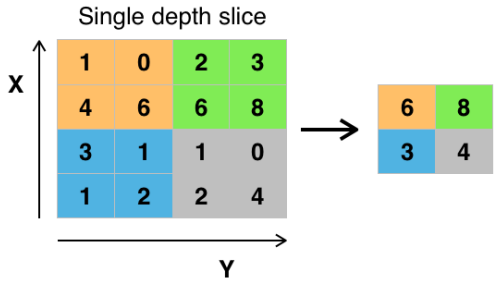


Figure 3.18. Example of max pooling.

function such as Rectified Linear Units (ReLU). In the last step, the output of the layer if further modified through a pooling function. A pooling function replaces the output at a certain location by considering the value of the neighbourhood outputs. For example, the *max pooling* operation [118] provide the maximum output within a rectangular neighbourhood. Other used functions include the average, the L^2 norm, or a weighted average based on the distance from the central pixel. In Figure 3.18, an example of max pooling is shown.

Independently from the chosen function, the pooling is used to make the feature representation invariant to small translations of the input. This is a useful property, since if the input is translated by a small amount, most of the values obtained from the pooling do not change. If pooling over spatial region produces invariance to translation, pooling over the outputs of separately parametrized convolutions makes the features to learn which transformation to become invariant. In Figure 3.19, an example of pooling allowing learning the rotation transformation is shown.

Since pooling summarizes the output of a neighbourhood, in a CNN it is possible to use fewer pooling units than activation functions. Moreover, pooling is essential in for handling input varying in size.

3.3.1 Deep Models for Object Detection

In this section, the models of deep object detection on which we based our improved network are reported. One of the first deep neural network model for object detection

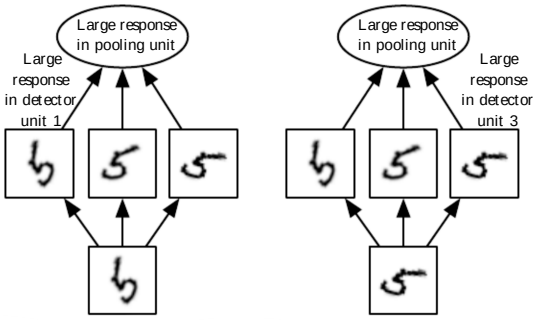


Figure 3.19. Example of pooling allowing learning rotation.

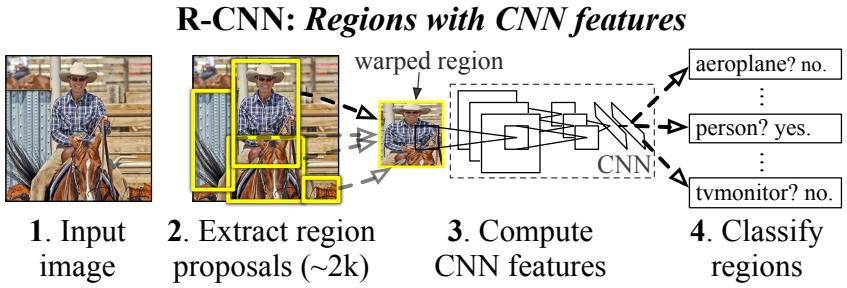


Figure 3.20. Architecture of R-CNN.

is the Regional Convolutional Neural Network (R-CNN) [36]. The network takes as input an image, and provide as output a set of bounding boxes together with the corresponding labels of the objects contained within them. The bounding boxes, or region proposal, is performed by using a method called Selective Search [98], allowing to generate about 2000 regions within the input image. At a high level, Selective Search looks at the image through windows of different sizes, and for each size tries to group together adjacent pixels by texture, colour, or intensity to identify objects. Once the proposals are generated, R-CNN warps the region to a square of standard size, and passes it to a modified version of AlexNet [52] to compute CNN features. On the final layer, a SVM is used to perform classification, and a simple linear regression is used to generate tighter bounding boxes. In Figure 3.20, the steps performed by R-CNN are shown.

Since the R-CNN requires a forward pass of the CNN for every region proposed, it is very slow in detection (about 50 seconds). Moreover, it is hard to train since the CNN, the classifier, and the regression model have to be trained separately. Both these problems have been solved with the improvement of R-CNN, namely Fast R-CNN [35]. The first improvement consists in using a ROI pooling algorithm. The latter allows to share the forward pass of a CNN across the subregions of the input image. In this way, the CNN features are obtained for each region by selecting a corresponding region from the CNN feature map. This takes only one forward pass instead one for each region proposal. The second improvement consists in combining

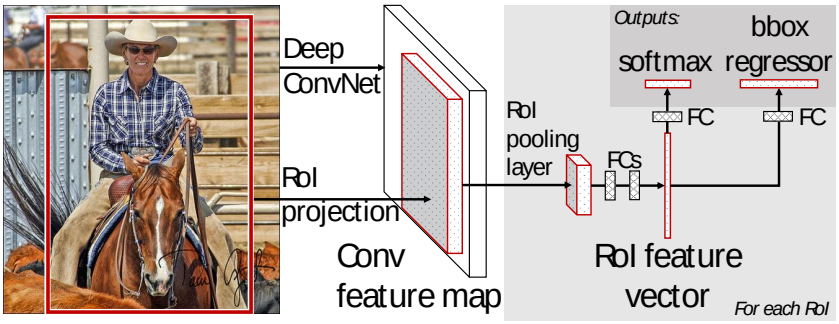


Figure 3.21. Fast R-CNN architecture. With respect to R-CNN, in this model the CNN, the classifier, and the linear regression are merged into one single model.

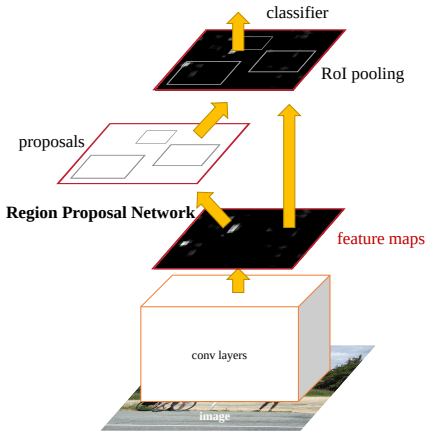


Figure 3.22. Faster R-CNN model architecture. As it is possible to see, the extracted CNN features are used both for region proposal and for classification.

all models into one single network. In Figure 3.21, the Fast R-CNN architecture is shown. In detail, the SVM classifier has been replaced with a softmax layer on top of the CNN, and a linear regression layer has been added in parallel to the softmax layer to output bounding box coordinates.

With the Fast R-CNN, the detection time drastically reduced from about 50 seconds to about 2 seconds. Despite this, the region proposal still remains a bottleneck. This is due to Selective Search algorithm, which is a fairly slow process. A solution is proposed in [79] with a network called Faster R-CNN, in which a cost-free region proposal algorithm is defined. In this model, the CNN features are used both for classification and region proposal, allowing to train only one CNN. In detail, a Region Proposal Network (RPN) is used, as shown in Figure 3.22. The RPN is implemented as a fully convolutional network within a convolutional layer, and it works by passing a sliding window over the CNN feature map and at each window, outputting k potential bounding boxes and scores for how good each of those boxes is expected to be. The k bounding boxes are generated by considering common aspect ratios. For instance, a vertical rectangular box may contain a person, while a horizontal rectangular box may contain a car. These are called the *anchor* boxes, and for each such anchor box one bounding box and score per position in the

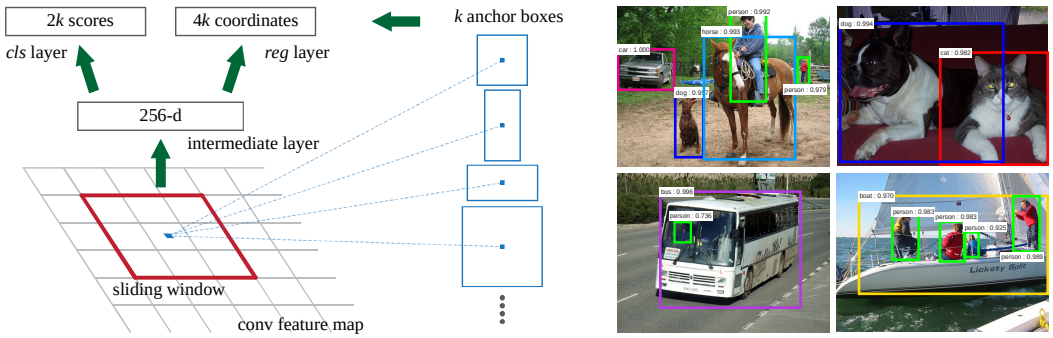


Figure 3.23. Representation of the usage of RPN. The RPN generates the anchor boxes, which are used to detect objects within the image.

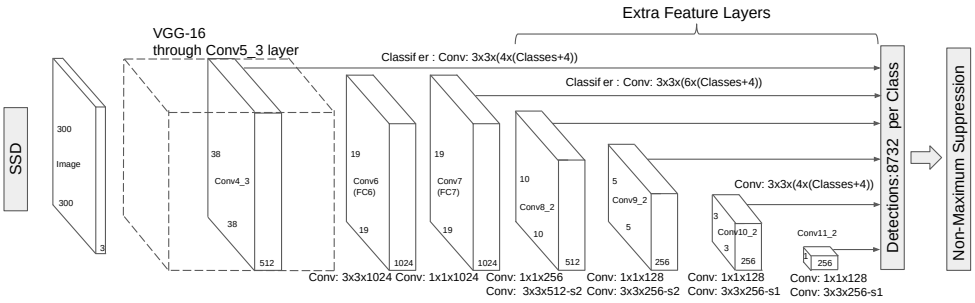


Figure 3.24. SSD model architecture. As it is possible to see, it does not have a network specifically designed for region proposal, but additional convolutional filters are used.

image are given as output. In Figure 3.23, the generation of anchor boxes through RPN is shown.

The last model is the Single Shot Multibox Detector (SSD) [60]. The SSD composes of two parts. The first is responsible of extracting the feature map, while the second applies convolution filters to detect the objects. For the feature map extraction, the VGG16 [86] deep neural network is used. From this network, the fully connected layer is removed and extra convolutional layer are added to perform the detection. In detail, after extracting the feature maps, SSD applies 3×3 convolution filters for each cell to make predictions. To detect objects independently, several convolutional layers are used (multi-scale feature maps). In Figure 3.24, the model architecture is shown.

Despite the SSD achieves better results in detection, it may be not easy to further improve it. Hence, by following the example given by Fast and Faster R-CNN, we improved the RPN of the latter, as explained in the next section.

3.3.2 Proposed Faster R-CNN Improvement

As seen, the Faster R-CNN is composed by two networks: a CNN, used for the classification task, and a RPN, used to generate the regions. By studying in depth the Faster R-CNN, we found that it is possible to drastically improve its performance. In detail, we have chosen as a CNN a 50-layers ResNet [41], due to its results in the ImageNet 2015 competition. Concerning the RPN, the proposed optimization

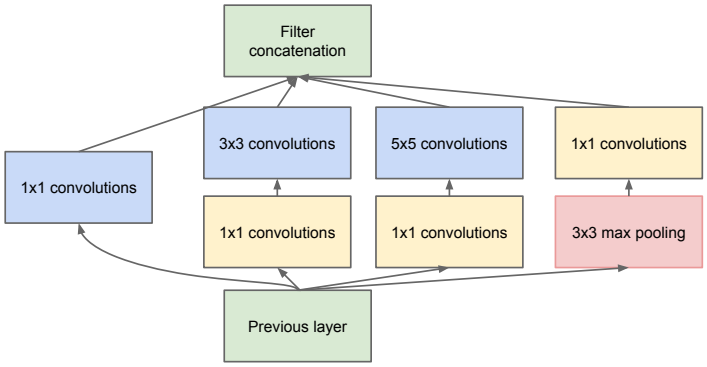


Figure 3.25. Inception module used in the RPN.

consists in creating and extended RPN, replacing the network of the classic RPN with an Inception deep network [91]. This allows achieving better results in detection, in reduction of the number of regions generated during the region extraction step, and in a speed up during the test stage. In Figure 3.25, the Inception module used in the extended RPN is shown. In Figure 3.26, instead, the proposed RPN is shown. The structure of this network consists of 9 stacked inception modules, which takes as input the features extracted with the CNN and its output is connected to the classification and regression layers. As it is possible to see, the resampling layers are missing. This is due to the fact that the original Inception model takes as input images at full size, while our Inception network works with the features provided by the CNN.

Concerning the number of parameters, since the classic RPN is implemented as a single convolutional layer, we have that the number of parameters is $ker^2 \times |filters|$, where ker^2 is the size of the kernel and $|filters|$ is the number of applied filters. For simplicity, in this computation the bias values and the parameters computing in the backward steps are removed. In the extended RPN, the number of parameters is obviously higher due to the several stacked inception modules. In Table 3.2, the number of filters and the kernel size of each layer is reported.

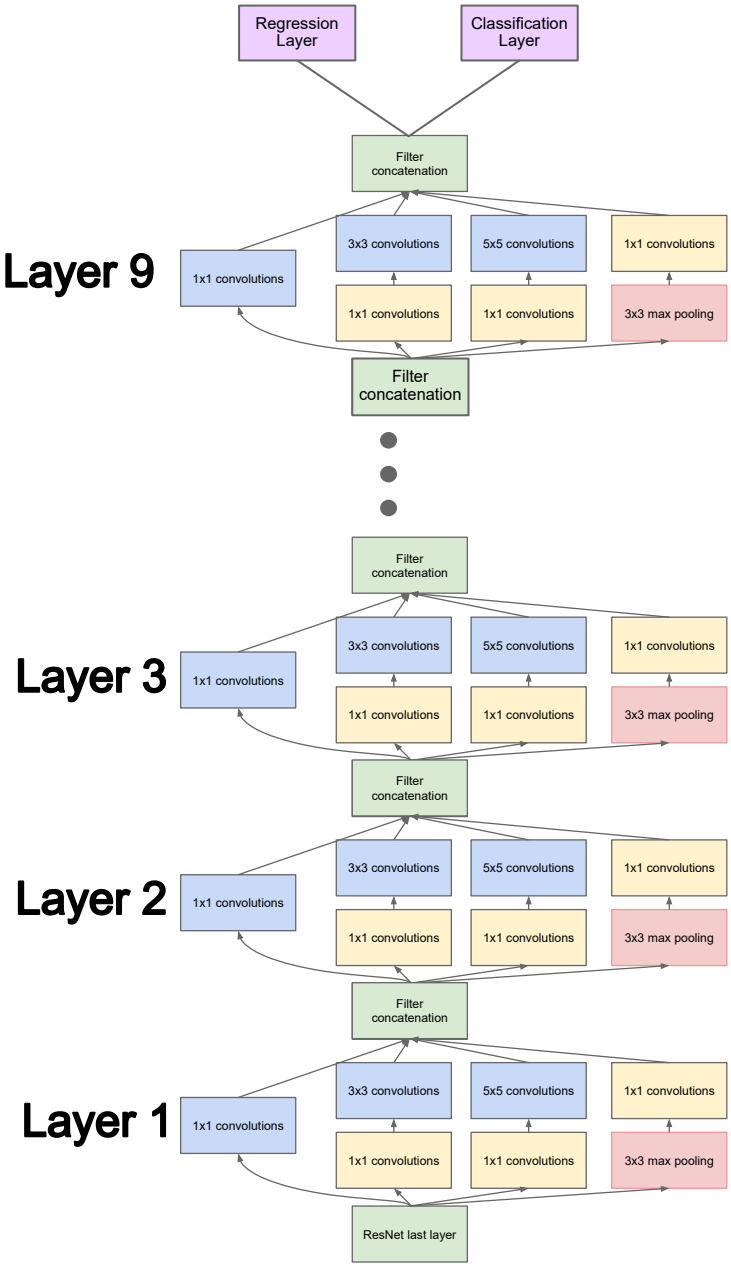


Figure 3.26. The stack of the inception modules that replaces the classic RPN.

Table 3.2. Parameters used in the proposed extended RPN newtork.

Layer	No. of Filters	Kernel Size	No. of Parameters
Layer 1			
Branch1x1	64	1	64
Branch3x3	96	3	864
Branch5x5	64	5	1600
Layer 2			
Branch1x1	64	1	64
Branch3x3	96	3	864
Branch5x5	64	5	1600
Layer 3			
Branch1x1	64	1	64
Branch3x3	96	3	864
Branch5x5	64	5	1600
Branch_pool			
Layer 4			
Branch1x1	192	1	192
Branch7x7_1	192	7	9408
Branch7x7_2	192	7	9408
Layer 5			
Branch1x1	192	1	192
Branch7x7_1	192	7	9408
Branch7x7_2	192	7	9408
Layer 6			
Branch1x1		1	192
Branch7x7_1	192	7	9408
Branch7x7_2	192	7	9408
Layer 7			
Branch1x1	192	1	192
Branch7x7_1	192	7	9408
Branch7x7_2	192	7	9408
Layer 8			
Branch1x1	320	1	320
Branch3x3_1	384	3	3456
Branch3x3_2	384	3	3456
Branch3x3_3	384	3	3456
Branch3x3_4	384	3	3456
Layer 9			
Branch1x1	320	1	320
Branch3x3_1	384	3	3456
Branch3x3_2	384	3	3456
Branch3x3_3	384	3	3456
Branch3x3_4	384	3	3456

Chapter 4

Experimental Results

In this chapter, the experiments performed with the proposed system are presented and discussed. The experiments are performed both on the proposed dataset (i.e., the UMCD) and public datasets: the NPU Drone-Map Dataset [17], and the Okutama Action [7]. The NPU Drone-Map Dataset is used for testing the mosaicking algorithm at very hight altitudes (i.e., up to 300 metres), while the Okutama Action has been used to test further the capability of the proposed deep learning model to detect pedestrians within the scene.

4.1 Mosaicking Experiments

For testing the mosaicking algorithm, two recent public datasets were used. The first is the UMCD dataset, presented in Section 2. The second is the NPU Drone-Map dataset, that contains a collection of aerial video sequences acquired at high-altitude. In both cases, the sequences are acquired by small-scale UAVs. Regarding the first dataset, we tested the algorithm on 40 challenging video sequences and measured the quality of the obtained mosaics by a simple metric based on the difference between image regions. Regarding the second dataset, we compared the proposed mosaicking algorithm with that presented in [17]. The latter is one of the few works in the literature that makes available source code, video sequences (i.e., the NPU Drone-Map dataset), and obtained mosaics to support a concrete comparison with other approaches. In particular, 4 challenging video sequences were selected from the second dataset and a correlation measure was adopted to quantify the similarity between mosaics pairs.

4.1.1 Low-Altitude and High-Altitude Mosaicking

In this sub-section, key considerations about the quality of the obtained mosaics are reported and discussed. Regarding the low-altitude, the adopted 40 video sequences had an average acquisition height of about 15 meters. In 4.1a an example is shown. In order to measure the quality of the mosaics derived by these video sequences the image difference process presented in [6] is adopted. The main idea is that each part of the mosaic must have the same spatial and colour resolution with respect to the original frames that have generated it. For this reason, the difference between each

(a) (b)

Figure 4.1. Mosaicking experimental results: (a) example of mosaic at low-altitude. The three miniatures are the frame extracted from the mosaic (up), one of the original frames used to build the mosaic (middle), the difference between the overlapped regions (bottom), (b) examples of mosaics at high-altitude by the proposed method (up), the method proposed in [17] (bottom).

portion of the mosaic and the linked original frames is computed. Subsequently, a simple histogram is calculated on each image difference to evaluate the degree of deviation. Anyway, this simple but effective process has shown that each part of the mosaic generated by the proposed method is quite similar to the linked frames. On average, the difference images shown a deviation of about 15%. This can be considered a real good result taking into account all the geometrical distortion and error propagations that occur during the complex mosaicking process. Moreover, it should be considered that the incremental real-time mosaicking process at low-altitude is a topic that needs to be further investigated. By the implemented UMCD dataset and the provided results, the aim is to provide a concrete first contribute for the comparison of these algorithms. In 4.1b, examples of high-altitude mosaics are shown. In particular, the mosaic on the top of the 4.1b is generated with the proposed approach, while the mosaic on the bottom is generated with the method proposed in [17]. Both mosaics were created by using the same video sequence contained in the NPU Drone-Map dataset (named: phantom3-centralPark). How it is possible to observe, some visual differences are present. This is due to the fact that the proposed method applies only basic transformations, such as translation, rotation, and scale change, while the method with which we compare performs the orthorectification of the frames. Despite this, the degree of correlation between

Table 4.1. Time needed for generating the mosaics. The unit is in minutes.

Sequence	Frames	KFs	Proposed	Bu et al. [17]	Pix4D	Photoscan
phantom3-npu	19,983	457	7.2	9.32	140.08	538.38
phantom3-centralPark	12,744	471	6.01	8.49	127.73	563.57
phantom3-village	16,969	406	10.4	11.31	132.07	360.70
phantom3-huangqi	14,776	393	8	10.36	102.83	462.32

the two types of mosaic is impressive. To verify the similarity between them the following metric was adopted:

$$corr = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2) \sum_m \sum_n (B_{mn} - \bar{B}^2)}} \tag{4.1}$$

where A, B are the two mosaics, and $\bar{A}$, $\bar{B}$ are the means of the mosaics pixels. On average, considering all the 4 video sequences reported in 4.1, we obtained a correlation value of about 80% among the mosaics. It should be considered that due to the different image processing, such as geometric transformation, orthorectification, stitching, and so on, it is not possible to obtain a perfect overlap between the mosaics. In particular, the different perspectives of the obtained mosaics are seen as significant differences by the metric. In any case, the degree of correlation can be considered a very high value.

Since the low altitude mosaics are the most challenging to be performed, we are going to focus on them. As shown in Figure 4.2, countryside and dirt environments are the less challenging in low altitude flights. This is due to the fact that within the acquired environment there are no elements having a relevant height. This means that the parallax error is strongly reduced or even absent. Moreover, in these environments it is possible to obtain good results also with the camera having an angle of 45 degree with respect to the ground. In Figure 4.2(c), an example of mosaic generated with this kind of acquisition is shown. Regarding the urban mosaics, some examples are shown in Figure 4.3. As it is possible to observe, some artifacts in the final mosaics are present. This is mainly due to two factors. The first factor, that concerns the mosaics shown in Figure 4.3(a) and Figure 4.3(b), is the presence of elements with relevant height, which influences the mosaic generation through the parallax effect. Concerning the mosaic in Figure 4.3(c), the artifacts are present due to the very low flight altitude.

4.1.2 Mosaicking Performance

In this sub-section, the performance of the proposed method is presented. All the experiments were performed on a laptop equipped with an Intel i7 6700HQ CPU, 16 GB DDR3 RAM and a nVidia GTX960 GPU. In 4.1, the time needed for generating the mosaics is reported. More specifically, we compared the proposed method with the algorithm reported in [17] and with two commercial software, Pix4D[1] and Photoscan[2], also reported in the same work. The proposed method stitches 1 frame

[1] https://pix4d.com/
[2] http://www.agisoft.com/

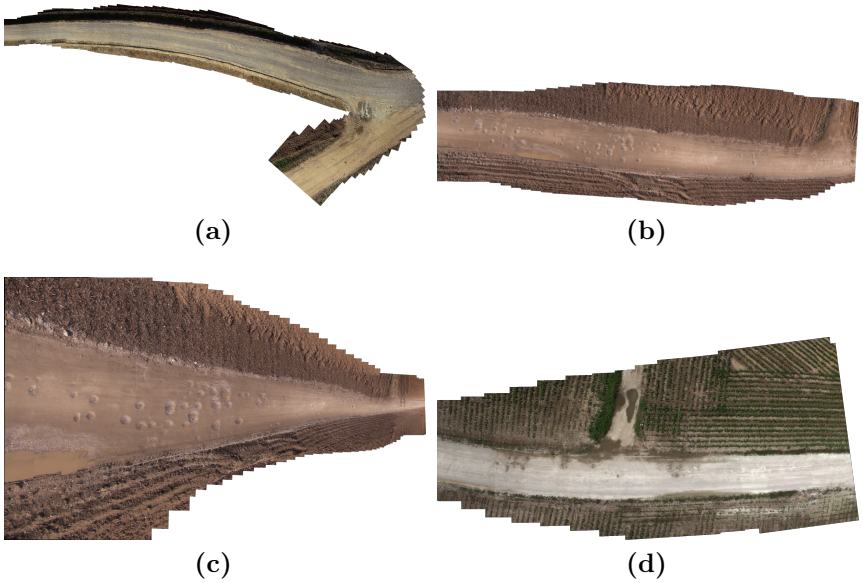


Figure 4.2. Examples mosaics of dirt and countryside environments acquired at very low altitudes.

per second, while the method proposed in [17] requires the stitching of 10 frames per second. Both Pix4D and Photoscan, instead, use only the keyframes to produce the final mosaic (i.e., similar to the proposed algorithm). Since all methods, with the exception of that proposed, use the GPU, a resize to the half of HD resolution (i.e., the original size of the frames) to be stitched is performed. In 4.1, the comparison is shown. As it is possible to observe, both the proposed and [17] algorithms take much less time than the commercial software. The proposed method show low processing times even with respect to the work proposed [17] and the generated mosaics by the two approaches result quite similar. Anyway, we are currently developing an approach to perform the orthorectification frame by frame.

4.2 Change Detection Experiments

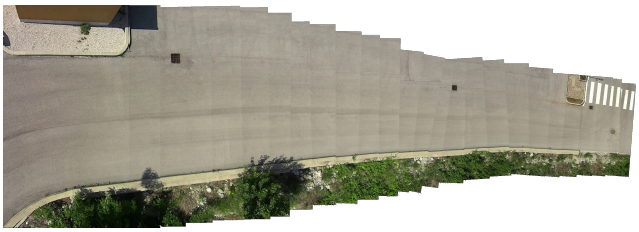
In this section, experiments performed on the change detection module are shown. Concerning the used algorithms parameters, in Table 4.2 our settings are reported.

If the parameters values are chosen too small, there could be a high number of false positive. Instead, big values could remove some important details, such as parts of the change that we want to detect. In the experiments, are set in order to always detect a change at the expenses of the precision.

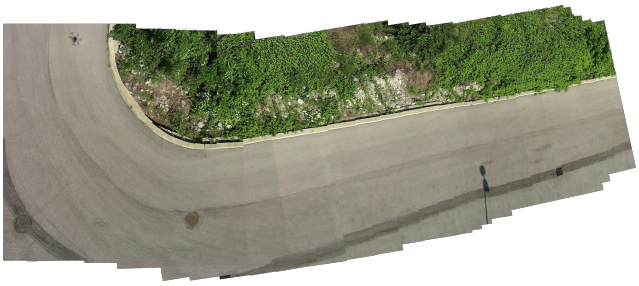
4.2.1 Detection Results

As metrics for measuring the detected changes, we used the *Precision (P)*, *Recall (R)*, *Accuracy (A)*, and *F1-Score (F1)*. In our tests, we assume that the Query and the Test images are already aligned with both GPS and features methods. The tests have been performed on 13871 images (i.e., the frames of the UMCD dataset videos), and in Table 4.3 the number of true/false positives and negatives are shown.

(a)



(b)



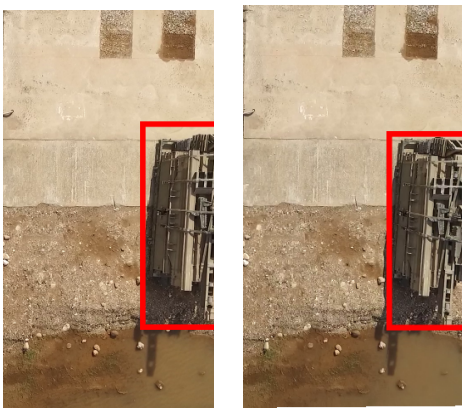
(c)

Figure 4.3. Examples mosaics of urban environment acquired at very low altitudes.

Table 4.2. Parameters settings used in the experiments.

Parameter	Value
W_a	10x10
W_l	15x15
T_{gray}	40
T_H	0.3
W_i	37x37
T_{iso}	0.25
T_{area}	0.000648
T_B	0.3

As it is possible to see, the system has a good detection rate and a low number of false detection, due to the several operator used in the pipeline. In Figure 4.4, an example of false positive is shown. The false detection has occurred since the object has been acquired with different perspectives, and this is due to the object dimensions. In detail, when an object is particularly small, the UAV acquires only one face of the object (i.e., the top). This is not true in our case, where the low-altitude makes the object not small enough, so more than one face are acquired.



(a) (b)

Figure 4.4. Example of false positive due to the different perspectives of the object within the red square in image (a) and (b).

In order to provide a general comparison result, in Table 4.5 per-pixel comparison with the state-of-the-art works are reported. In Table 4.4, the Precision, Recall,

Table 4.3. Number of true/false positives and negatives obtained during the experiments.

Parameter	Value
FP	884
FN	41
TP	3167
TN	9779

Accuracy and F1-Score values obtained are shown. As expected, the precision value

Table 4.4. Precision, Recall, Accuracy and F1-Score values obtained with the proposed system.

Metrics	Value
Precision	78.18%
Recall	98.72%
Accuracy	93.33%
F1	87.26%

Table 4.5. Per-pixel comparison with the state-of-the-art.

Change Detection System	Precision	Recall	Accuracy	F1
B. Wang et al. [100]	48%	76%	91.27%	-
Q. Wang et al. [103]	83.66%	-	-	-
Proposed	67.15%	95.88%	99.54%	78.98%

per-pixel is not high (i.e., 67.15%). This is due to the fact that,as mentioned before, the system aims to detect the whole change to provide a reliable input to more complex algorithms such as classification ones, so a higher accuracy value is preferable. In Figure 4.5, some examples of changes detected in different environments are shown.

4.2.2 Performance Evaluation

The change detection module has been implemented for supporting both single and multi thread execution. This twofold implementation allowed to simulate the behaviour of the pipeline on an embedded system, since most of the small-scale UAV have a CPU computational power amenable to a single thread desktop CPU. In detail, the multithreaded steps are the micro-differences removal, the image difference and the noise removal from the binary mask. In average, for a single change detection we obtained an execution time of 0.47 seconds in multithread implementation and 2.883 seconds in single thread implementation. With these performances, this system can be used for real-time applications in both embedded and desktop-based systems.

4.3 Object Detection Experiments

This section presents the experiments performed with the object detection module. For these tests, both the UMCD and the Okutama Action dataset are used. In detail, the UMCD has been used for the tests performed in Mode 1, while the Okutama Action has been used for the tests performed in mode 2. The training of the extended Faster R-CNN is performed in 4 steps:

1. In the first step, the RPN is trained as a normal CNN. For the error backpropagation, the Stochastic Gradient Descend (SGD) is used, while the learning rate and momentum values are, respectively, $0,0001$ and 0.9;

2. In the second step, the CNN is trained. In order to speed up the training, the transfer learning technique is used. The weights are initiated with the ones of a model pre-trained on the ImageNet dataset;

3. In the third step, the RPN weights are refined by using the CNN ones. This is achieved through the RPN and CNN shared layers;

4. In the final step, the CNN weights are further refined through the shared layers. This step has been performed for 2000 epochs.

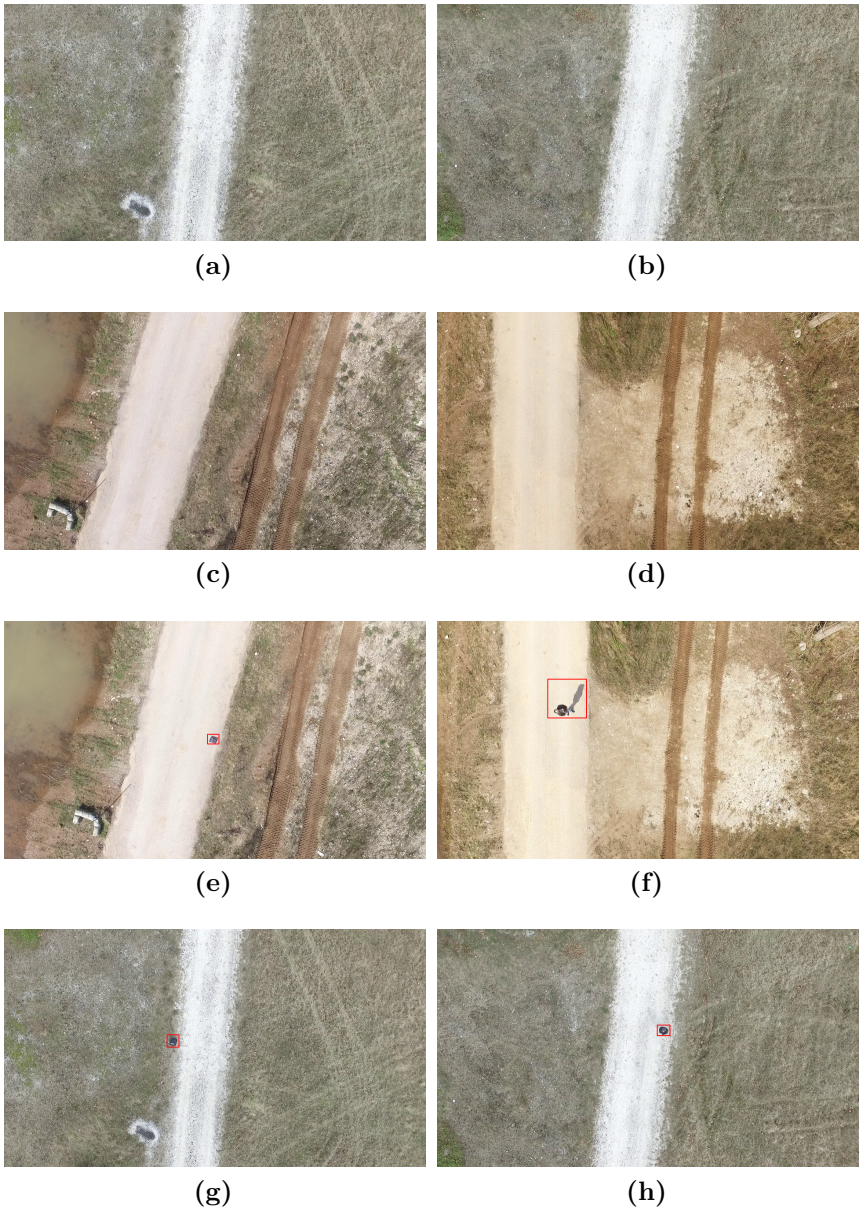


Figure 4.5. Comparison between a, b c, d) parts of mosaic without elements, and e), f), g), h) with elements detected.

Concerning the model evaluation, two metrics are used: the loss function and the Mean Average Precision (mAP). Regarding the loss function, it is computed on both CNN and RPN network. For the CNN, the loss refers to the classification level, while for the RPN it refers to the bounding box regression. For the CNN, the used loss function is the multi-task loss [79], defined as the sum of the classification loss L_{cls} and the regression loss L_{reg}. The first is given by the logarithmic loss between two classes, while the latter is given by the $L1$ smooth function [45]. After a normalization step, the two functions are sum together as follows:

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{4.2}$$

In the previous equation, p_i indicates the anchor box probability of containing an object, p_i* it ground truth and it can assume the value 0, i.e. if it contains no objects, or 1, i.e. if it contains an object. Instead, t_i is a vector containing the anchor box coordinates, while t_i* is its ground truth. Notice that, the loss function for the regression layer will activate only for positive anchor boxes.

For the RPN, to know if an anchor box contains an object a binary class to each of them is assigned. To do this, it is possible to proceed in two ways: the first is to evaluate the Intersection-over-Union (IoU) between the anchor box and the ground-truth-box, and choose the one with the highest value; the second is to take all the anchor boxes with IoU greater than 0.7 compared to each ground-truth-box. In our tests, we have chosen the second way.

Regarding the mAP, it is defined as the Average Precision (AP) function over the number of classes. The AP is defined as:

$$AP = \frac{TP}{(TP + FP)} \tag{4.3}$$

where TP and FP are the same defined in the change detection module. So, the mAP is the sum of the AP for each class divided by the total number of classes C. Formally:

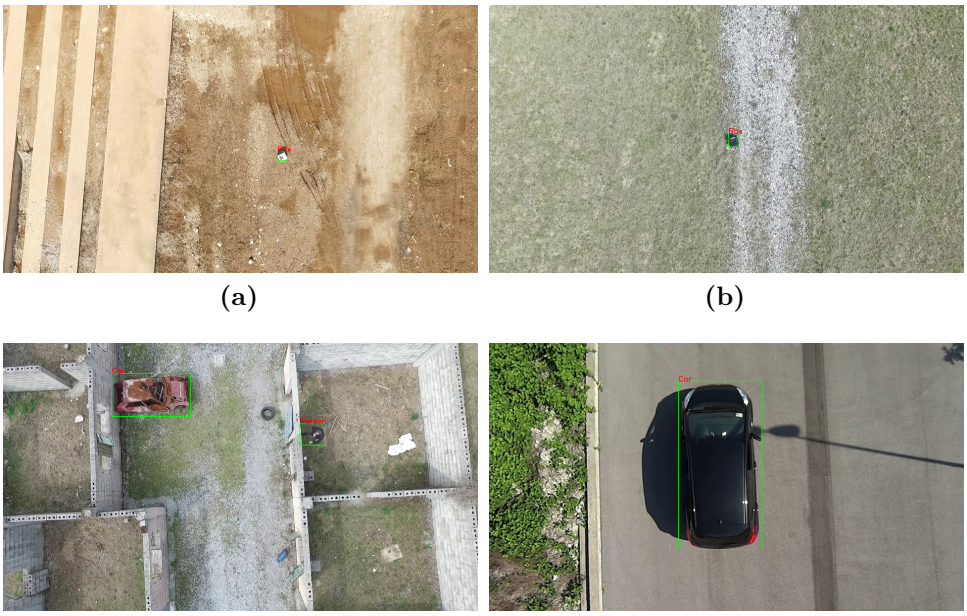
$$mAP = \frac{\sum_{i=1}^{C} AP(i)}{C} \tag{4.4}$$

For the UMCD dataset, the number of classes is 3, namely *person, car,* and *box*, while for the Okutama Action we have the single class *person*. The comparisons have been performed between the standard Faster R-CNN and our extended model. For each dataset, both the networks have been trained with the same number of epochs and the same input.

Regarding the Okutama Action dataset, it is composed by 54065 labelled frames. For the training, 45092 frames have been used (i.e., around 90% of the total number of frames), while for the test 8973 frames (i.e., around the 10% of the total number of frames) have been used. Concerning the UMCD, 4088 frames are extracted, and 3270 used for training the models (i.e., around the 80%), while 818 (i.e., around 20%) are used for testing. As it is possible to observe, for the Okutama Action a greater number of frames has been used for the training. This is due to the fact that the people within it are acquired from different angles, making more challenging the

Table 4.6. mAP values obtained with the two models on both testing datasets.

Model	Okutama Action mAP	UMCD mAP
Standard Faster R-CNN	69.3004%	95.5187%
Extended Faster R-CNN	76.8859%	99.8762%



(a) **(b)**

(c) **(d)**

Figure 4.6. Examples of detections performed in a) dirt, b) countryside and c,d) urban environments with the proposed extended Faster R-CNN.

pedestrian detection. In Table 4.6, the mAPs obtained with both the models on both datasets are reported. As it is possible to see, our model clearly outperforms the standard Faster R-CNN model, achieving better mAP in the performed experiments.

In Figure 4.6, some detection examples of different elements in different environments are reported.

Chapter 5

Conclusion

The following section concludes this thesis work by first resuming the contribution to the state-of-the-art, and then by showing the limitations of the proposed algorithms.

5.1 Contribution

Nowadays, small scale UAVs are widely used in several areas, such as border control, pipeline inspection, precision agriculture, and more. This thesis work presents a twofold UAV system that works in two modes. In the first mode, the UAV first performs a flight over an area of interest and sends video and GPS coordinate to the ground station, which builds a geo-referenced mosaic. Then, with a second flight over the same area, the UAV perform a change detection between the images acquired in real-time and the previous generated mosaic. In both mosaicking and change detection processes, the object found within the area are classified, allowing the system to know whether and where an object is appeared or disappeared within the environment. In the *Mode 2*, the UAV flights over an area of interest and classifies the objects present in it. While the *Mode 1* is useful for long term monitoring, *Mode 2* can be exploited for real-time critical tasks.

In this work, several contributions are introduced. Since the aim of the proposed system is to perform mosaicking, change detection, and object detection at very low altitudes (i.e., below 100 meters), the first contribution concerns the creation of aerial images acquired at these altitudes. The dataset is created by acquiring several types of terrain, i.e. dirt, countryside, and urban, at different altitudes by using different UAVs. This provides heterogeneity among the several videos, making the mosaicking process challenging. Moreover, the acquired areas are acquired twice. In the second acquisitions, elements such as persons, objects, and cars, are placed on the ground, allowing to perform change detection and object detection.

The second contribution regards the mosaicking module. In detail, different improvements have been made on the standard pipeline. The first improvement concerns the altitude estimation and the selection of the frames needed for the mosaic construction. These are performed automatically by considering parameters such as camera specifications, flight height, and flight speed of the UAV. The second improvement regards the feature extraction. This is performed by using A-KAZE feature extractor, which at low altitudes performs better than the classical

SIFT, SURF, and ORB algorithms. The last improvement concerns the frame transformation. Usually, in mosaicking literature the homography transformation is used. In the proposed mosaicking module, the similarity transformation is used. Since the latter has less degree of freedom with respect to the homography, it limits the transformations that can be applied to an image thus reducing errors in image stitching.

Concerning change detection module, an ad-hoc pipeline for detecting changes at low altitudes is provided. The pipeline exploits algorithms such as histogram equalization, RGB-LBP, and histogram comparison for robustly detecting changes. Several pre and post-processing operation are designed to reduce at minimum the number of false changes within the analysed images.

The last contribution involves the object detection module. In this work, an improved Faster R-CNN architecture is provided. In detail, the RPN of the standard Faster R-CNN has been substituted with an Inception network. This allows to drastically improve the performance on both bounding box generation and mAP.

Finally, the reproducibility of the results is discussed. The presented system has been implemented with open source frameworks, such as OpenCV [1]. By following step-by-step the algorithms proposed in this thesis, and together with the reported parameters, the obtained results can be easily reproduced.

5.2 Limitations

Although the proposed algorithms act well on most situations, they have some limitations.

Concerning the mosaicking module, there are mainly two limitations. The first is the problem concerning homogeneous texture of the area of interest. During the feature matching, if the acquired parts of the area have very similar appearance, the algorithm will mismatch the features. A classic example is the mosaic of the sea, in which the water has a very similar pattern in different frames. The second limitation is that the module does not perform orthorectification. This means that, if within the scene high elements such as trees, building, street lamps, are present, the mosaic will present some visual artifacts. This is due to the fact that, if an object is seen from different views, e.g., the facades of a building, the extracted features will be different for each facade, leading to a feature mismatch and to a bad frame stitching.

Regarding the change detection, the main limitation is the choice of the parameters. The tuning of the latter is very important, since a wrong parameter tuning will lead to false negatives of false positives changes. In detail, the several thresholds and kernel sizes must be chosen with respect to the light, terrain, environment, flight height, wind, and so on. This means that, for a real application, a small configuration step of the algorithms should be performed.

Considering the object detection module, the limitation is the quantity of data used for training the model. Usually, deep learning models are trained on thousand of images and object instances, and hundreds of classes. Unfortunately, the dataset introduced in this thesis work does not have these characteristics yet. This is due to the fact that presented dataset is a first step in providing reliable UAV imagery for

[1] https://opencv.org/

testing the set of algorithms presented in this work. Moreover, in our knowledge, from the time of its creation until now it is the only dataset providing images for mosaicking, change detection, and object detection from UAV at low altitudes.

Acknowledgements

First of all, I would like to express my special appreciation and thanks to my advisor Professor Luigi Cinque. I would like to thank you for encouraging my research since the Master Degree thesis, and for allowing me to grow up as a research scientist. Thanks for patience, motivation, and immense knowledge. Your guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

A special thanks to Professor Gian Luca Foresti. I would like to thank you for your insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

Another special thanks to Dr. Danilo Avola. Thank you for supporting me in the hardest times, for teaching me how to face problems, and for making out of me the researcher I am today.

I thank my friends and labmates Cristiano, Marco and Marco Raoul. Thank you for the support, for the stimulating discussions, and for all the fun we had in the last 3 years. Another special thanks goes to the lab coffee machine. Your coffee allowed me to do great research!

Last, but not the least, I would like to thank my family. You always supported me through all these years. There are no words describing how much I am grateful to you.

Bibliography

[1] P. F. Alcantarilla, J. Nuevo, and A. Bartoli. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–11, 2013.

[2] Nassim Ammour, Haikel Alhichri, Yakoub Bazi, Bilel Benjdira, Naif Alajlan, and Mansour Zuair. Deep learning approach for car detection in uav imagery. *Remote Sensing*, 9(4 - 312), 2017.

[3] N. Attari, F. Ofli, M. Awad, J. Lucas, and S. Chawla. Nazr-cnn: Fine-grained classification of uav imagery for damage assessment. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 50–59, 2017.

[4] D. Avola, G. L. Foresti, L. Cinque, C. Massaroni, G. Vitale, and L. Lombardi. A multipurpose autonomous robot for target recognition in unknown environments. In *INDIN*, pages 766–771, 2016.

[5] D. Avola, G. L. Foresti, N. Martinel, C. Micheloni, D. Pannone, and C. Piciarelli. Aerial video surveillance system for small-scale uav environment monitoring. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2017.

[6] Danilo Avola, Luigi Cinque, Gian Luca Foresti, Cristiano Massaroni, and Daniele Pannone. A keypoint-based method for background modeling and foreground detection using a ptz camera. *Pattern Recognition Letters*, 96:96 – 105, 2017.

[7] Mohammadamin Barekatain, Miquel Martí, Hsueh-Fu Shih, Samuel Murray, Kotaro Nakayama, Yutaka Matsuo, and Helmut Prendinger. Okutama-action: An aerial view video dataset for concurrent human action detection. *arXiv*, abs/1706.03038, 2017.

[8] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[9] M. B. Bejiga, A. Zeggada, and F. Melgani. Convolutional neural networks for near real-time object detection from uav imagery in avalanche search and rescue operations. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 693–696, 2016.

[10] Mesay Belete Bejiga, Abdallah Zeggada, Abdelhamid Nouffidj, and Farid Melgani. A convolutional neural network approach for assisting avalanche search and rescue operations with uav imagery. *Remote Sensing*, 9(2):1–22, 2017.

[11] A. Bhattacharyya. On a measure of divergence between two multinomial populations. *Sankhyā: The Indian Journal of Statistics (1933-1960)*, 7(4):401–406, 1946.

[12] S. Bianco, G. Ciocca, and R. Schettini. Combination of video change detection algorithms by genetic programming. *IEEE Transactions on Evolutionary Computation*, 21(6):914–928, 2017.

[13] D. Bobkov, S. Chen, R. Jian, M. Z. Iqbal, and E. Steinbach. Noise-resistant deep learning for object classification in three-dimensional point clouds using a point pair descriptor. *IEEE Robotics and Automation Letters*, 3(2):865–872, 2018.

[14] Marcel Bosc, Fabrice Heitz, Jean-Paul Armspach, Izzie Namer, Daniel Gounot, and Lucien Rumbach. Automatic change detection in multimodal serial mri: application to multiple sclerosis lesion evolution. *NeuroImage*, 20(2):643 – 656, 2003.

[15] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

[16] L. Bruzzone and D. F. Prieto. An adaptive semiparametric and context-based approach to unsupervised change detection in multitemporal remote-sensing images. *IEEE Transactions on Image Processing*, 11(4):452–466, 2002.

[17] S. Bu, Y. Zhao, G. Wan, and Z. Liu. Map2dfusion: Real-time incremental uav image mosaicing based on monocular slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4564–4571, 2016.

[18] Gershon Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin institute*, 310(1):1–26, 1980.

[19] Peter J. Burt and Edward H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transaction on Graphics*, 2(4):217–236, 1983.

[20] C. Cai, P. Wang, and Y. h. Liang. Fast image stitching based on improved surf. In *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 411–416, 2016.

[21] D. Capel and A. Zisserman. Automated mosaicing with super-resolution zoom. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 885–891, 1998.

[22] M. Che, P. Du, and P. Gamba. 2- and 3-d urban change detection with quad-polsar data. *IEEE Geoscience and Remote Sensing Letters*, 15(1):68–72, 2018.

[23] John B. Collins and Curtis E. Woodcock. An assessment of several linear change detection techniques for mapping forest mortality using multitemporal landsat tm data. *Remote Sensing of Environment*, 56(1):66 – 77, 1996.

[24] R. T. Collins, A. J. Lipton, and T. Kanade. Introduction to the special section on video surveillance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):745–746, 2000.

[25] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[26] Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. Handwritten digit recognition: applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989.

[27] Seohyung Lee Daeho Lee. Seamless image stitching by homography refinement and structure deformation using optimal seam pair detection. *Journal of Electronic Imaging*, 26:1 – 10, 2017.

[28] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[29] Patrick Doherty and Piotr Rudol. A uav search and rescue scenario with human body detection and geolocalization. In *AI 2007: Advances in Artificial Intelligence: 20th Australian Joint Conference on Artificial Intelligence*, pages 1–13, 2007.

[30] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski. Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85, 2017.

[31] D. R. Edgington, K. A. Salamy, M. Risi, R. E. Sherlock, D. Walther, and Christof Koch. Automated event detection in underwater video. In *Oceans 2003. Celebrating the Past ... Teaming Toward the Future*, volume 5, pages P2749–P2753 Vol.5, 2003.

[32] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of ACM*, 24(6):381–395, 1981.

[33] Changhong Fu, Ran Duan, Dogan Kircali, and Erdal Kayacan. Onboard robust visual tracking for uavs using a reliable global-local object model. *Sensors*, 16(9):1–22, 2016.

[34] Anna Gaszczak, Toby P. Breckon, and Jiwan Han. Real-time people and vehicle detection from uav imagery. In *SPIE: Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*, volume 7878, pages 1–13, 2011.

[35] R. Girshick. Fast R-CNN. *arXiv CoRR*, abs/1504.08083, 2015.

[36] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 580–587. IEEE Computer Society, 2014.

[37] M. Gong, X. Niu, P. Zhang, and Z. Li. Generative adversarial networks for change detection in multispectral imagery. *IEEE Geoscience and Remote Sensing Letters*, 14(12):2310–2314, 2017.

[38] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[39] N. Gupta, G. V. Pillai, and S. Ari. Change detection in optical satellite images based on local binary similarity pattern technique. *IEEE Geoscience and Remote Sensing Letters*, 15(3):389–393, 2018.

[40] Botao He and Shaohua Yu. Parallax-robust surveillance video stitching. *Sensors*, 16(1):1–12, 2015.

[41] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv*, abs/1502.01852, 2015.

[43] A Huertas and R Nevatia. Detecting changes in aerial views of man-made structures. *Image and Vision Computing*, 18(8):583 – 596, 2000.

[44] M. Irani and P. Anandan. About direct methods. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice*, pages 267–277, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[45] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv CoRR*, abs/1702.05659, 2017.

[46] S. Jeeva and M. Sivabalakrishnan. Survey on background modeling and foreground detection for real time video surveillance. *Procedia Computer Science*, 50:566 – 571, 2015.

[47] Yan Jin, Yan Liao, A. A. Minai, and M. M. Polycarpou. Balancing search and target response in cooperative unmanned aerial vehicle (uav) teams. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 36(3):571–587, 2006.

[48] D. J. Jobson, Z. Rahman, and G. A. Woodell. A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image Processing*, 6(7):965–976, 1997.

[49] D. J. Jobson, Z. Rahman, and G. A. Woodell. Properties and performance of a center/surround retinex. *IEEE Transactions on Image Processing*, 6(3):451–462, 1997.

[50] John Johnson. Analysis of image forming systems. *Image Intensifier Symposium, AD 220160 (Warfare Electrical Engineering Department, U.S. Army Research and Development Laboratories, Ft. Belvoir, Va., 1958)*, pages 244–273, 1958.

[51] J. E. Korteling and W. van der Borg. Partial camera automation in an unmanned air vehicle. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 27(2):256–262, 1997.

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, 2012.

[53] Eric N. Landis, Edwin N. Nagy, Denis T. Keane, and George Nagy. Technique to measure 3d work-of-fracture of concrete in compression. *Journal of Engineering Mechanics*, 125(6):599–605, 1999.

[54] K. Lebart, E. Trucco, and D. M. Lane. Real-time automatic sea-floor change detection from video. In *OCEANS 2000 MTS/IEEE Conference and Exhibition.*, volume 2, pages 1337–1343 vol.2, 2000.

[55] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[56] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[57] Ming Li, Ruizhi Chen, Weilong Zhang, Deren Li, Xuan Liao, Lei Wang, Yuanjin Pan, and Peng Zhang. A stereo dual-channel dynamic programming algorithm for uav image stitching. *Sensors*, 17(9 - 2060), 2017.

[58] Rui Li, Minjian Pang, Cong Zhao, Guyue Zhou, and Lu Fang. Monocular long-term target following on uavs. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 29–37, 2016.

[59] X. Li, Q. Yang, Z. Chen, X. Luo, and W. Yan. Visible defects detection based on uav-based inspection in large-scale photovoltaic systems. *IET Renewable Power Generation*, 11(10):1234–1244, 2017.

[60] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[61] Y. Liu, Q. Wang, Y. Zhuang, and H. Hu. A novel trail detection and scene understanding framework for a quadrotor uav with monocular vision. *IEEE Sensors Journal*, 17(20):6778–6787, 2017.

[62] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[63] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[64] Niki Martinel, Christian Micheloni, and Gian Luca Foresti. The Evolution of Neural Learning Systems: A Novel Architecture Combining the Strengths of NTs, CNNs, and ELMs. *IEEE Systems, Man, and Cybernetics Magazine*, 1(3):17–26, jul 2015.

[65] Philip F McLauchlan and Allan Jaenicke. Image mosaicing using sequential bundle adjustment. *Image and Vision Computing*, 20(9):751 – 759, 2002.

[66] Xiangyun Meng, Wei Wang, and Ben Leong. Skystitch: A cooperative multi-uav-based real-time video surveillance system with stitching. In *ACM International Conference on Multimedia*, pages 261–270, 2015.

[67] S. Minaeian, J. Liu, and Y. J. Son. Vision-based target detection and localization via a team of cooperative uav and ugvs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(7):1005–1016, 2016.

[68] R. Mitchell and I. R. Chen. Adaptive intrusion detection of malicious unmanned air vehicles using behavior rule specifications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(5):593–604, 2014.

[69] George Nagy, Tong Zhang, W.R. Franklin, Eric Landis, Edwin Nagy, and Denis T. Keane. Volume and surface area distributions of cracks in concrete. In Carlo Arcelli, Luigi P. Cordella, and Gabriella Sanniti di Baja, editors, *Visual Form 2001*, pages 759–768, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[70] Phuc Ngo, N. Passat, Y. Kenmochi, and H. Talbot. Topology-preserving rigid transformation of 2d digital images. *IEEE TIP*, 23(2):885–897, 2014.

[71] Chengcheng Ning, Huajun Zhou, Yan Song, and Jinhui Tang. Inception single shot multibox detector for object detection. In *IEEE International Conference on Multimedia Expo Workshops*, pages 549–554, 2017.

[72] S. Oh and et al. A large-scale benchmark dataset for event recognition in surveillance video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3153–3160, 2011.

[73] OpenCV Stitching Pipeline. `https://docs.opencv.org/2.4/modules/stitching/doc/introduction.html`.

[74] P. Planinšič and D. Gleich. Temporal change detection in sar images using log cumulants and stacked autoencoder. *IEEE Geoscience and Remote Sensing Letters*, 15(2):297–301, 2018.

[75] A. Price, J. Pyke, D. Ashiri, and T. Cornall. Real time object detection for an unmanned aerial vehicle using an fpga based vision system. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2854–2859, 2006.

[76] Juntong Qi, Dalei Song, Hong Shang, Nianfa Wang, Chunsheng Hua, Chong Wu, Xin Qi, and Jianda Han. Search and rescue rotary-wing uav and its application to the lushan ms 7.0 earthquake. *Journal of Field Robotics*, 33(3):290–321, 2016.

[77] Martin Radolko, Fahimeh Farhadifard, and Uwe von Lukas. Change detection in crowded underwater scenes - via an extended gaussian switch model combined with a flux tensor pre-segmentation. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 405–415, 2017.

[78] Martin Radolko and Enrico Gutzeit. Video segmentation via a gaussian switch background model and higher order markov random fields. In *Proceedings of the 10th International Conference on Computer Vision Theory and Applications*, pages 537–544, 2015.

[79] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[80] David Rey, Gérard Subsol, Hervé Delingette, and Nicholas Ayache. Automatic detection and segmentation of evolving processes in 3d medical images: Application to multiple sclerosis. *Medical Image Analysis*, 6(2):163–179, 2002.

[81] Deepak Kumar Rout, Badri Narayan Subudhi, T. Veerakumar, and Santanu Chaudhury. Spatio-contextual gaussian mixture model for local change detection in underwater video. *Expert Systems with Applications*, 97:117 – 136, 2018.

[82] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011.

[83] H. S. Sawhney and R. Kumar. True multi-image alignment and its application to mosaicing and lens distortion correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 450–456, 1997.

[84] A. Shimada, H. Nagahara, and R. i. Taniguchi. Change detection on light field for active video surveillance. In *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2015.

[85] Tianmin Shu, Dan Xie, Brandon Rothrock, Sinisa Todorovic, and Song-Chun Zhu. Joint inference of groups, events and human roles in aerial videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4576–4584, 2015.

[86] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, pages 1–14, 2014.

[87] Gurkirt Singh, Suman Saha, Michael Sapienza, Philip Torr, and Fabio Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3637–3646, 2017.

[88] Byung Cheol Song, Myung Jun Kim, and Jong Beom Ra. A fast multiresolution feature matching algorithm for exhaustive search in large image databases. *IEEE TCSVT*, 11(5):673–678, 2001.

[89] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[90] Jingxuan Sun, Boyang Li, Yifan Jiang, and Chih-yung Wen. A camera-based target detection and positioning uav system for search and rescue (sar) purposes. *Sensors*, 16(11):1–24, 2016.

[91] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[92] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv*, abs/1409.4842, 2014.

[93] R. Szeliski and Sing Bing Kang. Direct methods for visual scene reconstruction. In *Proceedings of Representation of Visual Scenes, IEEE Workshop*, pages 26–33, 1995.

[94] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 251–258, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[95] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):16, 2017.

[96] Marco Tektonidis and David Monnin. Color consistency and local contrast enhancement for a mobile image-based change detection system. *Journal of Imaging*, 3(3 - 35), 2017.

[97] C. Teulière, L. Eck, and E. Marchand. Chasing a moving target from a flying uav. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4929–4934, 2011.

[98] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[99] S. Waharte and N. Trigoni. Supporting search and rescue operations with uavs. In *International Conference on Emerging Security Technologies (EST)*, pages 142–147, 2010.

[100] B. Wang, S. Choi, Y. Byun, S. Lee, and J. Choi. Object-based change detection of very high resolution satellite imagery using the cross-sharpening of multitemporal data. *IEEE Geoscience and Remote Sensing Letters*, 12(5):1151–1155, 2015.

[101] G. Wang, Z. Zhai, B. Xu, and Y. Cheng. A parallel method for aerial image stitching using orb feature points. In *IEEE/ACIS 16th International Conference on Computer and Information Science*, pages 769–773, 2017.

[102] H. Wang, J. Li, L. Wang, H. Guan, and Z. Geng. Automated mosaicking of uav images based on sfm method. In *IEEE Geoscience and Remote Sensing Symposium*, pages 2633–2636, 2014.

[103] Qing Wang, Xiaodong Zhang, Yao Wang, Guanzhou Chen, and Fan Dan. *The Design and Development of Object-Oriented UAV Image Change Detection System*, pages 33–42. Springer Berlin Heidelberg, 2013.

[104] Yi Wang, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, Yannick Benezeth, and Prakash Ishwar. Cdnet 2014: An expanded change detection benchmark dataset. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 393–400, 2014.

[105] Daniel Wischounig-Strucl and Bernhard Rinner. Resource aware and incremental mosaics of wide areas from small-scale uavs. *Machine Vision and Applications*, 26(7):885–904, 2015.

[106] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[107] T. Wu, J. Luo, J. Fang, J. Ma, and X. Song. Unsupervised object-based change detection via a weibull mixture model-based binarization for high-resolution remote sensing images. *IEEE Geoscience and Remote Sensing Letters*, 15(1):63–67, 2018.

[108] J. Yuan, S. S. Gleason, and A. M. Cheriyadat. Systematic benchmarking of aerial image segmentation. *IEEE Geoscience and Remote Sensing Letters*, 10(14):1527–1531, 2013.

[109] A. Zeggada, F. Melgani, and Y. Bazi. A deep learning approach to uav image multilabeling. *IEEE Geoscience and Remote Sensing Letters*, 14(5):694–698, 2017.

[110] Jing Zhang, Guangxue Chen, and Zhaoyang Jia. An image stitching algorithm based on histogram matching and sift algorithm. *International Journal of Pattern Recognition and Artificial Intelligence*, 31(04):1–14, 2017.

[111] Mingfeng Zhang and Hugh H. T. Liu. Cooperative tracking a moving target using multiple fixed-wing uavs. *Journal of Intelligent & Robotic Systems*, 81(3):505–529, 2016.

[112] Y. Zhang, D. Peng, and X. Huang. Object-based change detection for vhr images based on multiscale uncertainty analysis. *IEEE Geoscience and Remote Sensing Letters*, 15(1):13–17, 2018.

[113] Y. H. Zhang, X. Jin, and Z. J. Wang. A new modified panoramic uav image stitching model based on the ga-sift and adaptive threshold method. *Memetic Computing*, 9(3):231–244, 2017.

[114] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[115] X. Zhao, W. Li, Y. Zhang, T. A. Gulliver, S. Chang, and Z. Feng. A faster rcnn-based pedestrian detection system. In *IEEE 84th Vehicular Technology Conference*, pages 1–5, 2016.

[116] G. Zhou. Near real-time orthorectification and mosaic of small uav video flow for time-critical event response. *IEEE TGRS*, 47(3):739–747, 2009.

[117] H. Zhou, D. Zhou, K. Peng, R. Guo, and Y. Liu. Seamless stitching of large area uav images using modified camera matrix. In *IEEE International Conference on Real-time Computing and Robotics*, pages 561–566, 2016.

[118] Y. T. Zhou and R. Chellappa. Computation of optical flow using a neural network. In *IEEE 1988 International Conference on Neural Networks*, volume 2, pages 71–78, 1988.

[119] I. Zoghlami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2d mosaic from a set of images. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 420–425, 1997.

[120] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion. *Acta Numerica*, 26:305–364, 2017.

My Publications

Avola, D., Cinque, L., Foresti, G. L., and **Pannone, D. Visual Cryptography for Detecting Hidden Targets by Small-Scale Robots**. Lecture Notes in Computer Science, In Press

Piciarelli, C., Avola, D., **Pannone, D.**, and Foresti, G. L. **A video-based system for internal pipeline inspection. IEEE Transactions on Industrial Informatics**. In Press

Avola, D., Cinque, L., Foresti, G. L., Martinel, N., **Pannone, D.** and Piciarelli, C. **A UAV Video Dataset for Mosaicking and Change Detection From Low-Altitude Flights**. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018, pp. 1-11

Avola, D., Cinque, L., Foresti, G. L., Marini, M. R. and **Pannone, D. VRheab: a fully immersive motor rehabilitation system based on recurrent neural network**. Multimedia Tools and Applications, 2018

Avola, D., Cinque, L., Foresti, G. L., Martinel, N., **Pannone, D.** and Piciarelli, C. **Low-Level Feature Detectors and Descriptors for Smart Image and Video Analysis: A Comparative Study**. Bridging the Semantic Gap in Image and Video Analysis Springer International Publishing, 2018, pp. 7-29

Avola, D., Cinque, L., Foresti, G. L., Marini, M. R. and **Pannone, D. A Rover-based System for Searching Encrypted Targets in Unknown Environments**. Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM, SciTePress, 2018, pp. 254-261

Avola, D., Foresti, G. L., Martinel, N., Micheloni, C., **Pannone, D.** and Piciarelli, C. **Real-Time Incremental and Geo-Referenced Mosaicking by Small-Scale UAVs**. Image Analysis and Processing - ICIAP 2017 Springer International Publishing, 2017, pp. 694-705

Avola, D., Foresti, G. L., Martinel, N., Micheloni, C., **Pannone, D.** and Piciarelli, C. **Aerial video surveillance system for small-scale UAV environment monitoring**. 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2017, pp. 1-6

Avola, D., Cinque, L., Foresti, G. L., Massaroni, C. and **Pannone, D. A keypoint-based method for background modeling and foreground detection using a PTZ camera**. Pattern Recognition Letters, 2017, Vol. 96, pp. 96 - 105

Avola, D., Cinque, L., Foresti, G. L., Mercuri, C. and **Pannone, D. A Practical Framework for the Development of Augmented Reality Applications by using ArUco Markers**. Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM, SciTePress, 2016, pp. 645-654