Politecnico di Torino

INFRA-SCAN

Interdisciplinary Projects
Master Degree in ICT4SS
Politecnico di Torino

Professor:
Fabio Dovis

Reference professors:
Marco Piras
Vincenzo di Pietra

Group F
s275144 - Lina Maria Medina Grajales
s275142 - Juan Sebastian Rojas Velandia
s276990 - Felipe Suarez Flechas
s273856 - Juan Gabriel Pieschacon Vargas

June, 2021

Contents

List of Figures

List of Tables

1 Introduction

Bridges are a crucial component in the countries infrastructure. They enable the transport of raw materials and goods to markets, allowing people access to essential services. Hence, the economic activity is closely linked to the bridges availability. In Italy, there are more than 1034 kilometers of bridges and viaducts [1]. In particular, there are around 2000 highway bridges. Many of the bridges date back to the construction boom in the 1930s and the post-war recovery programs of the 1960s [2] and are now reaching the end of their lifespan. Moreover, most bridges also carry significantly more weight than originally expected [3]. Based on this, bridges maintenance, inspection, and monitoring should be an aspect of high priority for a nation. On the one side, it is important to ensure the bridge's readiness to maintain the economic equilibrium of its surrounding zones. However, on the other hand, it is vital to prevent fatal accidents such as the Morandi Bridge collapse in Genoa that caused 43 deaths [4].

Currently, the inspection of bridges is a time-consuming procedure that involves a team of engineers that carefully inspect the structure looking for any flaws, defects, or potentially problematic areas that may require maintenance. In addition, some special machinery like snooper trucks is used frequently to reach the zones of difficult access of the structure increasing the costs of the inspection. Moreover, the inspectors should operate at large distances from the ground, risking their lives. Bridge inspections are specified by different government guidelines, which can vary significantly by country. In Italy, they are given by the Minister of infrastructure and transport, which stipulates the process of risk classification, safety evaluation, and monitoring of bridges [5].

In this work, we present an alternative to the current bridge inspections. We propose the use of Unmanned Aerial Vehicles (UAV) and Machine Learning for the collection of images of the bridge and detection of damages in the structure. With this tool, we plan to improve the bridge inspection processes by reducing the use of expensive machinery, avoiding people's exposure to dangerous operations, and reducing the inspection time. In particular, the proposed system consists of a UAV that is guided around the bridge structure while it is recording different videos of the structure. The collected data is processed and input in the You Only Look Once (YOLO) [6] algorithm for damage detection.

The remaining of this report is organized as follows. Section 2 reviews some of the similar works found in the literature. Section 3 introduces the user requirements. Section 4 presents the workflow followed for the elaboration of the project. The design process and its validation are presented in Section 5. In Section 6 the data used for the algorithm training and testing is described. The test methodology and planning are discussed in Section 7. Finally, in the Sections 8 and 9, the results and conclusions are presented.

2 State of the art

The inspection of road infrastructures is a task that has been carried out manually in last years. For this procedure, it is usually necessary to hire a bridge inspection truck (see Figure 1a) and specialized personnel to detect and classify possible failures or infrastructure damages. This approach is expensive due to the rental of the necessary elements, personnel, and sometimes it is necessary to close roads for the correct location of the truck. In addition to this, the safety conditions of the operator are not optimal since he/she is exposed to high and dangerous places (see Figure 1b) [7].

The use of unmanned aerial vehicles (UAV), in this procedure, offers different advantages such as reduction of costs, revision time, and the risk of jobs in hazardous places. Moreover, this

(a) Inspection machinery.

(b) Inspection risks.

Figure 1: Traditional inspection process.

type of technology allows the storage of the recorded images, which enables the evaluation of the infrastructure by a group of specialists, reducing the subjectivity of the diagnosis when it is carried out by a single operator. For these reasons, different studies focused on the development and implementation of new technologies in this field have been carried out.

Mainly, the studies have addressed the collection of images and videos, and their analysis with or without the help of a previous Building Information Model (BIM). From this data, 3D and 4D reconstructions have been generated, also implementing augmented reality (AR) technologies [8][9]. Some specific applications include the inspection of buildings after earthquakes, detection of fractures in the facade of buildings, reconstruction of roof contours and orthographic mapping [10] [11].

However, the use of UAVs is not easy and various challenges must be faced, such as planning the route to follow, whether it is done automatically or manually, as well as take-off and landing procedures. Simultaneous Localization and Mapping (SLAM) techniques are currently being developed, used by robots or autonomous vehicles, to explore and reconstruct maps of unknown environments [12]. In addition to this, the configuration and control of the sensors used for data collection is required, either the cameras used or the positioning sensor (GNSS, IMU, optical flow module, etc.) [13].

Focusing on this project, J. Seo et al. [14] proposes a methodology composed of 5 stages where the geographic information of the bridge is initially evaluated, the risks that may arise, an initial pre-flight route to follow, a verification of the feasibility of execution of the route, and finally the identification of the main infrastructure damages. However, this methodology is proposed for a specific type of bridge and it is necessary to redesign or modify the methodology according to the geometry and type of bridge.

The use of Machine Learning brings additional benefits to the bridge inspection with UAVs. The subjectivity of the damage detection and classification, and the need of staff can be reduced even more. Some works employ image segmentation using neural networks to classify each pixel [15]. This type of approach generates good results but has a high computational cost. In this way, object detection is proposed as a solution to reduce the computational effort. In particular, the YOLOv4 model is used [6]. The selection of the model was mainly based on the work described in [16]. Find more details about the algorithm selection in section 5.3.

3 User requirements

A set of first requisites are needed to establish a foundation for the project scope, vision, and schedule to target certain product quality and performance. Considering the motivations of the project and after conducting a meeting with some experts in the field, the following requirements were constructed to reach the goals of the project. Each of them is divided into three work packages to discriminate the specific needs in each part of the work, Geoscan system, flight, and damage detection algorithm.

- **Geoscan system:**
  - Define which sensors are going to be used among the available ones.
  - Define the most suitable placement of the sensors in the UAV.
  - Program and/or configure the sensors for their specific tasks.
  - Integrate the sensors to the UAV.
  - The system should be able to capture pictures at the zenith angle.
  - The system should store the pictures in a data storage device.
  - The design should optimize energy consumption and therefore flight time.
- **Flight:**
  - Develop an algorithm that indicates to the UAV the route to follow.
  - The system should capture images periodically during the automatic flight.
  - It is necessary to associate manually or automatically the bridges section with the picture.
- **Damage detection algorithm:**
  - Collect images from open sources on the web.
  - Collect images with the UAV system.
  - Create a dataset with the collected images.
  - Label the images.
  - Pre-process the data.
  - Develop an algorithm for the detection and classification of damages, each image should be classified between crack, spalling, delamination, and no damaged surface.
  - The pre-processing and processing stages should be done offline.
  - The algorithm should generate a report with the image classification.
  - Evaluate the performance of the algorithm.

4 Project workflow

The project was carried out in seven main milestones. They were distributed over time, as is illustrated in Figure 2. The seven work packages consist of sub-activities that are described in detail in the Gantt chart on Appendix A.

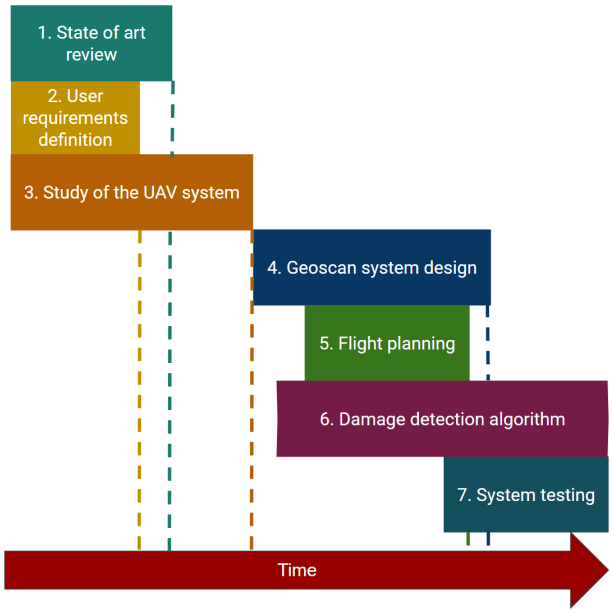


Figure 2: Workflow.

The first task was the state-of-the-art review. We explored essential background elements to understand the problem and context in which our project can be developed. Moreover, we analyzed the possible issues and solutions already implemented (see Section 2). The second phase was the user requirements definition. After a meeting with some experts in the field, the requirements of the project were established (see Section 3). The third stage was the study of the UAV system. In this part, the provided drone (Geoscan Pioneer) was assembled, its documentation was studied, and some initial tests were performed. Considering the effort needed (measured in time to perform the task and hours spent by each member of the group), all three aforementioned stages were simultaneously worked.

After completing the study of the UAV system, the next step was the Geoscan system design. In this stage, we examined and tested different possible sensors provided by the Geoscan Pioneer kit, considering not only their capabilities but also assessing their optimal placement in the drone, verifying weight, balance, and energy efficiency. In parallel to this phase, flight planning was executed. It consisted of performing different tests of both automatic and manual flight.

While doing the Geoscan system design and flight tests, we started the development of the damage detection algorithm. This sixth phase comprises algorithm selection, data collection, pre-processing and labeling, code development, and algorithm training. Finally, we did the testing stage in which the trained model was used on test data to extract some performance indicators.

5 Design

In this section, we describe the system design process. We present some tests performed to validate the applicability of the different options to our problem. The design system was divided into three main parts: the Geoscan system, the flight, and the damage detection algorithm.

5.1 Geoscan system

For the development of the project, the UAV Geoscan Pioneer manufactured by Geoscan LTD was provided. Figure 3 shows the drone.

Figure 3: UAV Geoscan Pioneer.

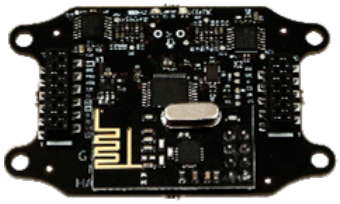
The Geoscan Pioneer is a multifunctional educational and methodical kit composed of the UAV and a set of modules that can be attached to it. The Characteristics of the drone are the following [17]:

- Flight time up to 17 min.
- Speed up to 65 km/h.
- Weight 230 g.
- Frequency 2.408 - 2.475 GHz.
- RX Sensitivity -92 dBm.
- Dimensions 290 x 290 x 120 mm.
- Brushless motors 1306 3100 KV.
- LiPo battery 2S 1300 mAh/ 9,62 Wh.
- Max altitude 500 m.
- Wind tolerance 5 m/s.
- Temperature range 0 to +40 °C.

The modules included in the Geoscan Pioneer development kit are presented in Figure 4. The first three modules (Figures 4a, 4b, and 4c) can be used for drone positioning. The GNSS module, as its name indicates, allows tracking the UAV's speed and position using global navigation systems. The optical flow module gives the Pioneer the ability to position itself without using a local or global navigation system. It is equipped with an optical distance sensor and can measure altitude above the floor or other objects from 0 to 1.5 m range. When Pioneer exceeds this value, a barometer is

used for altitude control. On the other hand, the onboard indoor navigation module is used along with a control module and 4 ultrasonic transducers for inside-building flights [18].

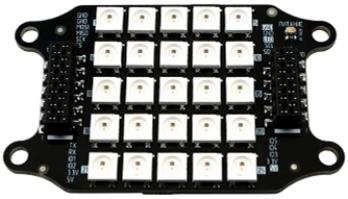
(a) GNSS.



(b) Onboard indoor navigation.

(c) Optical flow.

(d) Cargo module.



(e) LED module.

(f) OpenMV camera.

(g) RunCam.

Figure 4: Geoscan Pioneer modules.

Among the remaining modules, we have the cargo module (Figure 4d), which has a flexible gimbal with an electromagnet. It can be used for holding magnetic objects. Regarding the LED module (Figure 4e), it is a set of LEDs. Finally, among the modules, we have two cameras, the OpenMV camera (Figure 4f) and the RunCam (Figure 4g) [18].

Based on the characteristics of each sensor and the requirements of our project, we selected the most suitable ones. In particular, we discarded the onboard indoor module because it is designed only for indoor applications. Similarly, the optical flow module was not selected since it is useful for short flight distances. Its positioning error increases as the quadcopter is farther from the ground. For positioning, we chose the GNSS module. It can estimate the position of the drone in outdoor applications and large flight distances.

The LED and Cargo modules were not considered, being that they do not provide useful functionalities for the development of the project. Regarding the cameras, we decided to select both options since they can record at the zenith angle.

The reasoning for selecting and further evaluating the capabilities of these modules are described in Sections 5.1.1 and 5.1.2. We validate how adequate the sensors are performing some tests to verify their applicability in our project.

5.1.1 Cameras

Table 1 presents the main characteristics of the cameras selected in the previous section. It is observed that one of the main advantages of the OpenMV camera is that it is programmable. It allows us to develop scripts in the Micropython programming language through the OpenMV IDE (Integrated Development Environment). We can also deploy machine learning models by using TensorFlow Lite. This means that the OpenMV camera could eventually make damage detection onboard with its image processing capabilities. Additionally, this camera can communicate with the main board of the UAV through UART or I^2C. However, a significant drawback of the OpenMV camera is its resolution which could be not adequate for the needs of the project [19].

	OpenMV	**RunCam**
Programmable	✓	X
UART	✓	X
Resolution	640 x 480/ 320x240, 60 fps	HD or FHD with 30 or 60 fps
Storage	microSD card	microSD card
gimbal	—	Rotating

Table 1: Characteristics of the cameras.

The RunCam, on the other hand, is not programmable and is independent of the UAV. This means that it does not have communication with the main board of the drone. This camera can be managed in two ways, manually with some buttons or through a mobile application. In terms of resolution, this camera has a better resolution than the OpenMV. Another advantage of the RunCam over the OpenMV is that it has a rotating gimbal that allows recording videos pointing at different angles [20]. Regarding the storage, both cameras can save the photos and the videos on an external microSD card. Moreover, as mentioned previously, both cameras satisfy the requirement of recording at the zenith.

To validate the features of the cameras, a validation test was carried out. The test took place at Politecnico di Torino (Main campus - Corso Duca) in the lab of the Department of Structural, Geotechnical and Building Engineering (DISEG). In this laboratory, there are two bridge structures from Madrid that contain examples of crack and spalling.

The test consisted of acquiring photos of one of the structures that presented a notable spalling. We took photos with both cameras at 40 cm, 1 m, and 2 m of distance. The resulting pictures are shown in Figure 5. From the images taken with the OpenMV module, it is appreciated that the camera can capture the spalling only at a distance of 40 cm. At longer distances, the resolution of the camera is not sufficient to recognize the damage. In contrast, from the images taken with the RunCam, the higher resolution allows capturing the defects at longer distances compared to the OpenMV, up to about 1 m.

Based on the results, the OpenMV camera was discarded because of its low resolution. To use this camera in our project, the drone would have to fly at less than 40 cm from the bridge. This is unfeasible since the drone needs a safety margin to navigate near the structure. We proposed a safety margin of 50 cm. On the other hand, the RunCam could be suitable since it can record distinguishable defects from a distance of at most 1 m.

(a) OpenMV 40 cm.

(b) RunCam 40 cm.

(c) OpenMV 1 m.

(d) RunCam 1 m.

(e) OpenMV 2 m.

(f) RunCam 2 m.

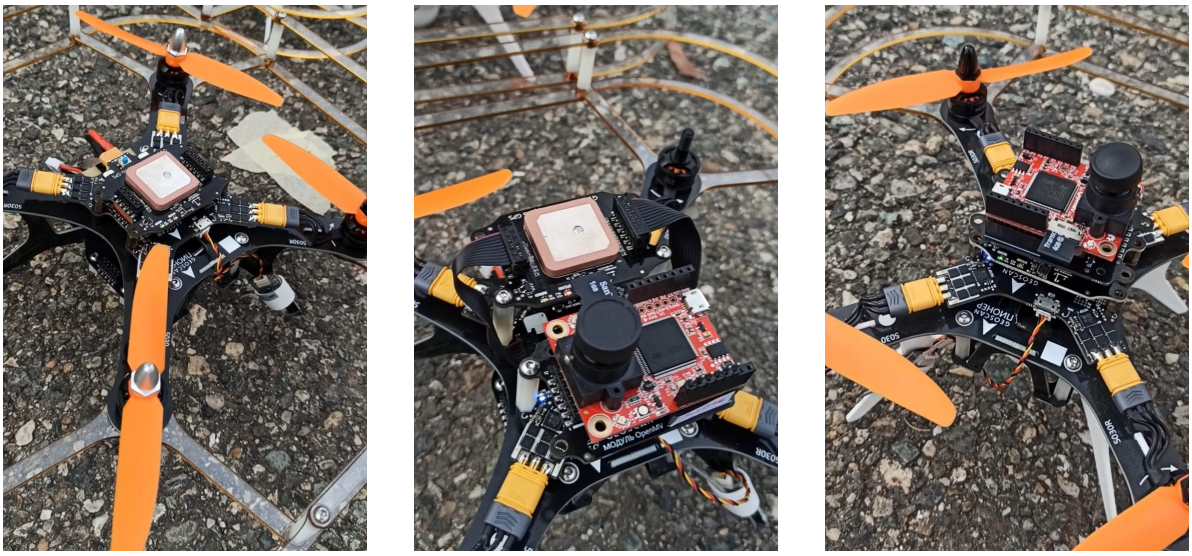
Figure 5: Test - cameras.

5.1.2 GNSS module

The GNSS module is composed of a U-blox NEO-6M series module, enabled to receive signals from the GPS and GLONASS constellations and a compass to get orientation data. The module is connected to the mainboard using a UART serial interface. After the first-time activation in a new location, the GNSS module executes the cold start to establish the actual position. This procedure will take around 1-3 minutes [17].

As mentioned in Section 3, the camera and GNSS receiver must be located over the UAV Geoscan Pioneer. In this position, the GNSS receiver will not have any obstructions and could have full sky visibility to detect orbiting satellites above the drone.

To evaluate the performance of the GNSS receiver, three different configurations were defined (see Figure 6). Figure 6a shows the first configuration that consists only of the GNSS module. Figure 6b depicts the second arrangement where the camera is located side by side with the receiver. The last sensors' organization is illustrated in Figure 6c, where the camera is over the GNSS module. These three configurations were tested using both cameras, one at a time, while the camera was turned on and recording.



(a) GNSS. (b) GNSS Camera-side. (c) GNSS Camera-up.

Figure 6: Test - GNSS module.

When the GNSS module is connected to the mainboard and the motors start rotating, the UAV Geoscan Pioneer starts recording different metrics related to the GNSS performance. Figure 7 displays some measurements of the GNSS receiver. It shows the comparison between the three configurations. In all the graphs, the green line represents the receiver alone configuration, the orange corresponds to the GNSS camera-side and the blue is the GNSS Camera-up setup.

Figures 7a, 7b, and 7c correspond to the position given by the GNSS module. The second and third configurations (blue and orange curves), show additional variations in the estimated position compared with the first configuration (the green curve), the measurement is less noisy (i.e., presents lower amplitude in its oscillations). This suggests that the camera placement affects the GNSS positioning precision.

(a) Altitude. (b) Latitude. (c) Longitude.

(d) PDOP. (e) Horizontal accuracy. (f) Vertical accuracy.

(g) GLONASS satellites. (h) GPS satellites.

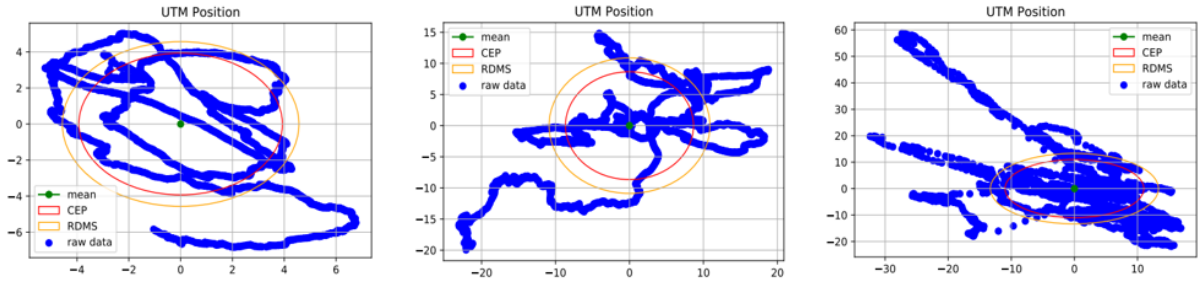
Figure 7: Metrics GNSS receiver.

The metrics related to the precision of the GNSS receiver are shown in Figures 7d, 7e, and 7f. It is observed that the first configuration performs better than the other two for the Position Dilution of Precision (PDOP) and horizontal and vertical accuracy. This is supported by the fact that this GNSS configuration displays for all three metrics values that are significantly lower than the values of the other configurations. Moreover, the green curve is always more stable than the others (presents less noise).

Regarding the number of satellites, we can see from Figures 7g and 7h that the number of available satellites, compared with the first configuration, is lower by 1 to 4 for the second and third configuration, with both the GPS and GLONASS constellations. This means that the cameras in both configurations obstruct to some extent the visibility of the GNSS. This is consistent with the results obtained in the previous graphs.

To compute the accuracy metrics, the Universal Transverse Mercator (UTM) coordinates were computed from the geographical coordinates given by the GNSS receiver. Figures 8a, 8b, and 8c contain all the computed UTM positions, also the Circular Error Probability (CEP), Distance Root Mean Square (RDMS) and the Spherical Error Probability (SEP). The CEP obtained for the three

configurations were 4 m, 8.6 m, and 11 m, respectively. These results also evidence the negative effects produced by the camera positioning with respect to the GNSS module.



(a) GNSS. (b) GNSS Camera-side. (c) GNSS Camera-up.

Figure 8: Accuracy measurements GNSS receiver.

5.2 Flight

According to the performance metrics for the GNSS module and the required distance for the cameras to capture a good image described in Section 5.1, the system has to deal with a trade-off between the GNSS precision and the image quality. This means that it is required a more precise GNSS receiver or a camera with a higher resolution to acquire images in which the structural damage can be identified.

Considering the CEP obtained for each configuration, between 4 m and 11 m, the UAV Geoscan Pioneer does not fulfill the trade-off due to the high GNSS error. For a distance of 4 m or more, the RunCam cannot obtain high-quality pictures to detect structural defects, and the GNSS receiver is not capable of estimating a more precise position to acquire images from a near position. Additionally, following the instructions to perform an automatic flight, the drone presents irregularities that always result in a crash. The UAV can be configured to improve the automatic flight. However, the available documentation to set up the parameters is unclear [17], and the Geoscan support desk has not provided additional information to set up the automatic flight. A manual flight was also tested, but the outcome was always the same, a crash. All the tests reported in this subsection were made on the flying field in the main campus of Politecnico di Torino (Corso Castelfidardo) shown in Figure 9.

Figure 9: Testing field.

Because of the difficult hand-operated and irregular automatic flight using the UAV Geoscan Pioneer and the above-mentioned reasons regarding the cameras and GNSS module trade-off, it is concluded that the drone is not suitable for this project. As a solution, the UAV Phantom 4 PRO, shown in Figure 10, is proposed to perform the flight manually. In this case, it is necessary for route guidance, an expert pilot to control the drone. Phantom 4 PRO uses a Real-Time Kinematic (RTK) module that reduces the vertical and horizontal error to 0.1 m. To collect images and perform the initial flight test, both cameras were mounted over the drone, as shown in Figures 10a and 10b. However, as described in Section 5.1.1, the OpenMV camera ends up being discarded. Therefore, the image collection was performed with the Phantom's integrated camera and the RunCam camera recording at the zenith angle.

(a) Front view. (b) Back view.

Figure 10: Phantom 4 Pro.

From the initial requirements introduced in Section 3 the work package **flight** must be changed, as an automatic flight is not more feasible with the Geoscan system and now is a hand-operated job with the Phantom 4 Pro system. The requirements are redefined as follows:

- **Flight:**
  - A hand-operated drone must be guided by an expert UAV pilot.
  - The drone must keep a safe distance margin of at least 50cm, to ensure the drone's integrity.
  - While recording with the RunCam, the camera-structure distance should be at most 1m to acquire images with recognizable defects.

5.3 Damage detection algorithm

The area of image processing has grown drastically in recent years. One of the most promising advancements in this field is the convolutional neural networks (CNNs) [21]. In general, deep learning (DL), including CNNs, are trained rather than programmed, and applications using these techniques often require less expert analysis and fine-tuning [22]. Trying to exploit these features in the field of infrastructure inspection, deep supervised learning with CNN was the approach adopted for the automatic detection of damages in a bridge.

Accurate damage detection can be achieved with image segmentation, L. Yang et al. [15] used this technique employing the VGG-16 model architecture [23]. The approach was deciding an appropriate size sliding window that could classify crack or spalling based on image segments. For each window in the image, a CNN must be applied, thus increasing the computational cost.

In inspection images, there can be multiple cracks and spalling. Another suitable method to detect and locate infrastructure defects is CNN-based object detectors. Unlike the image segmentation that outputs one value per pixel with the assigned category (classification), the object detection outputs bounding boxes that indicate the location and kind of object identified. This improves the efficiency of finding defects by directly predicting object regions [16].

Single- and two-stage object detector categories have been used for detecting crack and spalling [24], [25], [26], [16]. Two-stage object detectors can achieve good accuracy results. However, the region proposal step is a payload that slows down detection for the two-stage detectors [16], and in an environment that utilizes a drone as an inspection system is essential to speed up detection. Single-stage object detectors, on the other hand, can improve speed detection as they do not include the region proposal step. Results given by Redmon and Farhadi [27], [28] have shown that a single-stage object detector as the YOLOv3 could achieve a comparable accuracy compared to a two-stage detector as the Faster Region-based-CNN.

For the aforementioned reasons, we decided to select **You Only Look Once version 4 (YOLOv4)** [6] (an improved version of YOLOv3) as the single-stage object detector to identify structural damages. Although we are not performing detections in real-time for this work, we think that the single-stage object detector fits better for this application.

5.3.1 Structure damage definition

There are several defect categories with which damages can be identified. For example, C. Zhang et al. [16] used four classes: crack, spalling, pop-out, and exposed rebar for their single-stage object detector. They mentioned that class imbalance is a possibility, where damage classes with a larger number of samples may overwhelm training causing inaccurate results. Trying to avoid this problem, we decided to use two categories for the damage identification, crack and spalling. We adopt the definition of spalling as the type of damage where the concrete surface has peeled off, rebars can be exposed or not (see Figure 11).

(a) Spalling with exposed rebar. (b) Spalling with no exposed rebar.

Figure 11: Spalling examples.

5.4 Final system

According to the tests and analysis presented in the previous sections, the design was finally established, as depicted in Figure 12. The proposed solution involves an expert UAV pilot, qualified and authorized to control the drone. The pilot is in charge of operating the UAV securely while attempting to use the sensors most efficiently.

The hand-operated UAV proposed and used in this project was the Phantom 4 Pro (DJI), which has integrated a camera able to record pointing at the horizon and downwards. In addition, the RunCam camera is installed on the drone. This camera is directed at the zenith angle, facilitating the data collection of the underside of a bridge deck when the drone is under it.

The pilot then guides the UAV around the bridge while the cameras are recording; the obtained videos are stored in a microSD. Once the flight has finished, the videos are extracted for further offline processing with the YOLOv4 model for damage detection. A new video is generated where the damages are enclosed by colored boxes. The green boxes indicate the presence of spalling, whereas the pink ones show the cracks.

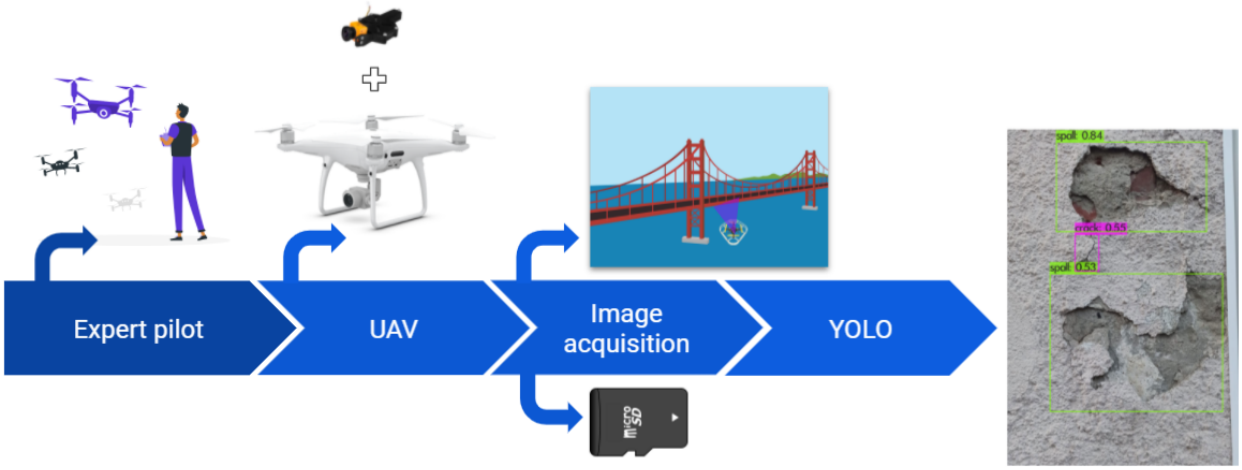


Figure 12: General diagram of the final system design.

6 Data Acquisition

The data for the algorithm training and testing were collected from five different sources. Table 2 presents these data distributions, their total amount of images, and the number of images that contain spalling, crack, both classes of damages, or any damage.

Combining the images of the five data distributions, we gather in total 7195 images, from which 1989 contain cracks, 3568 include spalling, and 1477 have both damages. This value evidenced a slight class imbalance that suggests that the algorithm could perform better for spalling detection than crack identification as it has more data to train from this class of damage. From these sources, some images with no defects were selected so that the algorithm can learn also the unharmed aspect of a bridge structure. In the following subsections, each distribution will be explained in detail.

Data distribution	Number of images	Crack	Spalling	Both damages	No damage
Google Images	829	178	277	313	61
Google Maps	34	11	11	8	4
Infra-Scan	1178	93	831	158	96
Paper	4551	1104	2449	998	0
Kaggle	603	603	0	0	0
Total	7195	1989	3568	1477	161

Table 2: Characteristics of the data distributions.

6.1 External data sources

For the data collection we used, in the first place, the Google Images search engine. The first searches were made by using specific keywords such as spalling, crack, and bridge damage. After that, we took advantage of the function "Search by image", to find more images similar to the previously found. The second source of images was Google Maps. In this case, the street view was used to acquire some screenshots from bridges in Colombia. Besides these manual searches, we found two already existing datasets. The first one was found on Kaggle, the online community of data scientists. It consisted of a dataset of 20000 images of cracks. From it, 603 images were used for this project. The second dataset was found on [15]. This data source was the biggest contributor of images for this project, we called this distribution *Paper* for the sake of simplicity.

Figure 13 shows examples of the images contained in the Google Images, Google Maps, Kaggle, and Paper distributions. It is observed that the Kaggle dataset and the Paper distribution contain images in which the camera was probably very close to the surface, giving as a result damages which are considerably big with respect to the image size. These images are not realistic, in a real scenario the drone will keep a safe distance from the bridge preserving its integrity. Nevertheless, these images are useful for training; with them, the algorithm will be able to learn low-level features such as edges and textures. In addition, as mentioned before, the paper dataset is the biggest contributor of data. Therefore, it is expected that it can have a great influence on the algorithm's behavior.

The Google Maps distribution contains images of different bridges simulating in this way a real scenario. However, the bridges found through Google Maps do not present considerable damages. In our search, we did not find bridges in significantly bad conditions. Therefore, in this dataset, there are some samples of hardly appreciable damages from which the algorithm could learn as well. Notice that this data distribution contains the lowest amount of images.

Finally, the Google Images dataset contains samples of images taken from the internet. This dataset has useful images, from which a considerable percentage are samples of scenarios very similar to our case study. Nonetheless, there are images of damages in other kinds of buildings and textures. This distribution has a variety of images with different characteristics such as size, source camera, and backgrounds.

(a) Google Images.

(b) Google Maps.

(c) Kaggle.

(d) Paper.

Figure 13: Examples of the distributions of images.

6.2 Infra-Scan system data acquisition

The remaining images were collected in collaboration with **DISEG** (Dipartimento di Ingegneria Strutturale, Edile e Geotecnica) and professor **Francesco Tondolo**. The UAV system was used to collect images in the test field **Bridge|50** project from the laboratory **SISCON**. In this field, there are several structures affected by crack and spalling from different parts of a bridge. Figure 14 shows a high view of the test field.

Figure 14: High view of the test field.

The data collection in the field was divided into two sessions. The first one was held on April 8th, 2021. From this data retrieval, 57 videos were recorded, and 235 images were extracted. The second session was made on May 6th, 2021. In it, 88 videos were recorded from which 943 images

were obtained. The pilot who guided the drone through the structures was **Professor Vincenzo Di Pietra**. For sake of simplicity, we called *Infra-Scan* to this dataset. Figure 15 shows some pictures of the activity held in the test field.

Figure 15: Activity held in the test field.

Figure 16 presents some examples of images collected with the Infra-Scan system in the test field. The Infra-Scan dataset is considered our target data. This means that it contains the most realistic images that were taken with real equipment, to a real bridge structure. We would like the algorithm to learn the most from this data, increasing the possibilities to have effective object detection in a real bridge inspection.

Figure 16: Examples of the Infra-Scan dataset.

In general, we can observe that the images from Google Images and Google Maps are very similar to the target, the images in those cases contain different backgrounds, bridges, and damage sizes are diverse with respect to the total image size. On the other hand, the Kaggle and the Paper data are convenient to help understand the model basic defect features. Our intention in building these distributions is to create a balance between them so that the algorithm can learn to predict from our target where and which damage is present in the picture, while it is learning the primary characteristics of the defect from the other distributions.

7 Test plan

The test plan was mainly focused on the damage detection algorithm since the UAV system was validated during the design phase, and its functioning was verified during the data collection. In this way, the testing consisted in implementing different strategies to improve the performance of the algorithm.

As mentioned in Section 5.3, the algorithm selected for our object detection problem was the YOLOv4. In particular, we used the model architecture implemented in the repository created by *AlexeyAB* [29]. The principal parameters selected for the model training are presented in Table 3.

Parameter	**Value**
Batch size	1 image per batch
Width of the images	416 pixels
Height of the images	416 pixels
Number of channels	3
Momentum	0.949
Learning rate	0.001
No. of iterations the training will be processed	6000
No. of iterations by which the learning rate is multiplied by *scales* factor	4551
Scales factor	0.1

Table 3: Parameters of the algorithm.

In addition, transfer learning (TL) was employed for fine-tuning the damage detection network. This method consists in using a pre-trained model as starting point for the convolutional layers. In this case, the weights have been pre-trained on the 80 classes in the Common Objects in Context (COCO) dataset which is one of the most popular open source object recognition databases and includes hundreds of thousands of images with millions of already labeled objects for training [30].

7.1 Methodology

For the test planning, we followed the guidelines given in the course Structuring Machine Learning Projects of the Deep Learning Specialization offered by deeplearning.AI [31]. The methodology consisted in a Machine Learning iterative process composed of three main phases as presented in Figure 17. In the first stage, an idea is proposed. After that, in the second phase, the idea is implemented in code. Finally, an experiment is carried out to see how well the formulated idea works. At the end of the entire process, a new idea emerges and the cycle starts again. The faster we can go around this loop, the quicker progress is made. In this work, four cycles in total were carried out. Their details are presented in the following subsections.

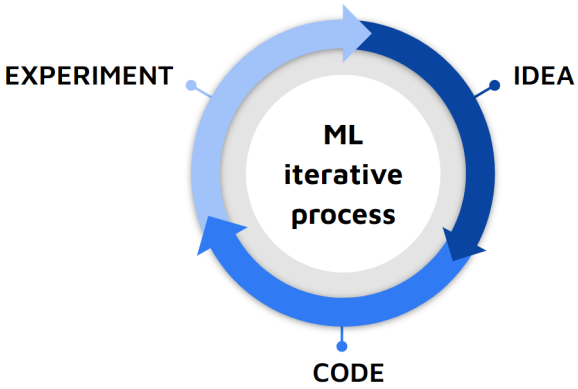


Figure 17: Machine Learning iterative process.

7.2 Mean average precision

To evaluate our algorithm is necessary to define a metric that indicates how well we are detecting crack and spalling. For object detection, the mean average precision (mAP) is used to evaluate the performance of a model.

Two terms that need to be understood first to calculate the mAP are precision and recall. **Precision** indicates how many of the predictions labeled as "positive" were actually "positive", i.e., the percentage of the correct detections. Precision metric is given by Equation (1)

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

where TP stands for true positives (predicted with sufficient IoU) and FP stands for false positives (predicted with insufficient IoU).

In object detection it is considered a true positive, a detected object that equals or surpasses a threshold of the intersection over union (IoU). The IoU measures the overlap of the predicted with the ground truth bounding boxes as illustrated in Figure 18.

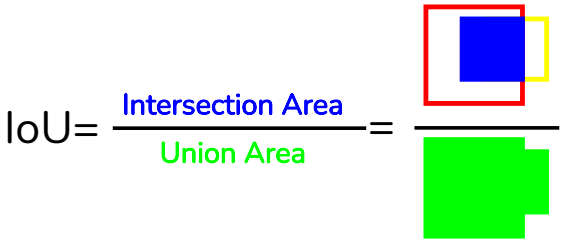


Figure 18: Intersection over area.

Recall, on the other hand, among all true positives (ground truth), calculates the percentage of those predicted as true positive. Those that were not detected and indeed were true positives are considered false negatives. Recall metric can be calculated by Equation 2

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

where TP stands for true positives (predicted bounding box as positive and was correct) and FN stands for false negatives (failed to predict an object that was there).

The average precision (AP) must be computed for all the classes, and then we calculate the mean across all the classes to obtain the mAP. The AP is determining the area size under the precision-recall curve and is calculated as explained in [32].

In this work, we set the IoU threshold to be more or equal to 50 %. Thus, a predicted bounding box is considered a true positive if the IoU is equal to or above 0.5. Otherwise, the predicted bounding box is treated as a false positive. From now on, to consider the IoU threshold of 0.5 the mAP metric is referred to as **"mAP@0.5"**.

7.3 Cycle 1

Knowing beforehand the development set (also called validation set) can save valuable time during the iterative process explained in Section 7.1. Defining a development set is important because when the model is assessed with the mAP@0.5, we can see how well it is performing on unseen data. Thus, for the first cycle of the iterative process, we first want to select our development set.

Considering the dataset explained in Chapter 6, we decided to divide train, development, and test sets into 80, 15, and 5 percent, respectively. We proposed three options of data grouping (explained in Sections 7.3.1, 7.3.2, and 7.3.3) and selected the more pertinent to our application.

7.3.1 Option 1

The first option to build train, development, and test sets is to combine all the data distributions together and shuffle the data. Once the data is shuffled, 80 % goes to train, and 15 % goes to development. Finally, the remaining 5 % of data is included in the test set. The data comprised in test, only consists of images taken by the Infra-Scan system. Figure 19 shows how the data grouping was performed for this option.

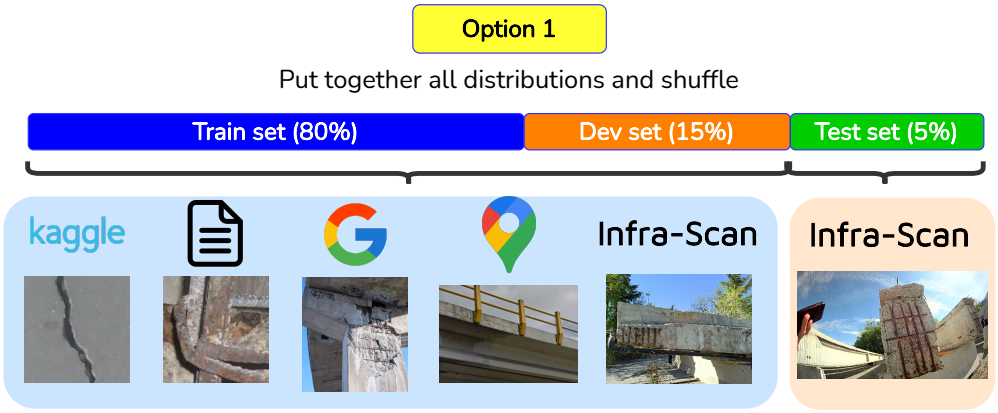


Figure 19: Data grouping option 1.

The advantages and disadvantages of organizing the train and development sets with this approach are as follows:

- PRO:
  - Development and train sets will come from the same distribution of data. In this case, we could generalize better to different scenarios.
- CONS:

– Development set will contain a large share of images that are not our target.

7.3.2 Option 2

The second idea to group the data distributions is to extract a larger share of images taken by the Infra-Scan system and include them in the development set. With this proposal, we aim better at our target since we evaluate more data taken with our system. The Infra-Scan dataset is divided into 25 % for the train set, 50 % on the development set, and 25 % on the test set. The train and development distributions are completed with shuffled data from the remaining datasets, while test is only composed of the Infra-Scan samples (see Figure 20).

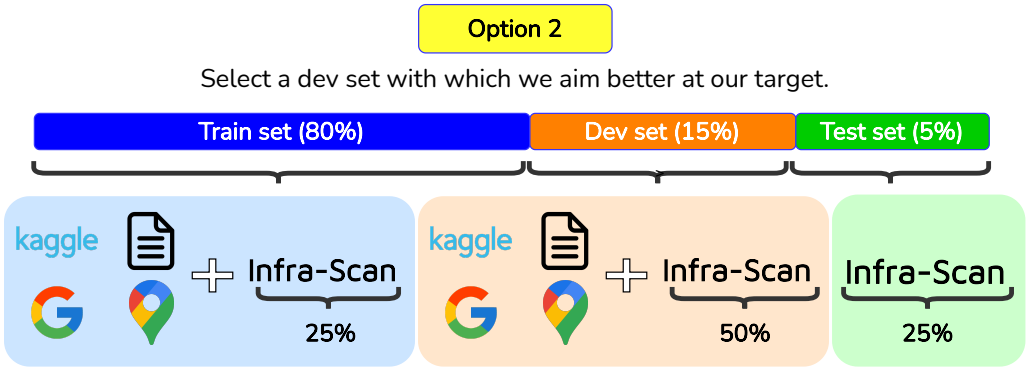


Figure 20: Data grouping option 2.

The advantages and disadvantages of organizing the train and development sets with this approach are as follows:

- PRO:
  - Having more Infra-Scan data in the development set allows an evaluation focused on the target task. We are aiming the target where we want it to be.
- CONS:
  - The training set will contain less target data, thus the learning on the target task could be reduced.

7.3.3 Option 3

With the configuration of option 3, we search for a suitable organization of data to generalize better to several scenarios. In this case, we follow the same distribution of Infra-Scan images as explained in Section 7.3.2. Additionally, the Google Images and Google Maps samples are divided 30 % into the train set and 70 % on the development set. Finally, the test set contains only images from the Infra-Scan cameras, as illustrated in Figure 21.

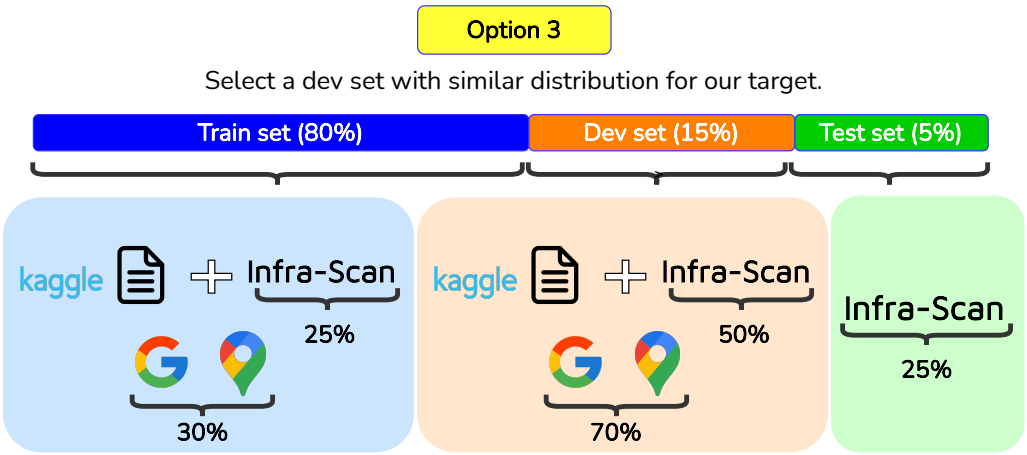


Figure 21: Data grouping option 3.

The advantages and disadvantages of organizing the train and development sets with this approach are as follows:

- PRO:
  - Having more Infra-Scan and Google data in the development set allows an evaluation focused on the target task and in similar scenarios. We can evaluate the generalization to different situations.
- CONS:
  - The training set will contain fewer target data and similar data. Thus, the learning on the target task could be reduced, and as a consequence, the difficult features such as complex damages (e.g., little cracks) might not be learned.

7.4 Cycle 2

After defining the most appropriate data grouping presented in Section 7.3, the objective was to keep improving the model performance (i.e., keep improving the mAP). The inclusion of a transfer learning (TL) method in the second cycle of the iterative machine learning process was the idea to continue enhancing the model.

A pre-trained model with a similar task can transfer more feature layers ("knowledge") to a target task [33], and resembling visual characteristics, such as color, and texture from the pre-trained and target data could also help in this matter [34], [35]. Based on this, we decided to imitate the novel TL method proposed by [16].

In the first cycle (Section 7.3), YOLOv4 was trained with the pre-trained weights on the whole COCO dataset (Transfer learning method A (TL-A)). In this cycle, the fine-tuning process for the single-stage object detector was extracting a similar portion of the COCO dataset first. Then, the pre-trained weights obtained from this process are used to fine-tune the damage dataset (Transfer learning method B (TL-B)). In this way, we can extract the common low-level features from COCO and fine-tune with these features the damage dataset and hopefully obtain better results. Figure 22 shows both of the TL methods employed.

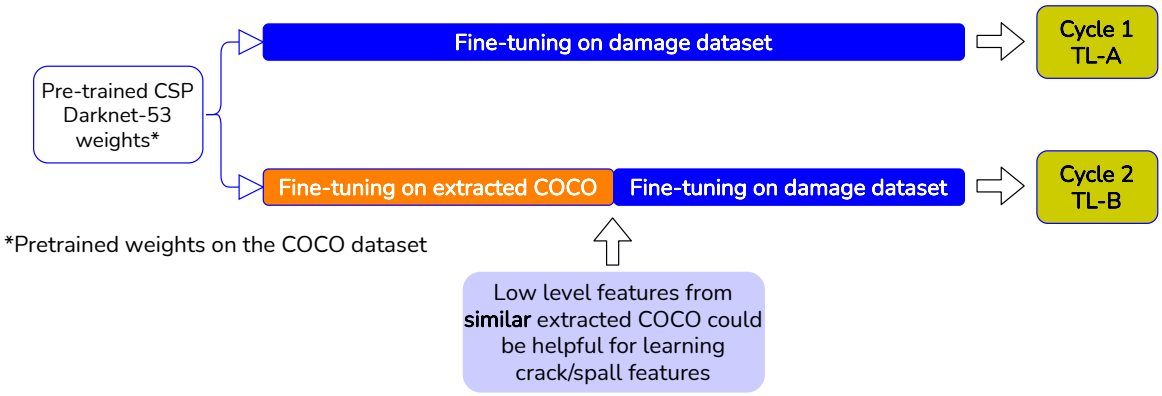


Figure 22: Extracted COCO transfer learning.

Zhang et al. [16] used the scale and the aspect ratio as the similarity metric to extract the annotated images from the COCO dataset. The object scale is defined as the size of the bounding box surrounding the object normalized by the total image size containing the damage. The aspect ratio is the ratio between the width and the height of the bounding box.

The scale and the aspect ratio for each of the 80 classes in the COCO dataset were measured and then compared to those in our damage dataset. For each class, the object scale was divided into 20 equally spaced intervals. For each interval, the number of bounding boxes was counted. The resulting amount of bounding boxes was divided by the total number of them to obtain the percentage of objects. Similarly, for the aspect ratio, space was divided and distributed into 20, as illustrated in the x-axis of Figure 24. Both similarity measures from each of the COCO classes and the damage dataset were confronted computing the sum of the Euclidean distances. The sum was sorted in ascending order, and the top 4 classes, shown in Table 4, were selected for the extracted COCO.

Figures 23 and 24 show the comparison of the similarity measures. The four selected classes from the COCO dataset were combined together to calculate the statistics. For the object scale, the damage dataset compared with the extracted COCO show less disparity than the whole COCO with the damage set. We can understand also from the object scale that our damage dataset contains about 10 % of damages that corresponds to at least the 95 % of the total image size.

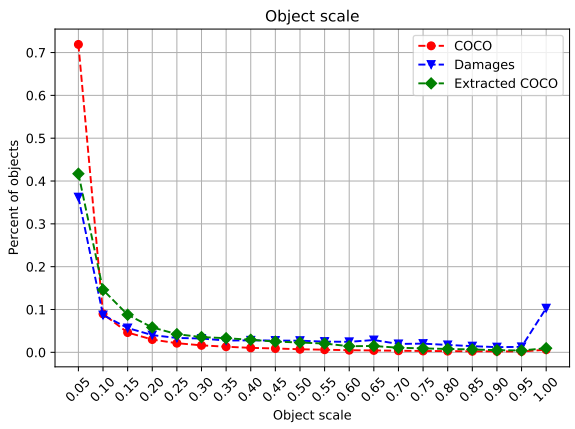


Figure 23: Similarity object scale.

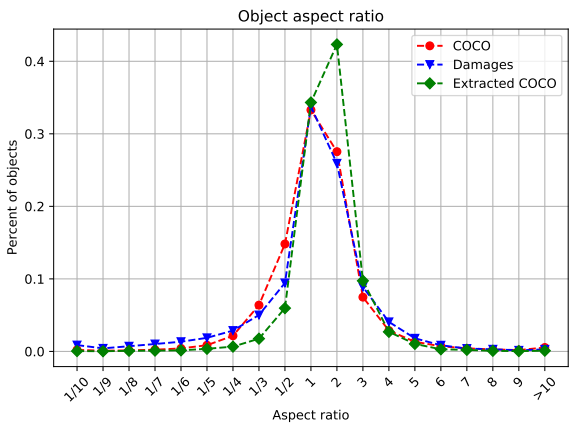


Figure 24: Similarity aspect ratio.

From the similarity aspect ratio, even the whole COCO dataset seems to better fit the damage dataset, especially, for the range comprising 2-3. However, from the results in Table 4, the sum between the two similarity measures still gives the edge to those 4 top classes instead of the whole COCO dataset.

Class	**Object scale Euclidean distance**	**Aspect ratio Euclidean distance**	**Sum of Euclidean distances**
69-Oven	0.1641	0.1159	0.2801
16-Dog	0.1258	0.2228	0.3486
5-Bus	0.1247	0.2346	0.3593
76-Scissors	0.2703	0.0950	0.3654

Table 4: Top 4 selected classes for extracted COCO.

7.5 Cycle 3

For the third cycle in the iterative machine learning process, the idea is to perform data augmentation to improve the mAP@0.5. The intention with the data augmentation is to cope with the limited amount of drone images and simulate a real scenario by adding slightly modified copies of already existing training data. For example, the motion blur can be helpful since the drone is moving, and it might capture these kinds of pictures. Also, the brightness under a bridge could be reduced, and illuminated scenes could harm the training as it might not learn the features from the damages under complex conditions. So, generating images with these characteristics could help the algorithm understand better situations that the drone can encounter.

For the data augmentation, we selected a set of images from the Google Images and Google Maps distributions; compared with the Kaggle and Paper datasets, the Google distributions resemble better the scenario of the pictures taken by the drone. Moreover, all the images captured with our sensors were included in the data augmentation process. Overfitting could be one of the possible problems from the data augmentation procedure. To be cautious, for each selected image a maximum of five can be generated. We proposed four categories in which new images can be produced:

- **Motion blur:** These images are generated with a probability p equal to 50 %. As it was mentioned, the drone can capture images while moving, and the effect of blur might be present.
- **Cropped:** These images are generated with a probability p equal to 50 %. The cropped image gives a new dimension to the damage in the picture, simulating that sometimes the drone can be nearer to the structure. The original image is randomly cropped up to 65 % of its total size.
- **Horizontal flip:** These images are generated with a probability p equal to 70 %. The damages can be very similar from one structure to another. For this reason, a horizontal flip could be useful as it simulates the same damage but in a different structure.

- **Brightness:** These images are generated with a probability p equal to 80 %. The new images generated with this technique, i.e., darker images, imitates the reduced amount of light under a bridge.

An example of how new images from all the four categories can be created is illustrated in Figure 25.

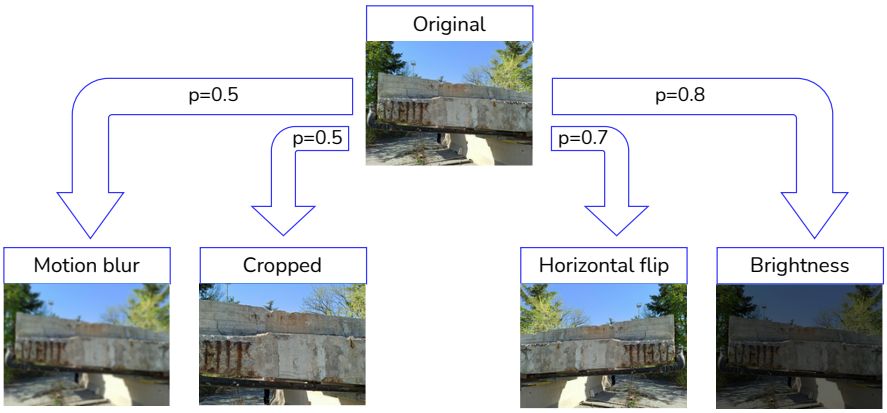


Figure 25: Data augmentation process.

7.6 Cycle 4

After completing the first three cycles, the one-stage damage detector was tested on the target data. The test set consisted only of images taken by the RunCam and the Phantom 4 Pro's camera. As presented in Table 7, the results on the mAP@0.5 tested in option 1 took a downturn compared to the development set (validation set). There is **data misalignment** between development and test set. Therefore, the results obtained in the previous cycles were not reflecting the real performance of the model in our target data.

The development set was overwhelmed by *"easy"* data. As stated in Section 6, Kaggle and Paper distributions were useful to the algorithm to train on the basic damage features. However, it is unlikely to find these kind of images in a bridge inspection campaign, and when they were included in the development set, we were not aiming at our target data. The development set contained 1046 samples with data of the Kaggle and the Paper distributions and 205 images of Google and Infra-Scan distributions.

To exhibit the data misalignment, we divided the development set into two, the "easy" data with the Kaggle and Paper images; and the "target" data with the Google and Infra-Scan samples. With this division, we evaluated again both with the mAP@0.5. Table 5 presents the results.

Model	**Best mAP@0.5, validation on Kaggle and Paper images**	**Best mAP@0.5, validation on Google and Infra-Scan images**
Option 1 TL-A	84.38 % (AP Crack: 72.73 % AP Spalling: 96.03 %)	28.63 % (AP Crack: 19.06 % AP Spalling: 38.20 %)

Table 5: Demonstration of data misalignment on development set.

The purpose of the fourth cycle in our iterative process is to define a proper development set so that the results given by the mAP@0.5 can be trusted. In this process, we also defined again the training set, we call this new arrangement of data distributions "option 4", which is fully explained in Section 7.6.1.

7.6.1 Option 4

We took the data from option 1 (option that shuffled all the data and was combined together into train and dev sets) and we distributed in a different way:

- **Train set:** The Kaggle and Paper images that originally were in the development set of option 1 now are part of the train set in this configuration. Besides, from the 943 new samples taken in the second data collection (held on May 6th, 2021), 216 were inserted. Finally, following the good results obtained by including augmented data (see Table 7), for this fourth option, the data augmentation was also included in the train set.
- **Development set:** The Kaggle and Paper images were eliminated, leaving only the samples from the Google distributions and the Infra-Scan cameras. The decision of leaving the Google distributions is the similarity of them regarding our target data, so the evaluation could also generalize to different scenarios with analogous data. Additionally, in this dataset were included 436 samples from the second data collection aiming towards our target.
- **Test set:** To test the algorithm, we took the 291 remaining images from the second data collection, all of them are unseen data for the model.

The new development and test sets from "option 4" became the composition of data to evaluate the performance of the proposed models (see results on Section 8.3).

8 Results

8.1 Cycle 1

To advance with the iterative process is imperative to select the development set. In this section, we evaluate which of the data grouping options (explained in Chapter 7.3) is more suitable to the Infra-Scan system.

YOLOv4 was trained with the pre-trained weights on the whole COCO dataset and evaluated on mAP@0.5 for each of the options. The results are summarized in Table 6. Option 1 had the better results on the development set since it trains in the most difficult data (there is a larger percentage of Infra-scan and Google images). Options 2 and 3, on the other hand, performed better on the train set. A reasonable explanation is the larger number of Kaggle and Paper samples. In these images, the defects are more recognizable. Nevertheless, the model with these two options produced low-grade validation results because the data in the development set is more difficult than the data that it was trained in.

	Option 1	Option 2	Option 3
mAP@0.5 on train set	84.08 % (AP Crack: 75.56 % AP Spalling: 92.60 %)	86.57 % (AP Crack: 78.66 % AP Spalling: 94.49 %)	89.48 % (AP Crack: 82.80 % AP Spalling: 96.17 %)
mAP@0.5 on dev set	71.18 % (AP Crack: 60.09 % AP Spalling: 82.27 %)	64.05 % (AP Crack: 56.95 % AP Spalling: 71.15 %)	41.05 % (AP Crack: 31.79 % AP Spalling: 50.32 %)

Table 6: Train and validation performance for the three data grouping options.

The issue with the data is the quantity of target data. With only the 16 % of Infra-Scan images and 12 % of Google samples, it becomes hard to leave most of this data in the development/test set as the algorithm will not recognize the most difficult damages features (e.g., small cracks in unexpected locations or spalling covered by shadows). Option 2 and 3 could be suitable if we have considerable amounts of target and similar data so the algorithm can also learn with sufficient amounts of data. However, this is not the case, the target and similar samples are limited.

To continue with the other cycles, we chose option 1, as it can train with more similar and target data so the algorithm can learn better for the target task. In the training set, option 1 has about 13 % more target (Infra-Scan) and similar (Google) data compared to option 2, and about 170 % more considering option 3.

8.2 Cycles 2 and 3

Once chosen the option 1, the transfer learning on the extracted COCO (TL-B) was performed. Then, to compare how well the TL-B was transferring the knowledge to our target task, we added the augmented data to the TL-A and TL-B in two different runs. Table 7 summarizes the results on the development set (dev set).

Model	Best mAP@0.5 on dev set	Best mAP@0.5 on test set
TL-A	71.18 %	21.11 %
TL-B	71.42 %	20.13 %
TL-A + Aug	72.10 %	26.00 %
TL-B + Aug	70.68 %	25.92 %

Table 7: Validation and test results for data augmentation (Aug) and transfer learning methods (TL-A & TL-B) for option 1.

There is not a significant difference in applying TL-A to TL-B from the results presented in Table 7. First, only employing the transfer learning methods without any data augmentation, there is a negligible advantage for the TL-B, the result is not conclusive. Then when adding the augmented data (Aug), applying TL-A showed slightly better results than TL-B. In fact, TL-B worsened its performance with the increased amount of data. A reasonable explanation for these results is that

the extracted COCO describes better the data without augmentation. Therefore, the transferred knowledge to the training dataset with augmentation is hurting the model. The TL-A method, instead, is providing better information to the target task when the data is augmented.

One would be expecting that applying the TL-B would translate into improved performance as the extracted COCO portion of data had *similar* characteristics to the damage dataset. From the analysis made in Section 7.4 maybe an improvement to select the objects in the COCO dataset is to give more weight to the aspect ratio because giving the same amount of importance to the object scale and the aspect ratio might not be transferring the important information. As seen in Figure 24, there is a portion of the extracted COCO that has en elevated percentage of objects with an aspect ratio between 2 and 3 (about 40 %), and for this same interval the damage dataset has a considerable lesser amount, about 10 %. The transferred features from the disparate aspect ratio might be influencing the little improvement on the target tasks.

8.3 Cycle 4

To justify the new approach adopted with the fourth cycle, all the options were again evaluated on the test set explained in Section 7.6.1. The results are summarized in Table 8. Configurations one through three were not validated again in the new development set. This set of images also included samples in the train set for the three options, so the validation would be biased at having good results, as some of the data is known beforehand.

Model	**Best mAP@0.5 on new dev set**	**Best mAP@0.5 on test set**
Option 1 (TL-A)	N/A	21.11%
Option 2 (TL-A)	N/A	18.66%
Option 3 (TL-A)	N/A	15.59%
Option 1 (TL-A + Aug)	N/A	26.00%
Option 4 (TL-A + Aug)	30.22%	40.97%

Table 8: Validation and test results on test and new development set.

Options 2 and 3 can be discarded, as in the test set both performed poorly compared to the other options. Indeed, from these results, the fact that option 1 is the arrangement of data to continue the improvements for the model is supported.

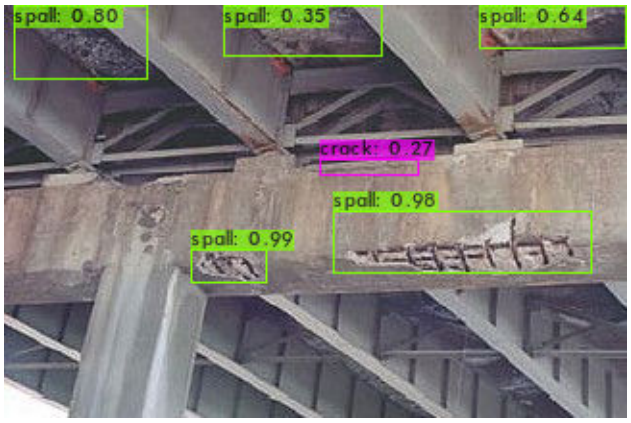
With the development set, the model is evaluated in a variety of situations, considering not only the samples taken with the Infra-Scan cameras but also including the Google distributions that add different background information and relatively small defects. The test set is built to assess the models with images taken with the Infra-Scan system, that is, obtaining infrastructure damage information with a drone (i.e., our target task).

For our task, which is obtaining images with RunCam and the Phantom 4 Pro camera while guiding the drone following a route scanning a bridge infrastructure, the best option is number 4 as it had the best results on the test set. The model still needs to be improved to generalize better to other scenarios as the validation on the development set showed worse results than in the test.

8.4 Analysis of results

Even obtaining a low mAP@0.5, the qualitative results coming from the testing videos are quite good as it correctly detects most of the defects in the bridge structures. Figure 26 displays detection examples for the Option 4 (TL-A + Aug) model.

(a) Google Images sample.



(b) Google Images sample.

(c) Infra-Scan sample.

(d) Infra-Scan sample.

Figure 26: Detection samples from Option 4 (TL-A+Aug).

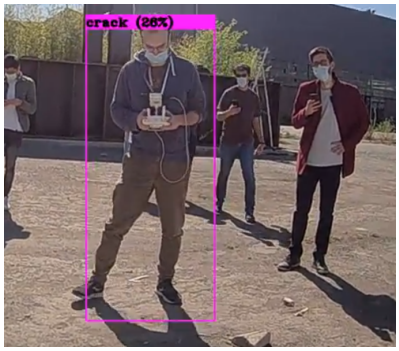
Images from Figures 26a and 26b are included in the validation set. The spalling in the pictures is correctly detected and with high class probability (close to 1). It is worth mentioning that the model can make good generalizations (at least qualitatively) to complex scenarios with different backgrounds, blur, lighting, size of the defect, and so forth.

Figures 26c and 26d show Infra-Scan samples that are included in the test set, where the spalling is correctly detected. The structure and defects shown in the pictures are the same but taken at different angles and slightly different camera-damage distances. This indicates that the model maintains a consistent behavior even if the drone is moving.

Figure 27 depicts examples that the trained model wrongly detects as structural damage. In Figures 27a and 27b, the network distinguished shadows as crack and spalling, respectively. The change in lighting from an illuminated surface to a darker one appear like a spalling. To mitigate these kinds of false positives, one possible solution is to include images with shadows that are not labeled in the training set so that the algorithm can learn that a brightness change is not a structural defect. Similarly, for the people that are detected as faults (Figure 27c), the model can be fed with non-labeled human pictures.

(a)

(b)



(c)

Figure 27: False positives examples from Option 4 (TL-A+Aug).

9 Conclusions and future work

In this work, we have evaluated the capabilities of the Geoscan Pioneer system to guide the drone automatically through a route to inspect bridge infrastructures. We tested the most suitable sensors provided by the manufacturer, which on paper seemed appropriate for the application. We found that the GNSS module estimated position is affected by the camera placement on the UAV.

For the selected camera, we determine that the distance for acquiring high-quality images is at most 1 m, defining a standard that can be replicated in future tests. Unfortunately, after assessing the system as a whole, we determine that for an automatic flight, it is not convenient to use this system. Apart from the irregular automatic flight, the trade-off between GNSS estimated position and cameras resolution were the factors that determined that Geoscan Pioneer system was not adequate for our application. Instead, we proposed a hand-operated drone with which it is possible to navigate a bridge infrastructure, the Phantom 4 Pro (DJI).

We suggested the YOLOv4, a single-stage object detector, to identify structural damages. The accuracy-time efficiency combination to detect the defects were the optimizing-satisficing metrics to select this model. Considering the mAP@0.5, option 1 with TL-A was the model that might generalize better to different scenarios. Instead, option 4 with augmented data and TL-A is the model that for the task of inspecting a bridge with the RunCam and Phantom 4 Pro cameras might have the best results.

To further improve the algorithm, it is advised to recollect more images, especially those containing little cracks, so the model can learn better the features for this kind of damages and hopefully get better AP for the class "crack". Moreover, trying to avoid possible false positives, we can include images that do not contain any crack nor spalling but are similar, such as hieroglyphs or shadows that resemble them.

The work can continue with:

- The inclusion of the extension of the damage, record it for a future inspection and compare it. Knowing the dimension of the defect can provide more information for the inspector. The size can suggest the origin of the defect, e.g., if the defect was generated by corrosion or if the fault was caused by the load stress in the infrastructure. Moreover, the extent of the damage can be an indicator of the level of attention that should be given depending on the hazard. A Stereo vision camera can be used to estimate the distance and size of the crack/spalling.

- As stated in Section 5.3.1, there are multiple categories with which an infrastructure defect can be classified. In this work, we treated crack and spalling as two wide definitions to cover the faults due to possible problems of class imbalance given the amount of data. For a future application, it is possible to think of adding more defect categories for detection. However, to proceed with this kind of approach, we advise having high amounts of data that contains preferable a balanced number of structural defects.

10 Acknowledgements

The INFRASCAN team would like to thank the Department of Structural, Geothecnical and Building Engineering (DISEG), professor Francesco Tondolo, and professor Pierclaudio Savino for the support given throughout the project and granting us the possibility of acquiring images in the test field: Bridge|50 from the laboratory Safety of Infrastructures and Constructions (SISCON). Also we want to thank professor Marco Piras and professor Vincenzo Di Pietra for all the recommendations, support, and feedback during the project.

References

[1] Colleen Barry. Bridge collapse highlights Italy's aging infrastructure. [Online] Available on: https://apnews.com/article/a3ef13dbbd3a4bb8b4bf8b2d316a3d9b, 2018. Accessed: 25.05.2021.

[2] Marco Bertacche and Alberto Brambrilla. No one knows what to do about Italy's deadly bridges. [Online] Available on: https://www.bloomberg.com/news/features/2020-03-06/italy-s-deadly-bridges-crumbling-infrastructure-poses-dilemma, 2020. Accessed: 25.05.2021.

[3] European Comission. Keeping European bridges safe. [Online] Available on: https://ec.europa.eu/jrc/en/news/keeping-european-bridges-safe, 2019. Accessed: 25.05.2021.

[4] The guardian. What caused the Genoa bridge collapse – and the end of an Italian national myth? [Online] Available on: https://www.theguardian.com/cities/2019/feb/26/what-caused-the-genoa-morandi-bridge-collapse-and-the-end-of-an-italian-national-myth, 2019. Accessed: 25.05.2021.

[5] Ministero delle Infrastrutture e dei Trasporti. Linee guida per la classificazione e gestione del rischio, la valutazione della sicurezza ed il monitoraggio dei ponti esistenti, 2020.

[6] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[7] FlyAvility. Bridge inspections: A complete guide. [Online] Available on: https://www.flyability.com/bridge-inspections. Accessed: 03.06.2021.

[8] Stefanie Zollmann, Denis Kalkofen, Christof Hoppe, Stefan Kluckner, Horst Bischof, and Gerhard Reitmayr. Interactive 4d overview and detail visualization in augmented reality. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 167–176, 2012.

[9] Stefanie Zollmann, Christof Hoppe, Stefan Kluckner, Christian Poglitsch, Horst Bischof, and Gerhard Reitmayr. Augmented reality for construction site monitoring and documentation. *Proceedings of the IEEE*, 102(2):137–154, 2014.

[10] Nathan Michael, Shaojie Shen, Kartik Mohta, Yash Mulgaonkar, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, Kazunori Ohno, Eijiro Takeuchi, and Satoshi Tadokoro. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, 29(5):832–841, 2012.

[11] Fausta Fiorillo, Belén Jiménez Fernández-Palacios, Fabio Remondino, and Salvatore Barba. 3d surveying and modelling of the archaeological area of paestum, italy. *Virtual Archaeology Review*, 4(8):55–60, 2013.

[12] Vincent Lui and Tom Drummond. Image based optimisation without global consistency for constant time monocular visual slam. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5799–5806, 2015.

[13] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, 07 2014.

[14] Junwon Seo, Luis Duque, and Jim Wacker. Drone-enabled bridge inspection methodology and application. *Automation in Construction*, 94:112–126, 2018.

[15] Liang Yang, Bing Li, Wei Li, Zhaoming Liu, Guoyong Yang, and Jizhong Xiao. Deep concrete inspection using unmanned aerial vehicle towards cssc database. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 24–28, 2017.

[16] Chaobo Zhang, Chih-chen Chang, and Maziar Jamshidi. Concrete bridge surface damage detection using a single-stage detector. *Computer-Aided Civil and Infrastructure Engineering*, 35(4):389–409, 2020.

[17] Geoscan LTD. Geoscan pioneer. [Online] Available on: https://www.geoscan.aero/en/pioneer/, 2020. Accessed: 05.20.2021.

[18] Geoscan LTD. Geoscan pioneer instruction manual: Extension modules. [Online] Available on: https://www.geoscan.aero/themes/geoscan/assets /products/tabs/pioneer/manual-en/module/module_main.html, 2021. Accessed: 05.27.2021.

[19] OPENMV. Openmv cam M7. [Online] Available on: https://openmv.io/products/openmv-cam-m7?gclid=Cj0KCQjwkZiFBhD9ARIsAGx FX8Acje5NNY4XNeSvwtg-AnWXWg8xgF37AE62QNjdXozZgbmd5w4OLtoaAga9E ALw_wcB, 2021. Accessed: 05.20.2021.

[20] RUNCAM. RunCam split 2. [Online] Available on: https://shop.runcam.com /runcam-split-2/, 2021. Accessed: 05.21.2021.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[22] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Science and Information Conference*, pages 128–144. Springer, 2019.

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[24] In-Ho Kim, Haemin Jeon, Seung-Chan Baek, Won-Hwa Hong, and Hyung-Jo Jung. Application of crack identification techniques for an aging concrete bridge inspection using an unmanned aerial vehicle. *Sensors*, 18(6):1881, 2018.

[25] Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani, and Oral Büyüköztürk. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):731–747, 2018.

[26] Ruoxing Li, Yachao Yuan, Wei Zhang, and Yali Yuan. Unified vision-based methodology for simultaneous concrete defect detection and geolocalization. *Computer-Aided Civil and Infrastructure Engineering*, 33(7):527–544, 2018.

[27] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[28] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[29] AlexeyAB. Yolo v4, v3 and v2 for windows and linux. [Online] Available on: https://github.com/AlexeyAB/darknet, 2020.

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[31] DeepLearning.AI. Structuring machine learning projects course. [Online] Available on: https://www.coursera.org/learn/machine-learning-projects/home/welcome.

[32] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[33] Minyoung Huh, Pulkit Agrawal, and Alexei A. Efros. What makes imagenet good for transfer learning?, 2016.

[34] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4109–4118, 2018.

[35] Joseph Jaewhan Lim. *Transfer learning by borrowing examples for multiclass object detection*. PhD thesis, Massachusetts Institute of Technology, 2012.

A Gantt Chart

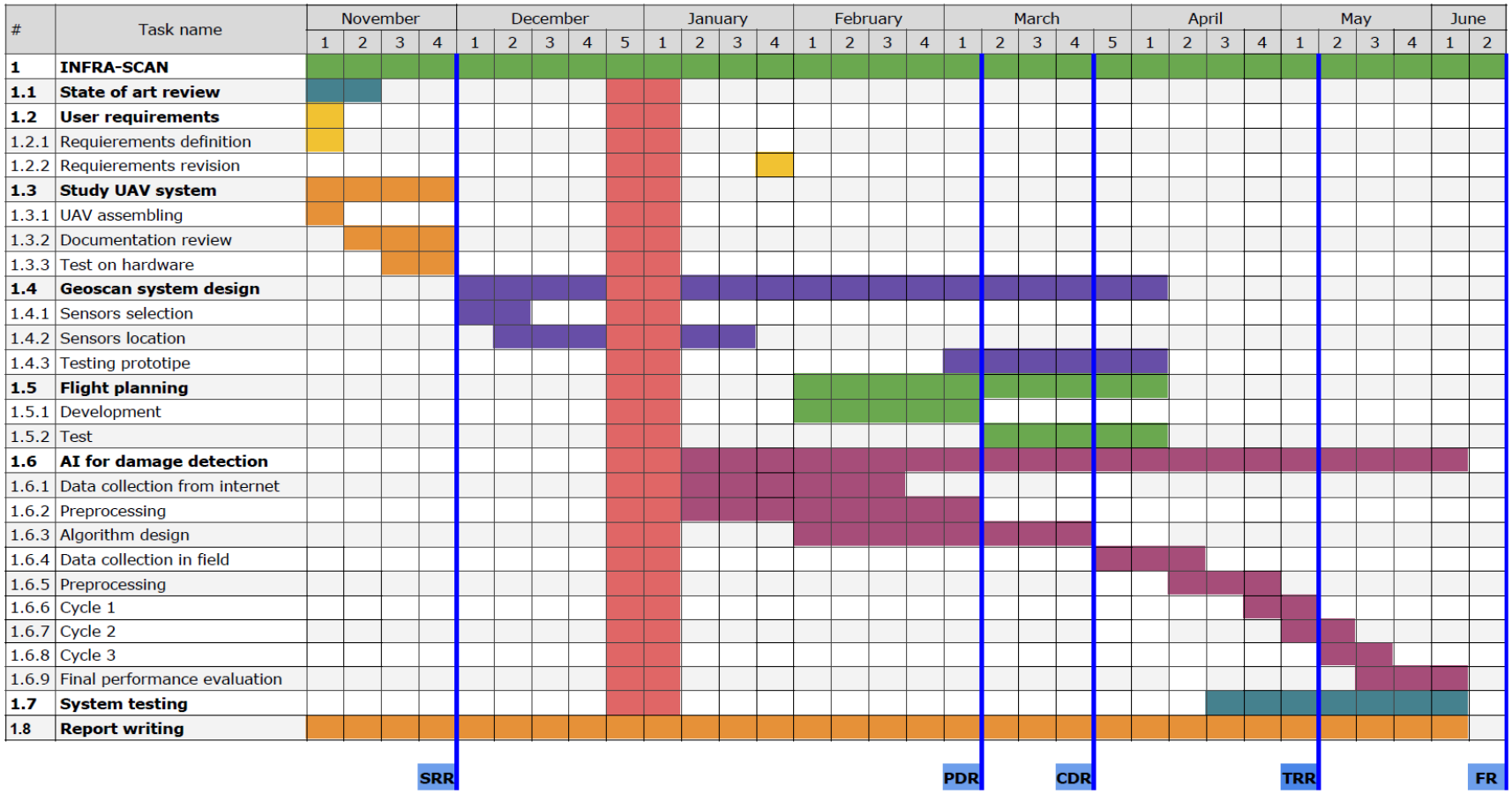


Figure 28: Gantt chart.