Fusion-Based Utilization and Synthesis of Efficient Detections (Project F.U.S.E.D.)

Thomas James[1], Parker Liberatore[2], Ethan Rogers[3]
Mississippi State University, Mississippi State, MS, 39762, United States

This study aimed to develop hardware and software for an object detection fusion system, using three different sensors. The system's performance was compared to that of individual sensors deployed for the same task. The focus of the research was to prove the competence and benefits of a decision-level fusion method as it was applied to lightweight object detection architectures, and the driving motivators behind the study were simplicity in implementation and good computational performance. In short, the algorithm would use lightweight models to perform object detection on the sensors individually, and then the detection coordinates would be transformed into the same coordinate planes for correlation. The method exploited sensor calibration and depth images to create an efficient workflow. Once the algorithm was created, it was tested and compared both qualitatively and quantitatively to individual models deployed on the same sensors with no fusion framework. Qualitative results clearly indicated that the fusion algorithm outperformed individual models in more challenging scenarios. The quantitative results indicated the same trend, but it was also clear that inaccuracies in the fusion methodology resulted in a small percentage of the true detections being missed when they were otherwise caught by individual models. Future work should consider investigating the deployment of the fusion algorithm on small devices, because the lightweight models were intended for mobile deployments. Other possible work should study the effect of improved extrinsic calibration, better-trained models, semantic segmentation models overlayed for improved depth resolution, and adding adaptability to the algorithm's decision-making process for different scenarios.

I. Nomenclature

K	=	intrinsic camera parameter matrix
$\vec{r}$	=	position vector
R	=	rotation matrix
T	=	translation vector
u	=	horizontal coordinate in a sensor's pixel coordinate system
v	=	vertical coordinate in a sensor's pixel coordinate system
z	=	depth coordinate in a sensor's pixel coordinate system

II. Introduction

Sensor systems are essential for a variety of aerospace applications. Vehicles such as drones, airplanes, and satellites are outfitted with arrays of sensors that collect data, and one of the most common sensor tasks is perceiving the surrounding environment. Commonly used sensors for this task include cameras, radars, and LiDAR sensors (Light Detection and Ranging). Advancements in AI and machine learning have made the use of object detection methods in camera systems conventional. Object detection is a subset of machine learning in which models are trained to detect, identify, and locate objects of interest in images. The models place bounding boxes around the objects and label them. Such models would prove useful when deployed on cameras.

[1] Undergraduate Student, Department of Aerospace Engineering, Student Member.
[2] Undergraduate Student, Department of Aerospace Engineering, Student Member.
[3] Undergraduate Student, Department of Aerospace Engineering, Student Member.

Consider the following scenario: if an autonomous drone was deployed in a crisis response situation to search for people, would a single camera provide enough information and confidence if an object detection model indicated that a person was spotted? The answer, in many situations, is no. There would be several scenarios in which a simple webcam (RGB camera) would be useless. If it was dark, or foggy, or if the drone was in an urban environment with people projected on screens and mannequins standing in windows, the object detection model deployed on the webcam would either miss detections of people or mistake an object for a real human.

Therefore, it becomes apparent that a combination of sensors with different strengths would be required to complete this task in more challenging scenarios. Making use of data from different modalities is an active area of research in several disciplines, termed data fusion. The research detailed in this paper aimed to develop a sensor system, both by assembling hardware and software, that would deploy lightweight object detection models in a decision-level fusion framework to address the person-detection scenario explained previously. Three sensors were used: a LiDAR sensor, a thermal camera, and a webcam. Prior to explaining the method of fusion and how this research expands upon the body of research, a short survey of relevant work will be presented next.

Several investigations have recently been conducted regarding the fusion of LiDAR sensors and cameras for object detection applications. One study used a 2-D LiDAR sensor with 360° range and a webcam [1]. Object detections from the webcam were fused with distance/angle data from the LiDAR sensor. The authors of the study recommended using geometric projections and transformations to make the fusion more accurate. Another study performed extrinsic calibration between a thermal camera and a LiDAR sensor so that points from the LiDAR's 3-D point cloud could be projected onto the thermal images, adding a second layer of useful data to the images [2]. Some work has also considered fusing camera images and depth images from LiDAR sensors together by using convolution layers in a neural network [3]. A more general survey of current research on LiDAR and camera fusion has been conducted, too [4]. Some of the discussed methods include depth completion, 3-D object detection, 2-D/3-D semantic segmentation, and 3-D object tracking. Another similar study used an altered version of Faster R-CNN to perform a non-maximum suppression algorithm on bounding box detections from thermal and depth images [5].

The method of fusion presented in this paper draws similar elements from the listed studies. The LiDAR sensor and both cameras were calibrated intrinsically and extrinsically to allow for projection of points between the sensors. Additionally, LiDAR depth images were used instead of point clouds. However, instead of overlaying the three images together or addressing a more complex task such as 3-D object detection or semantic segmentation, the sensor fusion method purely considered the transformation and projection of 2-D bounding box detections between the sensors. To reduce complexity even further, the only detections to be transformed would be from the LiDAR depth images, removing the need for stereo vision techniques.

In short, the core fusion methodology was as follows: given three images from the LiDAR sensor and the two cameras, object detection models trained on the individual sensors would perform detection individually. Next, the bounding boxes on the LiDAR depth image would be transformed into the webcam and thermal reference frames, separately. The boxes were transformed from the depth images by creating 3-D position vectors out of the bounding box corners. Depth information was taken at the centers of the boxes, hopefully aligning the boxes with the detections (people) in 3-D space. After that, a decision-level fusion process was used to correlate detections and decide which detections to keep and throw out.

This methodology was investigated primarily to assess how effective the framework would be at the given task. If effectiveness was proven when compared against individual sensor detections, then the benefit of the method would include its simplicity, ease of use, efficiency, and lightweight nature. The method of fusion avoided the complex areas of current research and instead focused on proving the utility of calibration and depth images for 2-D object detection. The object detection models that were trained and deployed used the MobileNet CNN architecture from the TensorFlow Lite library, further emphasizing the goal of developing a lightweight system. The intention was for such a system to be easily deployable by casual users on smaller systems, such as embedded computers on an autonomous drone. The combination of three sensing modalities, extremely lightweight models, and the bounding box correlation method is what distinguishes this study from the existing body of research. For more information about the project, please visit the project website [6].

The rest of the paper will be structured as follows: the development of the sensor system and performance assessment methods will be explained, observations and quantitative data will be presented, discussion will be provided on the results and possible future work, and conclusions will be drawn.

III. Methodology

This section will describe the physical setup of the sensor system, the sensor calibration process, data acquisition, model training, fusion workflow methodology, and performance assessment methodology.

A. Hardware Setup

For reference, the sensors used for the research were the following: a Logitech Brio 101 Webcam [7, 8], a pmd flexx2 LiDAR sensor [9-11], and the Teledyne FLIR Lepton Micro Thermal Imaging Core [12, 13]. They were attached to a custom fabricated mount where the sensors' relative positions could be kept approximately constant. A multi-USB port adapter was included on the back of the mount to provide a connection point for all sensors. A diagram of the sensor mount is given next in Fig. 1.

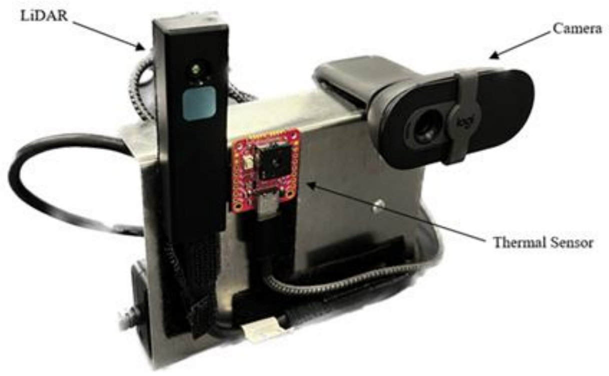

Fig. 1 Sensor Mount Diagram.

The mount also required the PureThermal 3 breakout board [14], used to house the thermal sensor and provide a USB connection to it. Prior to the calibration process, the mount was further secured so that the sensors would be unable to move relative to each other, and that configuration is shown in the section on data acquisition. With the sensor mount physically set up, the three sensors were connected through USB to a Windows 11 laptop.

For the virtual setup of the sensor array in a development environment, the three USB connections were bridged to the Windows Subsystem for Linux (WSL) with an Ubuntu installation on the laptop. The default WSL environment had settings disabled that blocked the passthrough of certain USB connections from the Windows environment to WSL, so a custom Linux kernel for WSL was built [15].

B. Software Setup

The sensors were passed into the WSL environment for two reasons. The first is because Robot Operating System 2 (ROS2), which works well in a Linux environment, was chosen as the framework for sensor deployment and processing [16, 17]. The second is that the software installation for the LiDAR sensor only worked in Linux.

Next, a containerized environment using Docker and Visual Studio Code was built to house the drivers for the three sensors and the ROS2 framework. The sensors were initialized in the environment using ROS2 GitHub repositories: one for USB cameras [18], another specifically for the LiDAR sensor [19], and one final repository for image processing [20]. This configuration, when linked to GitHub, would allow for efficient collaboration and sharing of work given that the appropriate Linux kernel was built and the LiDAR sensor software installed on a separate machine. The ROS2 environment was thus configured to handle image data streamed from the three sensors at the same time.

C. Calibration

With the sensors initialized in the virtual environment, two elements would be required to enable the detection fusion workflow: image synchronization and sensor calibration. Synchronization was required because the images from the three sensors would provide useless information for fusion if they were not capturing the same approximate moment in time. The ROS2 environment provided embedded tools for image synchronization, so this solution was simple. However, regarding the second element, all sensors in the mount needed to be calibrated to ensure accurate transformations of points between sensor coordinate systems. Although ROS2 provided tools for sensor calibration, the process was not trivial.

To provide some context, the primary method for the detection fusion workflow would be to take a data point (pixel) from the LiDAR image, convert that point to a 3-D position vector expressed in the LiDAR's reference frame, and then transform that vector into the pixel coordinate frames of the webcam and thermal sensor. These transformations would require two things: the intrinsic camera parameter matrices for all three individual sensors, and the extrinsic calibration matrices for translations and rotations between the sensors. Additionally, parameters for camera distortion would be required.

The intrinsic matrices are used for transforming between the pixel coordinate systems and the 3-D sensor frames on individual sensors. They require the focal length, horizontal and vertical pixel sizes, and the sensor's principal point. The principal point is defined as the point, expressed in units of pixels, in the center of the image. The pixel coordinate system is defined with the origin at the top left corner of the image, and the exact center of the image may not be exactly half the image resolution due to distortion. Thus, the principal point may not always be perfectly in the center of the given resolution.

The extrinsic matrices provide transformations between the different sensors. They contain a rotation matrix for aligning sensor orientations, and they also contain a translation vector that resolves the two frames together in 3-D space. Lastly, the distortion parameters are used to yield undistorted images from the raw images, and the undistorted version of the sensor images are the ones used for the intrinsic transformations.

ROS2 contains libraries with algorithms that converge to calibration solutions for intrinsic and extrinsic matrices, and those packages were used for the three sensors. The algorithms are derived from OpenCV's implementation of Zhang's calibration algorithm [21]. They require image streaming from the sensors viewing a calibration pattern, usually a checkerboard or an asymmetrical circular pattern. The algorithm would work by matching the predefined pattern configuration to pattern detections in real time, converging to the calibration solution after enough detections at varied distances and angles.

The sensors would need to clearly see the pattern at the same time for extrinsic calibration to work. If individual intrinsic solutions were the only parameters required, the calibration pattern would have been much easier to choose. To form a pattern that the LiDAR, thermal sensor, and optical sensor could see, an asymmetrical circular pattern was chosen with the circles cut out of the board for the LiDAR to easily identify the circular holes. The board material was white, and the calibration was performed with a dark background so the webcam could see the contrast. Additionally, a heat gun was used on the board so that the thermal sensor could easily see the pattern. This setup was derived from the calibration work by Choi and Kim [2], and the physical calibration setup is shown in Fig. 2.

The pattern was printed using a precision cutting machine, and tutorials were used for configuring the calibration library [22, 23]. During the initial calibration process, the results were far from what was expected, and it was found that the original asymmetrical circular pattern was symmetrical around the pattern's center as seen in Fig. 2. The calibration algorithm was frequently recognizing the pattern as right-side-up when it was truly upside-down, greatly skewing the results. To fix the problem, one column on the right side of the pattern was covered up, and the resulting calibration solution was sufficient. Lastly, the translation parameters that were output from the algorithm were severely off. The inaccuracy only in the translation result could possibly be attributed to the fact that the sensors were placed so closely together. Manual translation measurements were taken between the sensors to supplement this.

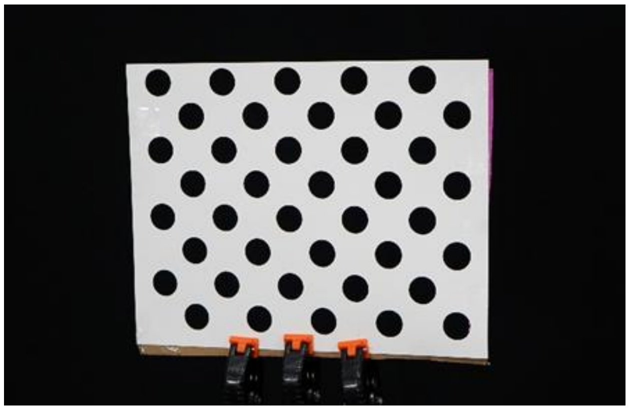

Fig. 2 Calibration Pattern.

D. Data Acquisition

Image data collection was carried out in a motion capture room to validate the future workflow results. The motion capture results were not critical for the research, and their results will not be addressed in this paper. However, the methodology taken for the motion capture will be explained, and the motion capture data will be shared [24]. To capture the location of the sensor mount, at least three motion capture markers needed to be attached to it as shown in Fig. 3. Likewise, all recorded objects of interest needed to be tracked. Markers were placed on the objects such that their reflections back to the LiDAR sensor were minimized. Note that the motion tracking system employed the use of OptiTrack sensors [25] and Motive software [26].

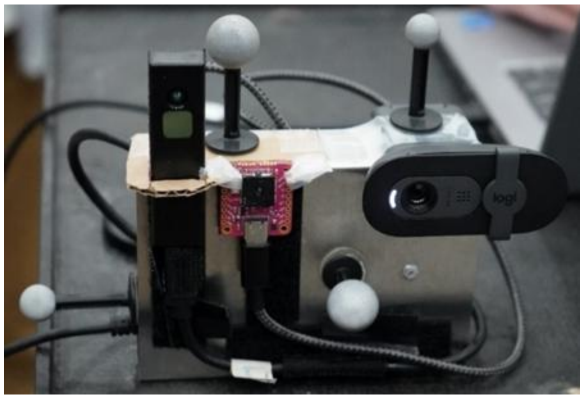
Fig. 3 Motion Capture Setup.

The object detection models needed to be trained to detect people first, so training images were collected by having one team member move around in the sensor mount's field of view for approximately 1 minute. Other training scenarios included one person sitting down, and a different person moving around upright. With a synchronized image collection time of about five images per second, each round of training data collection produced roughly 300 images.

Testing images were also gathered during the same session for assessing model performance and fusion algorithm performance. Different testing scenarios were recorded, including regular conditions like the training data. However, other scenarios were designed to exploit the weaknesses of the different sensors.

One scenario involved placing a "dummy" model of a person in front of the sensors. It was unable to stand upright, so it was seated in a chair. Training images of team members seated in a chair were also collected to compensate for the difference in posture and location of the seated dummy. Additionally, the authors dressed the dummy in clothes to help it look like the real people being recorded. This scenario was chosen because the models deployed on the webcam and LiDAR would probably identify the dummy as a human, but the model deployed on the thermal sensor would not recognize any heat signature.

Another scenario involved detecting people in the dark. All external light sources were blocked, and the lights in the motion capture studio were turned off to create total darkness. Then, a team member moved around in the sensor mount's field of view for approximately 20 seconds to capture images. The models for the thermal and LiDAR sensors would detect a person, but the model on the webcam would see nothing.

The last scenario involved displaying an image of a human on a screen and recording the screen. The model on the webcam would probably detect a person; however, the models on the thermal and LiDAR sensors would only see a flat surface. In addition to the image data with regular conditions, an additional testing scenario in which two team members were moving around in the field of view was collected to see if the algorithm could detect multiple objects at the same time.

E. Model Training

With the image data from the three sensors acquired, object detection models were able to be synthesized from the training data. The images were loaded into Label Studio [27] for manual bounding box drawing around the objects of interest in the images. Because the goal of the project was to evaluate and compare model performance between the fusion workflow and individual models, it was not important to have robust models trained. Thus, the training and testing images were captured with the same background, same people, same clothes, same poses, and same sensor mount orientation. As a result, very few images were required to train object detection models that would be able to identify the people in the testing data.

So, 75 images from the training data were chosen for training each of the three models: 25 images per training scenario (one person standing, the same person sitting, and a second person standing). Validation datasets were also created in Label Studio as required by the training process, and they contained 15 images each. Once the images were manually labeled, the datasets were exported in the COCO dataset format. After some small adjustments, the datasets were ready for model training.

To train the models and deploy the models for object detection, the MediaPipe framework from Google was chosen [28]. Google's team behind MediaPipe also provided a tutorial for data labeling and model training that was followed

[29]. The training script was written based on the tutorial [30], and each model was completely trained in less than 10 minutes. It is important to note that the models were trained using the MobileNet architecture from TensorFlow Lite. The resulting models were notably lightweight, which was a central focus of the research.

F. Fusion Workflow Methodology

With the object detection models trained, testing data acquired, and the sensor calibration completed, the detection fusion workflow was ready to be built. The workflow was developed in a virtual environment separate from the ROS2 environment [31]; also, note that the ROS2 environment was used for algorithm deployment in real time [32, 33]. The full step-by-step process that the workflow followed will not be addressed in this paper, but it has been detailed in the main fusion workflow environment [31].

Instead, in this section, the critical mathematical steps behind point transformations from the LiDAR sensor to the other two sensors will be presented. The rest of the workflow steps involved coding logic and decision-making based on confidence and box intersection thresholds, and the essential steps in that process will also be explained.

Given a pixel coordinate pair in the LiDAR image and the associated depth value at that pixel, denoted u_l, v_l, and z_l, respectively, the position vector of the point from the LiDAR's camera frame is given by

$$\vec{r} = z_l * K_l^{-1} \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix}, \tag{1}$$

where K_l is the LiDAR camera's intrinsic matrix. Note that the pixel coordinates were chosen to be the corners of the bounding boxes, and the depth values were chosen as the values at the centers of the bounding boxes which should match the depth of the person detected in the image. Next, the position vector can be extrinsically transformed into the other cameras' reference frames by

$$\vec{r_s} = R_{cs}^{-1} R_{cl} \vec{r} + T_{sl}, \tag{2}$$

where R_{cs} is the rotation matrix from the other sensor's reference frame to a placeholder orientation given by the calibration, R_{cl} is the rotation matrix from the LiDAR sensor's reference frame to the same placeholder orientation, and T_{sl} is the translation vector from the LiDAR frame to the other sensor's frame. Lastly, to transform the position vector expressed in the other sensor's reference frame to pixel coordinates, the formula is given by

$$\begin{bmatrix} u_s \\ v_s \\ \sim \end{bmatrix} = K_s \begin{bmatrix} r_{s,x}/r_{s,z} \\ r_{s,y}/r_{s,z} \\ 1 \end{bmatrix}, \tag{3}$$

where K_s is the intrinsic matrix of the other sensor, and $r_{s,x}$, $r_{s,y}$, and $r_{s,z}$ are the x, y, and z components of the position vector expressed in the other sensors' reference frame, respectively. Note that the coordinate system for any camera reference frame has the x-y plane in the plane of the image, and the z axis is pointing out in the direction the image is looking. By the right-hand rule, the x axis is pointing horizontally to the right, and the y axis is pointing down.

After the bounding boxes were transformed from the LiDAR sensor to the thermal and webcam sensors, a decision-level fusion process was executed to correlate detections between the different sensors, ultimately reaching a final decision on which detections should be kept or ruled out. Specifically, the degree of overlap between each bounding box from the two sensors was calculated, and the highest degree of overlap between two boxes was noted if it was above a threshold. The noted pair of boxes was then considered an "agreement" between the two sensors. After looping through the rest of the boxes and noting more overlapping box pairs, a final detection decision would be reached based on which sensors were required to agree.

Although other studies have investigated similar methods of box transformation and correlation for decision-level fusion, the use of lightweight detection models and the addition of a sensor agreement choice among three modalities was what distinguished the research from existing studies. In summary, instead of relying on feature-level or data-level methods of fusion, a straightforward decision-level approach was chosen. Exploiting the utility of LiDAR depth images allowed for fast and sufficiently accurate transformations.

Some of the important parameters passed into the fusion algorithm included the following: the maximum number of detection results that could be passed by the individual object detection models, the detection confidence threshold for the individual models, the intersection over union (IoU) threshold for box correlation, and the decision-making mode, which would decide which sensor detections needed to agree for a detection to be kept. The best workflow performance was found with the following parameters: max results at 3, confidence threshold at 0.5, IoU threshold at 0.4, and decision-making mode requiring the LiDAR and thermal sensors to agree. The proper tuning of these parameters was important for good fusion algorithm performance, and more discussion on this topic will be provided after the results are presented.

G. Performance Assessment Methodology

Once the fusion algorithm was developed, the new method was compared to the performance of individual models deployed on the individual sensors. The comparison was approached in two ways: a qualitative way, and a quantitative way. The qualitative method involved deploying the fusion workflow and individual models at the same time on the recorded testing videos [32, 33]. Viewing the two video streams side-by-side with detection outputs would be sufficient to indicate which method performed better. The quantitative method would compute average precision for the fusion algorithm and for each individual model on a hand-picked ground truth dataset [31, 34]. Because the fusion workflow performed best on confidence thresholds greater than zero, the average precision calculation did not cover the full confidence range, but the confidence thresholds were kept consistent between the two methods.

The ground truth dataset was randomly chosen from the testing data, but the same number of images was chosen from each scenario and sensing modality. Three regular scenarios were used: one person walking, one person sitting, and two people walking. Also, three other special scenarios were used: one person walking in the dark, the testing dummy sitting, and one person standing projected on a screen.

For each scenario, 30 images were chosen, with ten images corresponding to each sensing modality. The images were then imported to Label Studio and labeled for ground truth detections. Ground truth was also labeled for the webcam in the dark scenario by mapping the LiDAR ground truth boxes into the webcam's pixel coordinate system. The ground truth dataset was then exported and used in an analysis code to compute average precision.

Part of the analysis gave the option to perform a monotonicity correction on the precision-recall curve results prior to finding the area under the curve for average precision. The correction would ensure that precision values could not increase as recall values increased. Average precision results will be presented with both the monotonicity correction (smoothing) and without.

IV. Results

First, the quantitative results from the average precision computations will be presented in Table 1 and Table 2. Table 1 considered only the standard testing conditions, and Table 2 considered the full ground truth dataset, including the challenging scenarios. After that, qualitative results will be shown in Fig. 4 and Fig. 5, providing examples of outputs from two different scenarios. In the next section, discussion and interpretation of the results will be presented.

Table 1 Average Precision Results – Standard Conditions Only.

Result	*AP (smoothing)*	*AP (without smoothing)*
Idv LiDAR	95%	95%
Idv Thermal	95%	95%
Idv Webcam	90%	90%
Fused LiDAR	90%	90%
Fused Thermal	90%	90%
Fused Webcam	90%	90%

Table 2 Average Precision Results – All Conditions, Including Tricking Data.

Result	*AP (smoothing)*	*AP (without smoothing)*
Idv LiDAR	84.5%	76.2%
Idv Thermal	96%	96%
Idv Webcam	67.4%	66.8%
Fused LiDAR	90%	90%
Fused Thermal	90%	90%
Fused Webcam	90%	90%

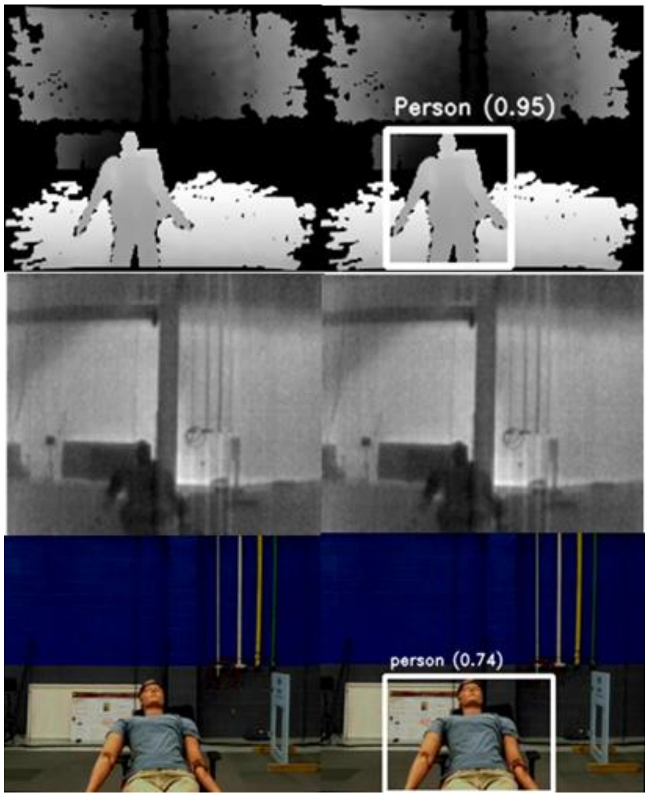

Fig. 4 Testing Dummy Scenario.
The left column is for the output from the fusion algorithm, and the right column is for individual model output. The modalities are ordered LiDAR, thermal, and webcam from top to bottom. Any returned detections are shown in the images.

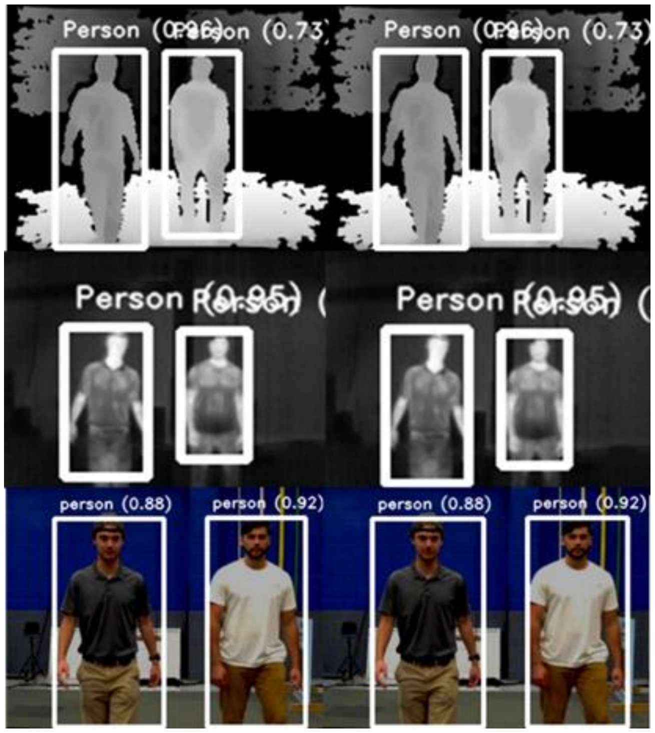

Fig. 5 Two People Scenario.

V. Discussion

When viewing the recorded testing videos with detections projected on them from the individual models and from the fusion algorithm, it was clear that the fusion algorithm performed more reliably than the individual models, especially on the more difficult testing scenarios. The individual model deployed on the webcam could not detect the person in the dark scenario, and it returned false detections in the testing dummy and screen projection scenarios. Similarly, the model deployed on the LiDAR sensor falsely detected the testing dummy. However, in the testing scenarios with regular conditions, the individual models performed well. In all scenarios, the fusion workflow performed reliably.

These observations from the qualitative assessment were somewhat matched by the quantitative results in Table 1 and Table 2. A higher average precision indicated better performance, and the fusion algorithm metrics agreed among all three sensors. The workflow produced a unified detection decision, so this result made sense.

When considering the full dataset, the individual models deployed on the LiDAR and webcam were outperformed by the workflow, but the individual model deployed on the thermal sensor performed better than the workflow. This happened because none of the testing images were designed to exploit a weakness in the thermal sensor, and calibration inaccuracies resulted in a few disagreements between transformed boxes when they should not have disagreed. Thus, the fusion algorithm missed a few detections in the ground truth dataset compared to the individual model deployed on the thermal sensor. In fact, this same inaccuracy is what resulted in the fusion algorithm having slightly worse performance than the individual models on the standard conditions in Table 1. If testing images were gathered to somehow "trick" the thermal sensor, then it would be likely that the fusion algorithm would outperform all three individual sensors. For example, a possible scenario for the thermal camera could be a person wearing a reflective thermal blanket, which should hide the heat signature. In all, the fusion workflow demonstrated competent performance and even superior performance in certain scenarios.

Moving on, the proper tuning of algorithm inputs was also essential for good workflow performance. If the maximum number of detection results was too high and/or confidence thresholds were too low, then the individual models would return numerous low-confidence detections that would sometimes interfere with the box correlation step, diminishing workflow performance. So, a confidence threshold of 0.5 was set and a maximum number of detection results was set at 3 to limit the number of potential low-confidence detections that would be passed by the models. One of the studies previously listed addressed this issue in a different way [5].

Similarly, if the IoU threshold set for box correlation was too high, then small inaccuracies in the calibration and depth-resolving steps would return IoUs below the threshold, even though the detections should have agreed. So, the IoU threshold was relaxed to 0.4 from its original value of 0.5, yielding much better model performance. Also, given that the testing data did not provide a challenging scenario for the thermal sensor, the decision-making mode was set such that the LiDAR and thermal sensors had to agree for the workflow to output the detection.

Lastly, from viewing the average precision results, it is likely that the models were overtrained, because the training data and testing data were quite similar. However, the overtrained models did not affect the primary goal of the study: comparing performance between the individual models and the combined workflow.

VI. Conclusion

The proposed method of decision-level fusion between lightweight object detection results from different sensors has been proven to perform sufficiently on the testing data. The usefulness of the method resides in its minimal complexity and efficiency. There is a wealth of future work that may be taken on to improve the method and build upon it. Adding semantic segmentation and better extrinsic calibration would improve depth resolving and transformation accuracy. Models trained on more diverse datasets would handle challenging scenarios more accurately. The problem of adapting the decision-making mode to different scenarios either autonomously or manually is another important factor, especially since there are three sensing modalities. Also, deployment of the method on embedded computers for real-time testing would be useful, because the motivation behind the study was a scenario involving an autonomous drone searching for people.

Acknowledgments

The authors thank their advisors, Mr. Matthew Berk and Dr. Chuangchuang Sun, for their guidance and support throughout this research. Gratitude is also extended to Raspet Flight Research Laboratory for providing access to their facilities, which were integral to the data acquisition efforts in this study. The authors' gratitude is also extended to Mr. Rob Wolz for his insight and assistance in lectures and laboratory meetings.

References

[1] Mulyanto, A., Borman, R. I., Prasetyawana, P., and Sumarudin, A., “2D Lidar and Camera Fusion for Object Detection and Object Distance Measurement of ADAS Using Robotic Operating System (ROS),” *JOIV: International Journal on Informatics Visualization*, Vol. 4, No. 4, 2020, pp. 231–236.

[2] Choi, J. D., and Kim, M. Y., “A sensor fusion system with thermal infrared camera and LiDAR for autonomous vehicles and deep learning based object detection,” *ICT Express*, Vol. 9, No. 2, 2023, pp. 222–227. doi: 10.1016/j.icte.2021.12.016.

[3] Ophoff, T., Van Beeck, K., and Goedemé, T., “Exploring RGB+Depth Fusion for Real-Time Object Detection,” *Sensors*, Vol. 19, No. 4, 2019, pp. 866. doi: 10.3390/s20061797.

[4] Zhong, H., Wang, H., Wu, Z., Zhang, C., Zheng, Y., and Teng, T., “A Survey of LiDAR and Camera Fusion Enhancement,” *Procedia Computer Science*, Vol. 183, 2021, pp. 579–588.

[5] Gutfeter, W., and Pacut, A., "Fusion of Depth and Thermal Imaging for People Detection," *Journal of Telecommunications and Information Technology*, Vol. 86, No. 4, 2021, pp. 53–60. doi: 10.26636/jtit.2021.155521.

[6] James, T., Liberatore, P., and Rogers, E., “Fusion-Based Utilization and Synthesis of Efficient Detections,” Google Sites [online], Feb. 19, 2025. URL: https://sites.google.com/view/projectfused [retrieved 28 Feb. 2025].

[7] Logitech, Logitech Brio 101 Webcam, Logitech International S.A., Lausanne, Switzerland, 2021.

[8] Logitech Support, “Specification – Brio 101,” Logitech Support [online], URL: https://support.logi.com/hc/en-ca/articles/16131499767191-Specification-Brio-101 [retrieved 26 Nov. 2024].

[9] PMD Technologies, PMD Flexx2 Time-of-Flight (ToF) Sensor, PMD Technologies GmbH, Siegen, Germany, 2022.

[10] PMD Technologies, “flexx2 | Getting Started Guide,” PMD Technologies [online], Feb. 2022, URL: https://media.automation24.com/manual/405065.pdf [retrieved 26 Nov. 2024].

[11] PMD Technologies, “flexx2 Development Kit,” PMD Technologies [online], 2024, URL: https://3d.pmdtec.com/en/3d-cameras/flexx2/ [retrieved 26 Nov. 2024].

[12] FLIR Systems, FLIR Lepton 3.5 Thermal Imaging Camera, FLIR Systems, Wilsonville, OR, 2021.

[13] Teledyne FLIR, “TELEDYNE FLIR LEPTON® MICRO THERMAL IMAGING CORE PRODUCT DATASHEET,” Teledyne FLIR [online], Nov. 17, 2023, URL: https://flir.netx.net/file/asset/13333/original [retrieved 26 Nov. 2024].

[14] GroupGets, PureThermal 3 FLIR Lepton 3.5 Breakout Board, GroupGets, Inc., Austin, TX, 2021.

[15] AgileDevArt, “Linux Tips - Record Video from USB Camera in WSL (2022),” YouTube [online], Aug. 27, 2022, URL: https://www.youtube.com/watch?app=desktop&v=t_YnACEPmrM [retrieved 26 Nov. 2024].

[16] ROS 2, Humble Hawksbill, Software Package, Ver. 2.7.3, Open Robotics, Mountain View, CA, May 2022.

[17] Open Robotics, “ROS 2 Documentation: Humble,” ROS.org [online], 2025, URL: https://docs.ros.org/en/humble/ [retrieved 26 Nov. 2024].

[18] ROS Drivers, “usb_cam,” GitHub [online], 2024, URL: https://github.com/ros-drivers/usb_cam [retrieved Sept. 2024].

[19] PMD Technologies, “pmd-royale-ros,” GitHub [online], 2024, URL: https://github.com/pmdtechnologies/pmd-royale-ros [retrieved Sept. 2024].

[20] ROS Perception, “image_pipeline,” GitHub [online], 2024, URL: https://github.com/ros-perception/image_pipeline [retrieved Sept. 2024].

[21] Zhang, Z., “A Flexible New Technique for Camera Calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, 2000, pp. 1330-1334. https://doi.org/10.1109/34.888718.

[22] Open Robotics, “How to Calibrate a Monocular Camera,” ROS Wiki [online], Sept. 19, 2019, URL: https://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration [retrieved 26 Nov. 2024].

[23] Open Robotics, “How to Calibrate a Stereo Camera,” ROS Wiki [online], June 28, 2018, URL: https://wiki.ros.org/camera_calibration/Tutorials/StereoCalibration [retrieved 26 Nov. 2024].

[24] Rogers, E., “Motion_Capture,” Google Drive [online], Feb. 25, 2025, URL: https://drive.google.com/drive/folders/1HFoMSVxoYN-99lTYG4PSzyQWTtgzm0vO?usp=sharing.

[25] NaturalPoint, Inc., “OptiTrack Prime^{X} 41,” Corvallis, OR.

[26] NaturalPoint, Inc., “Motive,” Version 2.3, Corvallis, OR, 2021.

[27] Label Studio, "Quick start guide for Label Studio," Label Studio Documentation [online], URL: https://labelstud.io/guide/quick_start [retrieved 26 Nov.2024].

[28] Google, MediaPipe, Software Package, Ver. 0.10.18, Google Inc., Mountain View, CA, 2023.

[29] Google for Developers, “Training an object detection model - ML on Raspberry Pi with MediaPipe Series,” YouTube video, YouTube, Dec. 18, 2023, URL: https://www.youtube.com/watch?v=X9554zNNtEY [retrieved 26 Nov. 2024].

[30] Rogers, E. “MediaPipe Model Maker,” GitHub, [online], 2025, URL: https://github.com/ethanrogers15/mediapipe_model_maker.

[31] Rogers, E. “Project FUSED,” GitHub, [online], 2025, GitHub, URL: https://github.com/ethanrogers15/project_fused.

[32] Rogers, E. “Project FUSED ROS,” GitHub, [online], 2025, URL: https://github.com/ethanrogers15/project_fused_ros.

[33] Rogers, E. “Project FUSED ROS Demo,” [online], 2025, URL: https://github.com/ethanrogers15/project_fused_ros_demo.

[34] Hui, J. “mAP (mean Average Precision) for Object Detection,” Medium, [online], March 6, 2018, URL: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173 [retrieved 12 Feb. 2025].