D2.5 – Safety-critical situation detection

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement nº 958450. This document reflects only the author’s view and the Commission is not responsible for any use that may be made of the information it contains.

Project Title	Improving Building Information Modelling by Realtime Tracing of Construction Processes
Project Acronym	BIMprove
Grant Agreement No	958450
Instrument	Research & Innovation Action
Topic	Industrial Sustainability
Start Date of Project	1st September 2020
Duration of Project	36 Months

Name and Number of the deliverable	2.5 - Safety-critical situation detection
Related WP number and name	WP2 - Components, technologies & functionalities
Deliverable dissemination level	Public
Deliverable due date	28 February 2022
Deliverable submission date	28 February 2022
Task leader/Main author	Anu Purhonen (VTT)
Contributing partners	Tommi Aihkisalo (VTT), Ilkka Niskanen (VTT)
Reviewer(s)	Christian Dalheim Øien (SINTEF)

Abstract

This deliverable describes how to use the Risk Object Visual Analysis System (ROVAS) for automatic detection of safety measures, especially safety nets, from photographs taken at construction sites. Furthermore, it describes the Risk and Image Data Management Service that has been used to store and provide access to the data created by the ROVAS model. The document contains the links to the developed software and model.

Keywords

Safety measures, computer vision, convolutional neural network, YOLO

Revisions

Version	Submission date	Comments	Author
v0.1	15th February 2022	Submitted for internal review	Tommi Aihkisalo (VTT), Ilkka Niskanen (VTT), Anu Purhonen (VTT)
v0.2	24th February 2022	Ready for SINTEF quality check	Tommi Aihkisalo (VTT), Ilkka Niskanen (VTT), Anu Purhonen (VTT)
V1.0	28th February 2022	Approved, final version	Christian Dalheim Øien (SINTEF)

Disclaimer

Acronyms and definitions

Acronym	Meaning
AI/ML	Artificial Intelligence based on Machine Learning
ANN	Artificial Neural Network
BIM	Building Information Modelling
CNN	Convolutional Neural Network
CURL	client URL
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
REST	Representational state transfer
RDF	Resource Description Framework
YOLO	You-Only-Look-Once

BIMprove project

In the past 20 years, productivity in the European construction industry has increased by 1% annually only, which is at the lower end compared to other industrial sectors. Consequently, the sector has to step up its digitization efforts significantly, on the one hand to increase its competitiveness and on the other hand to get rid of its image as dirty, dangerous and physical demanding working environment. Construction industry clearly needs to progress beyond Building Information Modelling when it comes to digitizing their processes in such a way that all stakeholders involved in the construction process can be involved.

The true potential of comprehensive digitization in construction can only be exploited if the current status of the construction work is digitally integrated in a common workflow. A Digital Twin provides construction companies with real-time data on the development of their assets, devices and products during creation and also enables predictions on workforce, material and costs.

BIMprove facilitates such a comprehensive end-to-end digital thread using autonomous tracking systems to continuously identify deviations and update the Digital Twin accordingly. In addition, locations of construction site personnel are tracked anonymously, so that **BIMprove** system services are able to optimize the allocation of resources, the flow of people and the safety of the employees. Information will be easily accessible for all user groups by providing personalized interfaces, such as wearable devices for alerts or VR visualizations for site managers. **BIMprove** is a cloud-based service-oriented system that has a multi-layered structure and enables extensions to be added at any time.

The main goals of **BIMprove** are a significant reduction in costs, better use of resources and fewer accidents on construction sites. By providing a complete digital workflow, BIMprove will help to sustainably improve the productivity and image of the European construction industry.

Contents

Index of Figures

1. Introduction

The images captured from the risk zones can be analysed by the Risk Object Visual Analysis System, ROVAS, for detecting the existence and placement of the required safety structures. The safety structures are expected to mitigate various risks on the construction site locations, e.g. fall risks. The ROVAS can make only positive detections, i.e. confirming the existence of such structures and thus aid inspecting safety structures' readiness state in the locations for safe construction work. The purpose of Risk and Image Data Management Service is to store and provide access to the results created by ROVAS.

ROVAS will be integrated with the other functionality developed in Work package 2 in BIMprove as described in the Deliverable 3.3 Proof of Concept System Description and Test Results. Earlier use of ROVAS has been demonstrated together with the Risk and Image Data Management Service (See Deliverable 3.1. Technology Demonstrations). However, this service may not be needed after ROVAS has been integrated with the other BIMprove components as its functionality may be included in those components.

2. Risk Object Visual Analysis System

Risk Object Visual Analysis System, ROVAS, bases on the convolutional neural networks, CNN, for its visual object detection task. CNN based AI/ML models are commonly used in modern computer vision systems and object detection. The deployment of such model as functional part of the ROVAS requires gathering the potentially suitable data, manually labelling the risk objects in the data set, installation of suitable tools and frameworks and training and evaluating maturity of it. The maturity of the model will be evaluated by relational development of the achieved accuracy and loss indicators. This is since systems to compare to are mostly non-existent and there are no common publicly shared datasets for rating CNN models in this specific application domain. The capabilities of the ROVAS with the visual object detection mature model will be exposed through the Internet accessible service interface.

The applied CNN model type here is You Only Look Once, YOLO, as described in [1]. Specifically, we applied the further improved open-source implementation of version 5 available from https://github.com/ultralytics/yolov5. This implementation includes also useful scripting and service provisioning examples all licensed with GPL-3.0. For the general principles, intended operational details and limitations see the Deliverable 2.4 Detailed Description of the Safety Functionalities and Worker Notifications.

However, to improve the spatial detection and position determination of the required safety structures, object detection or segmentation could be executed directly on the laser scanned point clouds. While processing point clouds requires a vast amount of computing power, storage space and some point cloud processing limitations it has not been considered at this point of this project, but it is a worth to investigate later.

2.1. Utilized Tools, Methods and Materials

To implement a ROVAS service, training the AI/ML based computer vision model for object detection is necessary. Preceding that, however, a data collection and annotation of it is required in supervised type of machine learning. The original data was provided by the project partners HRS, VIAS and ZHAW including video recorded by VTT from a construction site in Madrid. The group effort by VTT produced a sufficient labelling of the safety nets, barriers and trash appearing in the digital imagery data for the training purposes. The video material was split into individual frames for the labelling purposes. The publicly available labelled image datasets lacked the required safety related items required for this application and thus provided no solution.

The collected and labelled BIMprove dataset has over 3500 individual images split to training and validation sets of 2700 and 800 images, respectively. Each of the images contain one or more occurrences of the labelled items in bounding boxes including safety net, barrier or trash, totalling over 5500 individual labels. The labelling tool used for the YOLO formatted annotation and labelling purposes to classify and draw bounding boxes was the freely available labelImg tool from https://github.com/tzutalin/labelImg. Figure 1 below illustrates the tool and sample image from the dataset with associated labels.

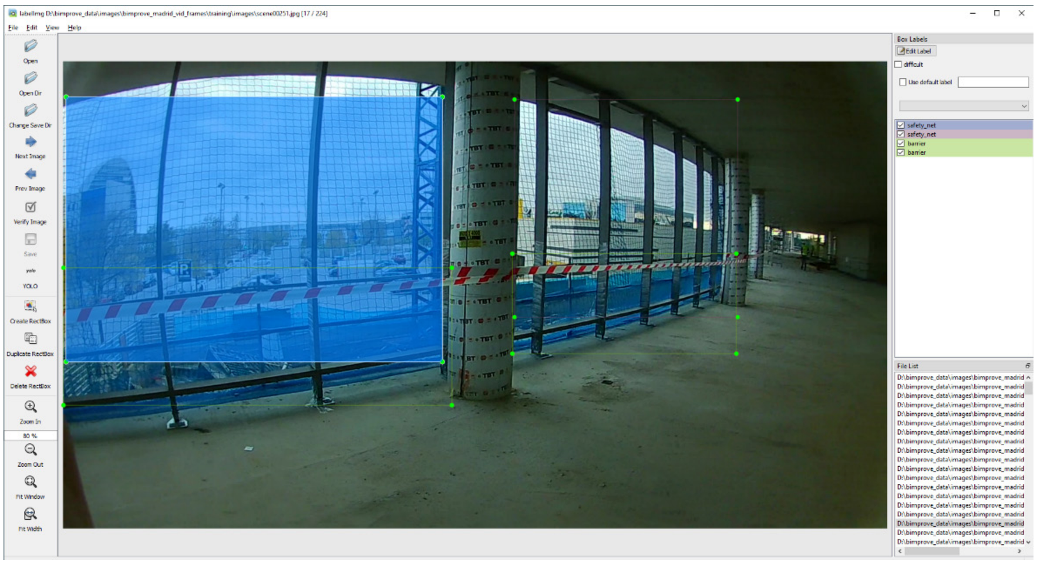

Figure 1: LabelImg tool and VTT sourced image with safety net and barrier labelling.

With the labelled and ready dataset we were able to start training the object detection model. To save time and effort transfer learning was used. This is done by freezing other layers in the model except the backbone responsible for the final object detection and classification and only to train and update those. This transfer learning method saves significant amounts of time and computing resources while producing reasonably accurate models.

The YOLO implementation (https://github.com/ultralytics/yolov5) used is built on the pyTorch (https://pytorch.org/) machine learning framework capable of accelerating the floating point and matrix operations with GPU. Installation and other requirements see the relevant documentation of pyTorch and yolov5. The YOLO implementation provided ready to use models out of which the base model chosen for this application was large type. This large model has over 4 million trainable parameters already trained with the COCO dataset. i.e. the model is already capable of detecting and locating generic objects in the images present in the COCO dataset, which unfortunately lacks the safety related objects needed here. The transfer learning process will shift the COCO detection capabilities to detect objects in our own BIMprove dataset. The training of the YOLO v5 based detection model was started with the following kind of command and parameters:

```
#python train.py --weights yolov5l.pt --data <path_to_data> --hyp
data\hyps\hyp.finetune.yaml --epochs 400 --imgsz 960 --workers 3 --freeze 10 --name
<name_of_session> --batch-size -1 --optimize AdamW --hyp data\hyps\hyp.finetune.yaml
```

This command starts the training session with following parameters:

- Model size – large type with 46642120 parameters in total
- Training epochs – 400
- Input image size – 960 x 960 pixels (optionally 640 for 640x640 pixels)
- Freezing of all layers below layer 10 trained on COCO dataset
- Automated batch size search – settles to 10 on the utilized computer
- Optimizer is AdamW
- Hyper parameters – as provided based on optimization with VOC dataset, initial learning rate is 0.0032

The utilized hyperparameter set activated automated and randomized artificial augmentation of the training dataset. The general intention is to increase the dataset by creating synthetically manipulated copies the original data and thus also improve generalization capabilities of the final model. The hyperparameters activated augmentations with probabilities of 0.898 for scaling, 0.602 for shearing, 0.5 and 0.00856 horizontal and vertical flipping, respectively. Included also was the mosaic effect, which composes the each of the training images from a set of random augmented image samples or even empty spaces. See Figure 2 below for some augmentation examples from our dataset (sample images from VTT and ZHAW) produced by the YOLO implementation and hyperparameters provided.

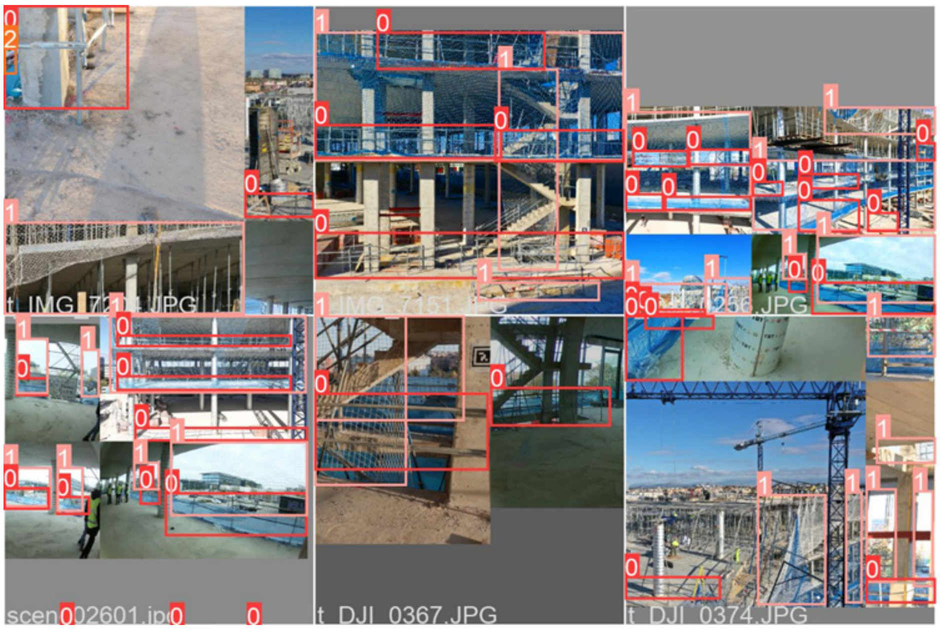

Figure 2: Tiled image samples (3 x 2 tiling) of the augmented data with artificial scaling, shearing, flipping and mosaic of the original dataset images.

The final model's detection capabilities are provided as a service through a suitable REST interface. One implementation available at https://github.com/superdupercodez/bimprove_rdc/tree/main/bimprove_rest, in *bimprove_restapi.py* file, loads the model and uses it to analyse the images send to it by HTTP POST. The detection results are returned as JSON text and described in the deliverable D2.4. Furthermore, additional guidance is provided for unintentional access to the */v1/risk_objects/* URL with a browser. Some YOLOv5 trained model files are provided in the git repository for testing purposes. Select the needed model by setting the *model_file_name* variable name as required in the *bimprove_rest/bimprove_restapi.py* file. Provided GPU acceleration is available change the model loading target to 'gpu'. See Torch documentation for the required GPU setup procedures.

```
#model = torch.hub.load("ultralytics/yolov5", 'custom', path=model_file_name).to('gpu')

model = torch.hub.load("ultralytics/yolov5", 'custom', path=model_file_name).to('cpu')
```

A simple request to the service can be made with a sample Python 3 client based on the *requests* library as

```
"""Perform test request"""

import pprint

import requests
```

```
DETECTION_URL = http://<SERVICE_URL>/v1/risk_objects/

TEST_IMAGE = "20.jpg"

image_data = open(TEST_IMAGE, "rb").read()

response = requests.post(DETECTION_URL, files={"image": image_data}).json()

pprint.pprint(response)
```

Where <SERVICE_URL> is the address URL of the provided service and TEST_IMAGE a file name of the sample request image. This sample is also available in https://github.com/superdupercodez/bimprove_rdc/blob/8e3f1065440e1a8b6790af2cf2ac6afb727f1396/bimprove_rest/bimprove_example_request.py. With the sample image the response was:

```
[{'class': 1,

  'confidence': 0.414974153,

  'name': 'safety_net',

  'xmax': 1620.0979003906,

  'xmin': 0.0,

  'ymax': 1082.6301269531,

  'ymin': 37.0090942383},

{'class': 1,

  'confidence': 0.3516817391,

  'name': 'safety_net',

  'xmax': 1724.8165283203,

  'xmin': 810.5069580078,

  'ymax': 610.0239257812,

  'ymin': 4.0267028809},

{'class': 1,

  'confidence': 0.263441056,

  'name': 'safety_net',
```

'xmax': 1870.6253662109,

'xmin': 507.2440185547,

'ymax': 1088.0,

'ymin': 525.5383300781}]

The returned ROVAS response contains the detection results encoded as JSON, where

Class – object class: barrier=0 | safety_net=1 | garbage=2

Confidence – detection confidence from 0-1.0

Name – human readable name for the detect object class
x1 – upper left corner x-coordinate of the bounding box
y1 – upper left corner y-coordinate of the bounding box
x2 – lower right corner x-coordinate of the bounding box
y2 – lower right corner y-coordinate of the bounding box

The coordinates are counted as pixels that of the original image dimensions.

All confidence results over 0.5 can be considered quite reliable detections but may vary significantly in practise as most of the available training data was quite homogeneous in image quality and scenery composition. The lower end confidence levels in this example ranging from 0.26 to 0.41 are due to the purposedly challenging sample image used. The sample image is an individual captured frame from a video with added challenges coming from the equipment used to record, lightning conditions and distance to the detected nets. See the image in https://github.com/superdupercodez/bimprove_rdc/blob/8e3f1065440e1a8b6790af2cf2ac6afb727f1396/bimprove_rest/madrid_test_image.jpg.

3. Risk and Image Data Management Service

The purpose of the BIMprove Risk and Image Data Management Service is to store and provide access to the data created by the Safety Related Visual Object Detection described above. The Risk and Image Data Management Service includes three main parts that are:

1. The database used to store the metadata provided by the Safety Related Visual Object Detection. The selected database solution is Apache Fuseki graph database that supports RDF (Resource Description Framework) format and SPARQL query language. The database structure follows the risk data ontology specifically developed to address the requirements of risk data management. A visualization of the ontology is shown in Figure 3.

2. REST interface that provides an access to the database. The interface includes two basic access methods that are insert data and query data. The interface is implemented using Python language and Tornado web framework with its asynchronous networking library.
3. HTML based GUI that enables examining the content of the database (as well as the results of the Safety Related Visual Object Detection). The GUI is supported by the Flask micro web framework written in Python.

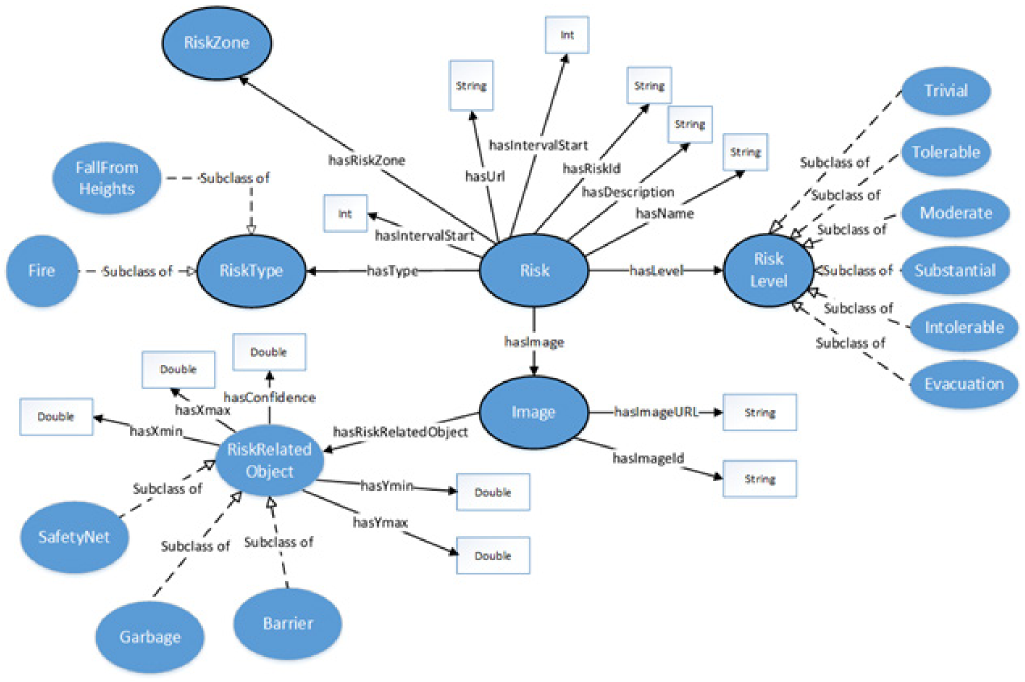

Figure 3: Risk ontology

The source code implementing the aforementioned modules is available at https://github.com/superdupercodez/bimprove_rdc. In the following sections the installation and usage of the Risk and Image Data Management Service is explained.

3.1. Installation of the Risk and Image Data Management Service

As a prerequisite you should install Python3 into your deployment machine.

Fuseki database and the interface implementing the provided functionalities can be installed with the following instructions:

1. **Install and start Fuseki**

a) Download Fuseki database deployment files from https://jena.apache.org/documentation/fuseki2/

b) Navigate to the folder you saved the files and start Fuseki with the following command:

‘command ./fuseki-server --mem /ds’

c) Navigate to http://localhost:3030/ (or whatever URL your Fuseki instance is deployed on) and create a new Dataset called "constructionSiteRiskData"

2. Install and start Risk and Image Data Management Service

a) Download the Github repository https://github.com/superdupercodez/bimprove_rdc, unzip the files and navigate to folder ImageMetadataStorage/DatabaseInterface

b) Execute command:

python3 fusekiProxy.py

c) Open folder ImageMetadataStorage/html and execute command:

python3 app.py

d) Open your web browser and navigate to URL http://localhost:8081/ (in case your are running the code in your local machine). If you see an empty table with the title "Safety factor and risk detection results" you have installed the service successfully.

3.2. Testing the installation and provided functionalities

To verify the interfaces of the service, you can add test data to the database. The interfaces can be accessed using e.g. CURL tool. The following scripts can be used (please note that the examples are valid for localhost installation):

1. Add data

The add data functionality enables simulating how results generated by the Safety Related Visual Object Detection service are inserted into the database.

```
curl -iX POST \

'http://localhost:8084/fusekiAddImage' \

-H 'Content-Type: text/plain' \

-d '{"imageID": "2MppzXLWeT", "name":"Safetynet", "confidence": "0.44544", "xmax":
"675.44312", "xmin": "345.86625", "ymax": "876.31249", "ymin": "436.5287",
"imageURL": "https://bit.ly/3glwtiH", "anchorBoxImageURL":
"https://bit.ly/3glwtiH_2"}'
```

As can be seen, many of the elements of the input JSON are similar to the data elements of the ROVAS response. In addition, the input data contains a unique id of the image and two URLs. The first URL leads to a web location where the analysed image is stored. The second image URL points to another version of the same image where the bounding boxes of labelled items (safety net, barrier or trash) are shown. You can verify the data insert operation by navigating to http://localhost:8081/, the table should now contain one data instance.

2. Free query

The free query enables executing SPARQL queries against the database.

```
curl -iX POST \

'http://localhost:8084/fusekiQueryData' \

-H 'Content-Type: text/plain' \

-d 'SELECT ?imageId  WHERE { ?image  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.semanticweb.org/safetyOntology#Image>. ?image <http://www.semanticweb.org/safetyOntology#hasRiskRelatedObject> "Safetynet".?image <http://www.semanticweb.org/safetyOntology#hasImageId> ?imageId}'
```

The shown SPARQL query statement retrieves the IDs of all database objects that a) are instances of the class ‘Image’ and b) have “Safetynet” as detected safety related object.

4. References

[1] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., “You Only Look Once: Unified, Real-Time Object Detection”, http://arxiv.org/abs/1506.02640, 2015.