DEVELOPMENT OF AUTONOMOUS SYSTEMS FOR INTELLIGENCE-GATHERING MISSIONS

Neo Hao Jun[1], Ng Jia Wei[2], Tan Yu Yao[1], Eu Jun Liang Benjamin[3], Ng Woon Ting Elizabeth[3], Jeremy Tian Wen Loong[3]

[1] Hwa Chong Institution (High School), 661 Bukit Timah Road, Singapore 269734
[2] Nanyang Girls' High School, 2 Linden Drive, Singapore 288683
[3] Defence Science and Technology Agency, 1 Depot Road, Singapore 109679

ABSTRACT

Given the myriad of advantages Unmanned Ground Vehicles (UGVs) offer in military land operations and combat, this project aims to develop a solution that allows UGVs to assist in intelligence-gathering in unknown or unsafe territories. Basic mapping of the area is done through a Simultaneous Localisation and Mapping algorithm on the UGV, and the UGV's video feed is then streamed to a central receiver where it is passed through a computer vision model. Through our tests, we have shortlisted eight optimal video transcoding options that achieve a 99% performance on our model with minimal bandwidth usage when streaming the UGV's video feed. Our trained model also managed to achieve a mean Average Precision (mAP) of 0.437 when considering an Intersection Over Union (IOU) of 50% to 95% overlap , with a majority of the detections having a confidence of 0.4 or higher.

TABLE OF CONTENTS

INTRODUCTION

With the advent of a new technological age over the past few decades, Unmanned Ground Vehicles (UGVs) have become a centrepiece in many military and civilian applications around the world and have already been playing a critical role in military land operations and combat. Controlled by an onboard controller, either through independent Artificial Intelligence (AI) or remote control by human operators, UGVs offer many advantages especially in areas that pose a risk to human safety, such as contaminated radioactive sites, areas under terrorist attack, or new areas that have yet to be explored extensively. Under such circumstances, UGVs' autonomy ensures that intelligence-gathering can proceed without interruption.

Figure 1. *An example of a UGV conducting intelligence-gathering in combat.*
Figure 2. *An example of a CV model conducting unattended baggage detection.*

This project thus aims to develop a solution that allows a UGV to navigate and map a terrain via a Simultaneous Localisation and Mapping (SLAM) algorithm, while gathering information that can be broadcasted as a live stream back to a human operator, negating the need for a physical human to be present on site. This function will be enhanced through the use of computer vision (CV) to detect specific items specified by the human operator, as seen in Figure 2, reducing the potential for errors stemming from operator inattention or fatigue, and also allowing the system to act as a "force multiplier", for one human operator can now easily watch over the data streams of multiple UGVs simultaneously.

In addition, since the UGV is mobile, it must rely on a mobile network to broadcast its live stream. Yet, network bandwidth and stability is oftentimes unpredictable, which will greatly affect the quality and speed of the stream, a negative effect that trickles down to the performance of the CV model. With limited bandwidth available, this project seeks to optimise the video transcoding so as to allow a scalable, consistent, and low-latency video stream from the UGV that does not impede object detection severely.

MATERIALS AND METHODS

Hardware Selection

Based on size, the UGV market is divided into small (4.5 to 90kg), medium (100 to 250 kg), large (250 to 500kg), very large (500 to 1000kg), and extremely large (> 1000 kg). Small UGVs (SUGVs) would be most suited for the aforementioned application, for their small size allows

them to access tight spaces that their bigger counterparts are unable to, producing a more representative and comprehensive map of the site. Furthermore, due to the comparatively low cost of such vehicles, more SUGVs can be sent to a specific location to gather information simultaneously, an increase in coverage that would be especially useful in data-gathering applications.

Such SUGVs are abundant in consumer robotics, with Educational Robot Kits serving as one such example. Equipped with four mecanum wheels capable of delivering 19W of power each, the Educational Robot Kit, even with its small size, is able to carry a relatively large payload while manoeuvring omnidirectionally around tight spaces. This makes it suitable for the aforementioned application, which would involve the attachment of multiple payloads to aid the SLAM and data-gathering workflows.

A commercial off-the-shelf (COTS) product is chosen because at such a small scale, the performance benefit of opting for a military off-the-shelf (MOTS) or bespoke system is negligible despite the substantial increase in cost. Besides, this project is intended to be platform-agonistic, and the Educational Robot Kit only serves as a proof of concept (PoC) that can be easily repackaged for alternative SUGV options, or even expanded into larger, more robust systems and applications.

Hardware Control

Typically, the Educational Robot Kit is controlled via an application, which, due to its simplicity, poses a challenge when unsupported sensors and peripherals have to be used or when more complex algorithms have to be implemented, both of which would be the case in this project.

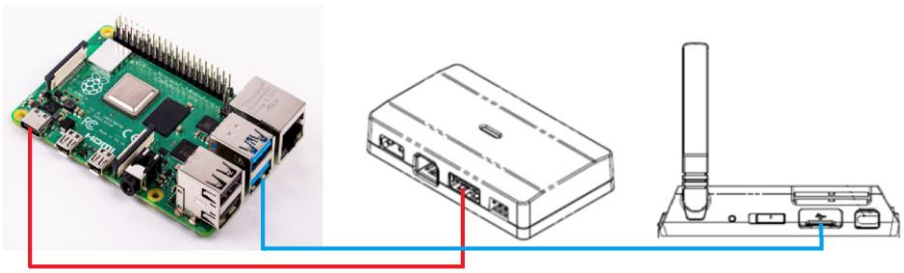

Figure 3. *Connecting the Raspberry Pi to the Robot Kit via the USB connection mode.*

As such, we opted to interface with the robot via the Raspberry Pi using a compatible software development kit (SDK), extending the capabilities of the robot kit through connection with external peripherals such as the Intel® RealSense™ D435i and T265 for the SLAM and CV algorithms.

Simultaneous Localisation and Mapping (SLAM)

Figure 4. *Intel® RealSense™ Depth Camera D435i (left) and Tracking Camera T265 (right).*

The Intel® RealSense™ Tracking Camera T265 and Depth Camera D435i, pictured above in Figure 4, are used in conjunction with each other to provide the SUGV with a spatial understanding of its environment and its own position and orientation in that space. The Intel® RealSense™ Tracking Camera T265 estimates the SUGV's position and orientation relative to a gravity-aligned static reference frame, while the Intel® RealSense™ Depth Camera D435i performs stereo matching to obtain a dense cloud of 3D scene points. Together this input can be used to obtain a point cloud that is registered with respect to a gravity-aligned static reference frame. The full algorithmic stack has been detailed below in Figure 5 and the proceeding paragraphs.

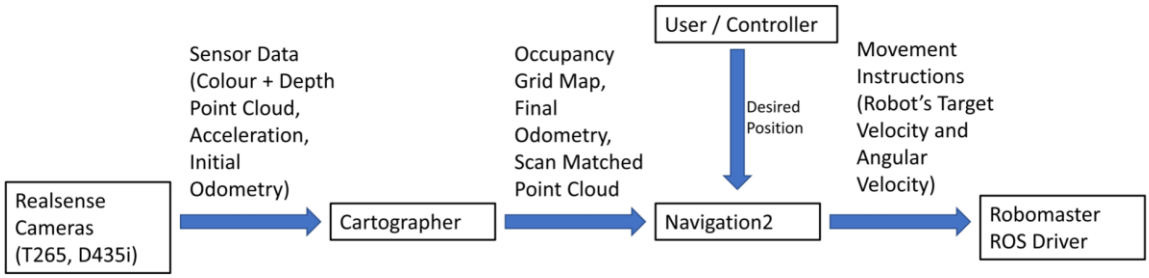

Figure 5. *Conceptual overview of SLAM and navigation algorithmic stack, powered by ROS2.*

The Robot Operating System 2 (ROS2) ROS2 wrapper for Intel® RealSense™ devices provides ROS2 nodes for using the aforementioned Intel® RealSense™ cameras, meaning that data from these devices can be easily obtained and published onto a ROS2 network. By doing so, data from these cameras can be passed through the ROS-enabled Google Cartographer, a system that supports real-time SLAM with the Intel® RealSense™ cameras.

The SLAM data is then used to create a map that is used for robot navigation via Nav2, a ROS navigation stack that allows for robot wayfinding with obstacle avoidance and dynamic path planning. This navigation information will then be piped back to the robot via a ROS2 message received by the Raspberry Pi. The feed from these cameras also double up as the video stream that will be transcoded and passed through the object-detection model, as detailed in the following sections.

Transcoding UGV Stream

The video streams from the UGV are broadcasted to a transcoder through the real-time streaming protocol (RTSP). The transcoding engine used was Wowza Streaming Engine due

to its accessible interface and ability to take input and output streams through many different protocols such as RTSP. The RTSP protocol is used due to its wide usage in Internet Protocol (IP) cameras which allows for easy scalability.

To identify which transcoding configuration is optimal, we adjusted the video stream's bitrate for each resolution (160p, 240p, 360p, 720p) and each frame rate setting (10 fps, 30 fps). We then rated the performance of each output stream after being annotated with bounding boxes from the CV model through the following criteria: latency, the approximate performance of the model and average bandwidth usage. The optimal transcoding configuration at each resolution and frame rate setting was the one that resulted in an approximate performance of 99% from the CV model at the lowest bitrate possible. The CV model's performance is assessed based on the stability of its detection of unattended bags: if an unattended bag is detected in 99% of consecutive test frames streamed from the UGV, the model is deemed to have an approximate performance of 99%.

We then tested 2 different combinations of streams on a 2x2 video wall to see how bitrate is changed when all 4 screens are displayed at the same time, as compared to when only 1 screen is displayed. All streams are taken with a stationary camera and no movement in the video captured.

Training and Assessing the CV Model

YOLOv5 is a modern object detection model known for its fast speed, high accuracy, ease of installation and use. It is pre-trained on the Common Objects in Context (COCO) database, a large-scale, challenging, and high-quality object detection, segmentation, and captioning dataset.

Our CV model builds upon the YOLOv5, ingesting the transcoded RTSP stream to obtain raw frames that can be used for inference. For this project, the CV model is trained to detect unattended bags in an unknown territory, though the model can be easily re-trained for alternative use cases when the need arises.

Specifically, using a basic unattended bag framework, the model compares the pixel distance between identified humans and bags, alerting the user if it picks out a bag that is a distance away from the surrounding individuals. When an alert needs to be sent out, the CV model annotates the bounding boxes of unattended bags onto the original stream. To allow the operator to verify the detection made by the model, bounding boxes of humans and other bags are also drawn, but are not as conspicuous. These annotated frames are then rebroadcasted to RTSP through FFmpeg for ease of further processing.

As an extension to the unattended bag detection model, we have also trained the YOLOv5 model on a custom rotary-winged UAV dataset made by Mehdi Özel. The training produced a fairly consistent and robust UAV detection model and was assessed using a self-labelled test dataset made by picking out 50 frames from online footage of drones.

Video Wall

With multiple resulting output streams from the CV model, it is important to display the consolidated streams neatly on a video wall for observation. We opted for the Milestone XProtect Smart Wall, which enables us to manually tweak the settings for the video streams depending on the context and available bandwidth at that moment.

RESULTS AND DISCUSSION

Extension of Hardware Control of the Educational Robot Kit

While present workarounds do allow for customised control of the Educational Robot Kits, such interfaces still fail to provide the low-level control that would be useful in fully modifying the Robot for specialised missions. To resolve this, the CAN BUS port on the Robot, as depicted below in Figure 6, can be intercepted to send signals to individual components on the Robot rather than running our code through the Original Equipment Manufacturers (OEM) frameworks.

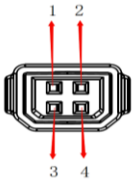

Figure 6. *CAN BUS port on the Robot Kit (1: GND, 2: +12V, 3: CANH, 4: CANL).*

We have managed to receive, interpret, and send individual commands through the Robot kit's CAN network using the Raspberry Pi 4B and Waveshare's RS485 CAN HAT. Our findings on the various IDs we managed to map out the CAN Network as detailed below in Table X.

Table 1. *CAN IDs of various Robot Kit components.*

CAN ID	Description	CAN ID	Description	CAN ID	Description
0x200	Self-assigned	0x204	Blaster	0x214	Chassis left armour
0x201	Intelligent controller	0x211	Chassis rear armour	0x215	Gimbal right armour
0x202	Motion controller	0x212	Chassis front armour	0x216	Gimbal left armour
0x203	Gimbal	0x213	Chassis right armour	0x221	Infrared distance sensor

Although we found that the CAN network on the Robot could indeed be accessed, a working implementation would involve reproducing the OEM's proprietary communication protocols on the software side to control the Robot without the OEM's low-level controller. Due to the short-term nature of this project, the discoveries we have made on this front remain a PoC that should be expanded upon in future work.

Outcome of SLAM Algorithm

Based on our tests, our SLAM algorithm works as expected, with the Google Cartographer for ROS generating an output map similar to the one shown below in Figure 7. Using the map generated by the SLAM stack, our navigation system enables the Robot to be successfully controlled by a remote operator through inputting desired waypoints, enabling semi-autonomous data gathering to be conducted in areas of interest.

Figure 7. *Example of a map generated by the Google Cartographer for ROS.*

Analysis of Transcoding Settings

We tested different resolution settings and decided on 160p, 240p, 360p and 720p as the most suitable. Too high of a resolution is unnecessary and requires too much bandwidth, while too low of a resolution will result in the CV model being unable to detect unattended bags.

Through our tests, we have found that the following optimal transcoding configurations that achieve an approximate CV model performance of 99%, as noted in Table 2. A full list of results can be found in the Annex.

Table 2. *Optimal configurations for various resolution and fps settings.*

Configuration				**Performance**		
Resolution	**Codec**	**Frame rate / fps**	**Bitrate / kbps**	**Latency / s**	**CV performance**	**Bandwidth Usage / kbps**
720p	H264	30 (source)	200	2.5	~ 99%	~ 300
720p		10	120	4		~ 100
360p		30 (source)	80	2.5		~ 100
360p		10	50	4.5		~ 60
240p		30 (source)	50	2.5		~ 60
240p		10	50	3		~ 70
160p		30 (source)	50	2		~ 60
160p		10	30	3.5		~ 40

As seen, the latency and quality must be balanced with the scale of streaming as these two factors are inversely proportional to each other, so as to accommodate the limited bandwidth available. In addition, a higher quality allows for more accurate detection by CV but results in much greater bandwidth usage, buffering, and latency, making it more difficult to scale up the number of streams. Conversely, a lower-quality video stream will have lower latency and be easier to scale up but make it difficult for the CV model or the human operator to detect objects of interest. The same trend is seen when the number of streams is increased, as shown by Table 3 below.

Table 3. *Combinations of streams on a 2x2 video wall.*

Latency / s	**Bitrate / bps**	**Latency / s**

160p 30 fps	160p 10 fps	240p 30 fps	240p 10 fps	3 (30 fps), 4 (10 fps)	1M (4 screens)
720p 30 fps	720p 30 fps	360p 30 fps	360p 30 fps	3 (30 fps), 5.5 (10 fps)	2.5M (4 screens)

While the aforementioned configurations are all optimal, the ideal configuration to be chosen still depends on its application. For example, when there are more than 4 screens to be shown at the video wall at a time, the resolution does not need to be as high, so using 240p or lower is recommended, so as to minimise bandwidth consumption. When there is not much movement captured, using the 10 fps settings is recommended since the bandwidth consumption is lowered by a great amount, in exchange for slightly greater latency.

Performance of CV Model

Through the tests we conducted on our CV model to detect UAVs, we have obtained the following data validation results, as depicted in the figures below.

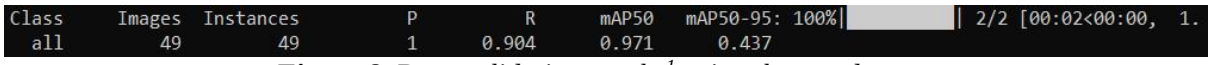

Figure 8. *Data validation results[1] using the test dataset.*

The mean Average Precision (mAP) is a standard metric used to analyse the accuracy of an object-detection model, defined as the mean of the average precision obtained every time a new positive sample is recalled. As illustrated by Figure 8 above, the CV model has demonstrated a mAP of 0.437 when considering an Intersection Over Union (IOU) of 50% to 95% overlap, a performance that is substantial for a model trained on fewer than 2,000 labelled images.

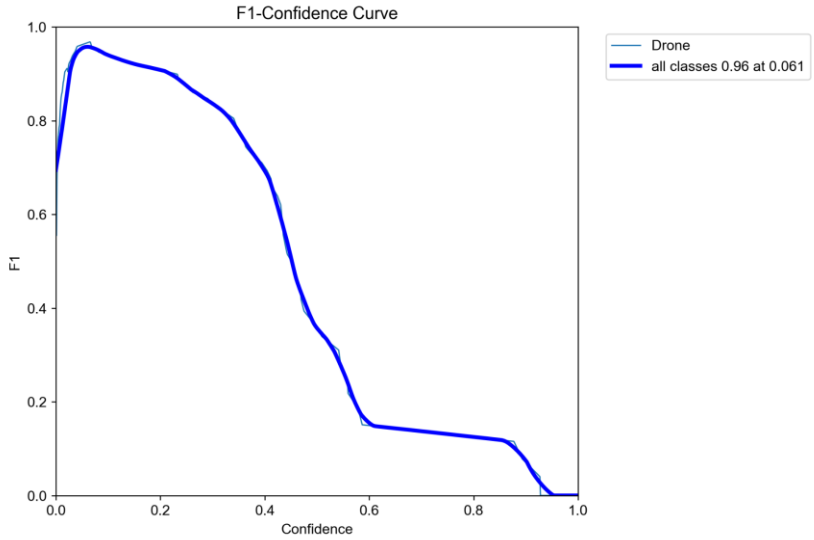

Figure 9. *Graph of F1 score against Confidence.*

The F1 score is calculated as the ratio of correctly predicted positive examples divided by the total number of positive examples that were predicted and conveys the balance between the precision and the recall. It is a popular performance metric for classification systems and is generally more useful than accuracy, especially in cases with an uneven class distribution.

[1] Precision (P) represents the proportion of Resultant Detections meeting the real truth (i.e. the proportion of bounding boxes being also part of the given labels). Recall(R) represents the proportion of the labels that actually were reflected as bounding boxes in the test runs.

The F1 score of the model at a specific confidence level denotes how consistent the model is at detecting the target object at that confidence level. As shown by the graph in Figure 9, the biggest drop in the F1 score occurs at about a confidence of 0.4, meaning that a majority of the detections have a confidence of 0.4 or higher.

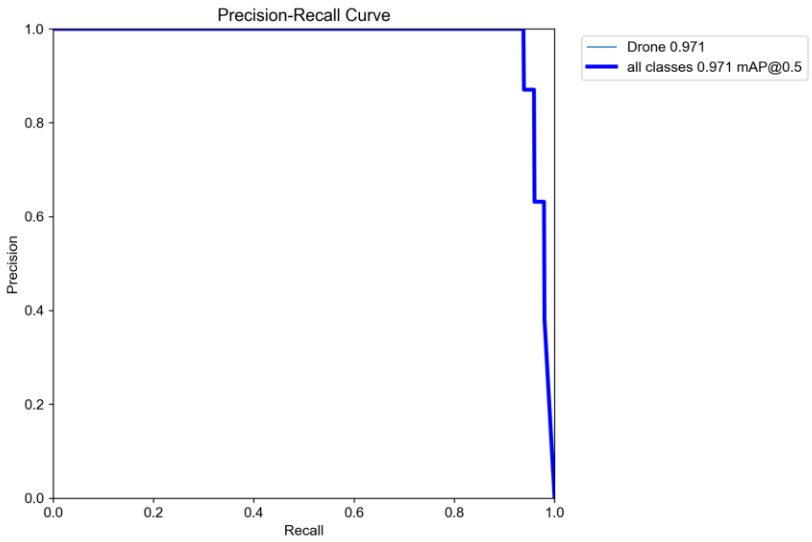

Figure 10. *Precision-Recall Curve.*

The precision-recall curve shows the trade-off between precision and recall for different thresholds. A large area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate.

As seen in Figure 10, our model demonstrates rather high levels of recall and precision, with the precision only dipping to 0.6 at a recall of 0.95, meaning that in the best 95% of the dataset, the model has a precision of at least 60%. This is partly because the model was occasionally unable to detect the drone due to its small size and indistinguishable features, as seen in Figure 11 below.

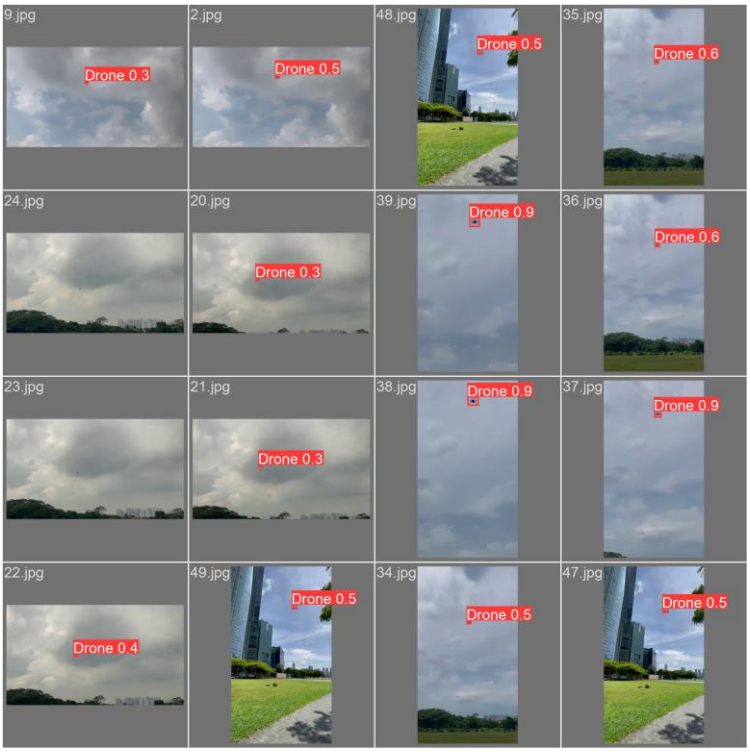

Figure 11. *Examples of UAV detection by the model.*

CONCLUSION

Recommendations for transcoder configurations

The stream configurations to be used depend on the operational conditions, and the different transcoding options allow the human operator to easily modify the stream settings to optimise bandwidth usage depending on the scenario. In essence, resolution can be decreased if many video feeds are being concurrently observed, as the numerous streams will be compacted into the same screen with a fixed resolution, while frame rate can be sacrificed if the frame is mostly static or the objects are moving relatively slowly, in which a slight delay in reaction time is insignificant.

Ultimately, the available bandwidth is still the most important factor in the selection of a stream configuration. If there is insufficient bandwidth, some parts of the stream quality have to be sacrificed in order to ensure a continuous and smooth stream even under such circumstances.

Real-Life Applicability of CV Model

Although the UAV dataset had only about 1,500 images, it was able to generate a fairly robust CV model. With more training images and precise labels, the CV model can be made more consistent and reliable.

In addition, as the video streams are all piped back to a central receiver rather than being analysed remotely on the various UGVs, this model can be continuously updated or substituted with better alternatives and be supplemented by more power-hungry resources that can help the model detect targets more effectively by conducting more extensive processing on each frame received by the receiver.

LIMITATIONS OF STUDY

Due to the limitations of the COCO database, not all types of bags are able to be detected by the CV model, and the detection is only accurate when the bag is upright and facing the camera. This is infeasible in cases where the UGV is not able to adjust the angle of the camera and thus the unattended bag may be missed out.

Future work could look at using 3D space identification to track objects blocked by obstacles by using another camera that is able to track the relative x, y and z coordinates of that object. This further enhances the tracking ability of the UGV, even in areas where the camera's field of view is partially obscured.

ACKNOWLEDGEMENTS

We would like to thank and express our deepest gratitude to our mentors, Mr Benjamin Eu, Ms Elizabeth Ng and Mr Jeremy Tian for their invaluable support and guidance for the entire duration of this project. Without their mentorship, this paper would not have come to fruition.

In addition, we would like to thank Ms Chua Hui Ru for her assistance in the administrative portion of this project, as well as DSTA for providing us with the opportunity to participate in Research@YDSP 2022.

REFERENCES

Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, Kalen Michael, TaoXie, Jiacong Fang, imyhxy, Lorna, 曾逸夫(Zeng Yifu), Colin Wong, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, … Mrinal Jain. (2022). ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation (v7.0). Zenodo. https://doi.org/10.5281/zenodo.7347926

Grunnet-Jepsen, A., Harville, M., Fulkerson, B., Piro, D., Brook, S. & Radford, J. (n.d.). *An Introduction to Intel® RealSenseTM Visual SLAM and the T265 Tracking Camera* (Version 1.0). Intel Corporation. https://www.intelrealsense.com/download/9275/?-1818208019.1672666677.

Intel Corporation. (2018, 16 April). *Unattended Baggage Detection Using Deep Neural Networks in Intel® Architecture*. https://www.intel.com/content/www/us/en/developer/articles/technical/unattended-baggage-detection-using-deep-neural-networks-in-intel-architecture.html

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

Lindholm, V. (2022). Unmanned Ground Vehicles in Urban Military Operations: A case study exploring what the potential end users want.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Schmidt, P. (2019) *Intel® RealSenseTM Tracking Camera T265 and Intel® RealSenseTM Depth Camera D435 - Tracking and Depth* (Revision 001). Edited by James Scaife Jr., Michael Harville, Slavik Liman, Adam Ahmed, Intel Corporation. https://www.intelrealsense.com/wp-content/uploads/2019/11/Intel_RealSense_Tracking_and_Depth_Whitepaper_rev001.pdf?_ga=2.257078728.299532539.1672835296-1818208019.1672666677

ANNEX

Table 4. *Test results for various transcoding configurations.*

Configurations				Performance			
Resolution	Codec	Frame rate / fps	Bitrate / kbps	Latency / s	CV	Bandwidth Usage / kbps	Remarks
720p	H264	30 (source)	3,000 (default)	2	~ 100%	~ 3,000	
720p		30 (source)	1,000	2.5	~ 99%	~ 1,000	
720p		30 (source)	300	2.5	~ 99%	~ 500	
720p		30 (source)	30	2.5	< 70%	~ 100	Stream is not smooth (some frames skipped), false detections
720p		30 (source)	80	2.5	< 80%	~ 200	Artefacts start appearing, sometimes doesn't identify correctly
720p		30 (source)	120	2.5	~ 90%	~ 250	Image is rather blurry
720p		30 (source)	200	2.5	~ 99%	~ 300	
720p		10	120	4	~ 99%	~ 100	
720p		10	60	4.5	~ 70%	~ 100	A lot of artefacts on screen, sometimes doesn't detect bag

720p		30 (source)	200	2.5	~ 99%	~ 300	Uses more GPU
360p		30 (source)	36 (default)	2.5	100%	~ 400	
360p		30 (source)	150	2.5	~ 99%	~ 220	
360p		30 (source)	50	2.5	< 90%	~ 60	
360p		30 (source)	80	2.5	~ 99%	~ 100	
360p		30 (source)	30	2.5	< 50%	~ 80	
360p		30 (source)	80	2.5	~ 99%	~ 100	
360p		10	80	4.5	~ 99%	~ 80	
360p		10	50	4.5	~ 99%	~ 60	
360p		10	30	4.5	< 90%	~ 40	
240p		30 (source)	145 (default)	2.5	100%	~ 150	
240p		30 (source)	50	2.5	~ 99%	~ 60	
240p		30 (source)	10	2.5	< 50%	~ 30	False detections, very blurry
240p		30 (source)	30	2.5	< 90%	~ 40	
240p		30 (source)	40	2.5	< 90%	~ 50	Flickering, sometimes doesn’t detect
240p		30 (source)	50	2.5	~ 99%	~ 60	
240p		10	50	3	~ 99%	~ 70	

240p		10	30	3	< 80%	~ 50	
160p		30 (source)	105 (default)	2	~ 99%	~ 120	
160p		30 (source)	40	2	~ 50%	~ 50	
160p		30 (source)	70	2	~ 99%	~ 90	
160p		30 (source)	50	2	~ 99%	~ 60	
160p		30 (source)	50	2	< 90%	~ 80	Stops detecting bag sometimes randomly
160p		10	50	3.5	~ 99%	~ 80	
160p		10	30	3.5	~ 99%	~ 40	
160p		10	20	3.5	< 70%	~ 40	
160x90		30 (source)	50	-	0%	-	-