BigData Fusion for Trajectory Prediction of Multi-Sensor Surveillance Information Systems

G. Cascavilla, *TU/e - JADS*, Den Bosch, (NL), g.cascavilla@tue.nl
A. Cuzzocrea, *University of Calabria*, Rende, (IT), alfredo.cuzzocrea@unical.it
D. De Pascale, *TiU - JADS*, Den Bosch, (NL), d.de.pascale@tue.nl
M. Omidbakhsh, *Concordia University*, Montreal, (CA), mandana.omidbakhsh@concordia.ca
D. A. Tamburri, *TU/e - JADS*, Den Bosch, (NL), d.a.tamburri@tue.nl

2023 IEEE International Conference on Big Data (BigData) | 979-8-3503-2445-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/BigData59044.2023.10386779

***Abstract*—Video surveillance information systems assist forensics to examine and analyze the evidence from crime scenes to develop objective findings in the investigation of crime. Often, the existing surveillance information systems exploit an array of security cameras and IoT devices monitoring the same crime scene from different points of view while the crime unfolds over a range of time. However, none can automatically and selectively merge big data streams connected to such systems to provide a holistic, end-to-end safety picture.**

This work proposes a trajectory prediction architecture framework within a multi-sensor surveillance system. We developed a novel position measurement technique using monocular depth perception networks with multi-camera setup using triangulation. We tested and compared our technique with a single camera sensor in our first experiment and as the multi-camera setup determines the position of our target more accurately, we used our measurement function in our second experiment. In our second experiment, we employed the Unscented Kalman Filter (UKF) for predicting the trajectory of the target, and proved that UKF has good potential for being used in surveillance systems. Lastly, we designed a general architecture framework for big data analysis in multi-sensor surveillance systems consisting the four layers: the Sensor Layer, the Single Sensor Computation Layer, the Data Fusion and Interpretation Layer, and the Human Interaction Layer.

***Index Terms*—Surveillance Systems, Anomalous Trajectory Recognition, Multi-sensor Data Fusion, Unscented Kalman Filters (UKF), Distributed Sensor Networks (DNS).**

I. Introduction

A multi-sensor surveillance system is a system to secure a specified area utilizing multiple sensors. There are many different implementations of such systems (i.e., office security, home security, event security, public space security), but they usually include one or more camera sensors. Examples where such systems are implemented, are banks, shopping malls, public transport stations, festivals, or cities [1]–[6].

Within multi-sensor surveillance systems, trajectory prediction is one of the fundamentals for capabilities like automatic target monitoring or anomaly detection [7], [8]. However, the target is needed to be located and its position determined before a target trajectory can be predicted. Several approaches, even inherited from related contexts, have largely considered the problem of supporting *approximate query answering over data streams* (e.g., [9], [10]).

Multiple big data fusion techniques have already been researched to determine a target's position using one or more sensors. The triangulation technique is one of the techniques employed in multi-sensor surveillance systems. The angle determines the position of the target in 3D space towards the target from two or more sensors with the condition that the target is visible from at least two sensors. Placing sensors to make everything visible from at least two places can be very challenging [11].

A second technique used to fuse data from multiple sensors is the Kalman Filter (KF), applied for linear system models, and the Extended Kalman Filter (EKF), used for non-linear system models [12]. Although EKF is a popular method, it heavily relies on local linearity, can only approximate the function to the accuracy of the first-order derivative, and in practice proves to be very complex in implementation [13]–[15]. UKF solves the shortcomings of EKF by using a technique called Unscented Transformation to capture the true mean and covariance of the system. It then uses the non-linear system to reach its prediction [15].

To overcome the triangulation approach limitations, our work proposes a novel position measurement technique that uses only one sensor and a method called monocular depth perception. The monocular depth perception method uses a neural network to determine depth from a 2D image. Our proposed technique runs the output of a single (monocular) camera through a depth perception network, and by using the artificially added depth and the angle from the camera towards the target, we calculate the position of the target. To validate the performance of our technique, we compared our proposed single-sensor method against a setup using the triangulation method, forming our first research question (RQ):

RQ1. *To what extent does a multi-sensor data fusion approach using triangulation outperform a single sensor approach using monocular depth perception in trajectory mapping?*

Then, to evaluate the prediction with multi-sensors in a non-linear system, we propose to use UKF as a data fusion method for predicting the trajectory of our target, forming our second research question:

RQ2. *To what extent does the UKF outperform the baseline approach and be able to predict the trajectory, if the performance of triangulation for trajectory mapping in a multi-sensor system used as a baseline?*

Thus the main contributions of this work are as the fol-

lowing:*(1) We defined a generic architecture framework for surveillance systems including the physical sensors to the users screen. (2) We showcased that our multi-camera setup is computationally lighter and more precise in mapping our target's trajectory than our single-camera setup. (3) We showed that the UKF data fusion method can successfully be applied to predict target's trajectory. (4) An online available repository reporting the raw data and a replication package in the study context for further research and new considerations by the community available in [16].*

II. Background and Related work

In this section, we review monocular depth perception, data fusion for non-linear systems methods and target tracking in two dimensions used in surveillance systems.

A. Monocular depth perception

Monocular depth perception methods are mainly researched in autonomous driving vehicles [17]–[21]. The main motivation comes from the downside of the alternatives: depth cameras, stereo cameras, and LiDAR. The main disadvantage of depth cameras is their viewing range. For the stereo cameras, instead, it is their calibration [20]. LiDAR (Light Detection And Ranging) is the most popular alternative. However, compared to a monocular camera, it is much more expensive, especially if a high-resolution LiDAR is required [18]–[20]. The main advantages of monocular cameras are that they are readily available in every modern car, can see up to approximately 100 meters, and are much cheaper in comparison [18]–[20].

A monocular depth perception network can be trained with supervised or unsupervised learning [21]. The data for supervised training consists of 2D images combined with ground truth depth data, which has to be captured by specialized equipment. As described in [21], unsupervised training data consists of two images captured by calibrated binocular cameras and is, therefore, easier to capture [21]. In this work, we focus on networks that are trained unsupervised as described by Godard et al. [21]. Capturing data for unsupervised training does not require special equipment. Instead of a model trained with images and the ground truth depth data, these networks are trained on pairs of images (left and right images) that are captured by a pair of calibrated binocular cameras. The problem the network tries to solve is the disparity between the two images. In other words, it tries to find the function that warps the left image to the right image. In [21], they trained networks on five different data sets using two different encoders, resulting in ten different model networks. An overview of the models with their data set and encoder can be found in table I.

In the work of [20], monocular depth perception is used in a system designed to determine the position of objects. It focuses on detecting vehicles around a car and the distance between these vehicles from the car, using monocular cameras in the car. The detection has been done by determining depth using a monocular depth perception network and converting the result into a Pseudo-LiDAR Point Cloud. Following this operation, LiDAR-based algorithms are used with the created point cloud to achieve the result. Unlike this research, our paper uses monocular depth perception in a system that focuses on trajectory prediction and mapping of a human target in 3D space within the context of a surveillance system.

TABLE I
Models considered for monocular depth perception in the single-camera setup.

Model name	Data Set	Encoder
City Scapes	Cityscapes	Self defined 14 layer encoder
Eigen	Eigen	Self defined 14 layer encoder
Kitti	KITTI	Self defined 14 layer encoder
City 2 Eigen	Cityscapes with Eigen Split	Self defined 14 layer encoder
City 2 Kitti	Cityscapes with KITTI split	Self defined 14 layer encoder
City Scapes Resnet	Cityscapes	resnet
Eigen Resnet	Eigen	resnet
Kitti Resnet	KITTI	resnet
City 2 Eigen Resnet	Cityscapes with Eigen Split	resnet
City 2 Kitti Resnet	Cityscapes with KITTI split	resnet

B. Data Fusion for non-linear systems

The Kalman Filter (KF) is a data fusion algorithm that analyzes and solves a broad class of estimation problems [22]. The KF framework consists of a system model, a measurement equation, a predicted state estimate, and an error covariance matrix. The system model represents how the system behaves. In the case of KF, this is always a linear (dynamic) system.

Nevertheless, scarcely a target walks in a straight line with the same speed. They usually stand still, start running, or perhaps turn around. Hence, their dynamic behavior can not be solved using an underlying constant velocity linear model, but it needs to be addressed using non-linear models. Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) are designed as estimators for non-linear systems [14].

EKF is designed to tackle non-linear problems [14], [23]. It linearizes all the non-linear models and applies the original KF. However, this process can produce unreliable estimates if the assumptions of local linearity are severely violated [13], [14]. Then, the state distribution is approximated by a Gaussian random variable (GRV) and put analytically through the linear approximations of the non-linear function. The linear approximations are calculated by taking the first-order derivatives of the non-linear function, namely the Jacobian matrix. Unfortunately, in some systems, it is numerically tricky to compute the Jacobians [13], [15], and EKF turns out to be too complex to implement or tune as it can have many variables [13]–[15].

UKF iis designed to tackle the main shortcomings of EKF [13]–[15]. The UKF is easier to implement with the assumption of calculating the Jacobians, and the model of the system and measurements can be treated as *black boxes* [14]. UKF specifies the state distribution, represented by a GRV, using an Unscented Transformation (UT) technique to create a minimal set of sample points [15]. These points capture the true mean

and covariance of the GRV completely. When propagated through the true non-linear system, it accurately captures, for any nonlinearity, the posterior mean and covariance to the third-order [15], which is more accurate than the first-order approximations that EKF produces [14], [15]. UKF is easier to implement and can more accurately approximate the state function [14].

Ellis et al. [24] present an object-tracking system using a network of multiple cameras. It was a proof-of-concept system using overlapping and non-overlapping CCTV cameras to detect and track vehicles and pedestrians in 3D.

Black et al. [25] focus on tracking objects within the same observed scene. The 2D object tracking was done by background extraction. Object tracking in 3D was done by using the KF and assuming a constant velocity motion model. In contrast, UKF is used assuming a nonlinear estimation problem. The weaknesses they address in their approach are a performance drop when an object splits into multiple objects or a possible breakdown when an object changes its trajectory significantly during occlusion. In the work of [26] the (linear) KF is used to track objects over multiple cameras. The network of cameras has non-overlapping views and the focus is not on tracking an object in 3D but on using the KF to track the object over multiple (non-overlapping) camera views and the blind spots between them. The tracked objects in the paper were only vehicles on the highway and can rely on trajectories that always follow the road.

C. Target tracking in two dimensions

Before a target's location in 3D space (the observed scene) can be calculated, its position in the camera's view must be determined. The process of tracking a target from the viewpoint of a camera is referred to as *2d target tracking*.

In surveillance systems, background subtraction is often used to track objects in a scene [27]. The technique implies the localization of the pixels' centroid moving in a scenario's foreground. The main downside with background subtraction methods is that the camera must always be stationary. Otherwise, the background changes too much, and the foreground can not be extracted. Therefore, this research uses a single object tracking algorithm from the popular Open Source Computer Vision (OpenCV) library to be more future-proof for moving or panning cameras. These algorithms track an object based on the features of an object (e.g., color, shape) and their position in previous frames [28]. The library contains the following algorithms: Boosting, MIL, MedianFlow, MOSSE, TLD, KCF, GOTURN, and CSRT.

The OpenCV object tracking algorithms have been compared in [28], [29]. The algorithms have different strengths (e.g., robustness, speed, accuracy, recovery failure) and should be chosen under the context and purpose of the 2d tracking. Nonetheless, the 2D tracking of the target is not the main focus of this research. Therefore, to determine what algorithm best suits our context, every algorithm in the OpenCV library is tested on our data, and our findings show that Channel and Spatial Reliability Tracker (CSRT) [30] is the best tracker in terms of robustness and precision. This finding aligns with research on the OpenCV comparison object tracking algorithms [28], [29].

In the work of [26] the (linear) Kalman Filter is used to track objects over multiple cameras. The network of cameras has non-overlapping views and the focus is not on tracking an object in 3D but on using the Kalman Filter to track the object over multiple (non-overlapping) camera views and the blind spots between them. The tracked objects in the paper were only vehicles on the highway and can rely on trajectories that always follow the road.

III. System Architecture Framework Design

This section presents our surveillance system architecture, explaining all the layers within the architecture. We defined four layers, each with specific tasks and purposes. However, each layer can influence the other layers based on its design choices (e.g., the types of sensors used, the purpose of the entire system). The architecture can be found in fig. 1. Our proposed architecture is inspired by the JDL model [31]. Similar to JDL model levels, our architecture defines layers with their responsibilities and considerations. However, one of the differences between them is that the JDL model is designed as a functional model [32], whereas this architecture is a process model. A process model shows the interaction between parts of the system (e.g., data from the first layer goes to the second layer) and can not be executed. Another difference is that the JDL model focuses entirely on the data fusion domain, abstracting the sources (e.g., sensors input) and the interface [32], [33], though our proposed architecture is focused on surveillance systems with a higher-level abstraction that takes (amongst others) the specific sensors, the purpose of the system, and the operators of the system into consideration.

A. Sensor Layer

Starting on the left in the architecture, we have the sensor layer, referred to as the physical layer. It includes all physical devices used to gather information. There can be two-to-many sensors in a multi-sensor surveillance system. There are many types of sensors (e.g., motion detection, temperature, cameras, microphones), each of which can be produced by different companies. The sensors in a system can be either of the same type or a combination of different types. The sensor layer produces the data processed in the single sensor computation layer (e.g., videos from the camera, sound from a microphone, temperature from a temperature sensor). The main considerations in this layer are the type and placement of the sensors used in the system.

B. Single Sensor Computation Layer

All the data collected by the sensors are processed in the single sensor computation layer. If a system has three cameras and two microphones, each camera and microphone requires further processing, depending on the sensor type used and the system's objective. For example, data from a microphone can be processed with a speech recognition algorithm, converting

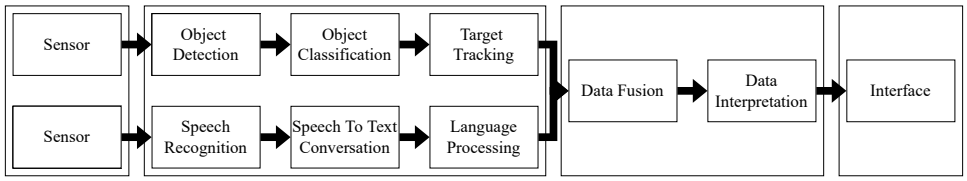

Fig. 1. Third generation surveillance system architecture.

the data into text. Likewise, the camera sensor data can be processed in the same system to detect and track an object. However, in a different system, the microphone data can be used only to detect noise, and the camera sensor data can only detect movement. These examples show why choosing the right sensor and defining the system's purpose is important, as it is challenging to monitor a street with only a microphone and detect noise with only a camera. This layer produces the data for the data fusion layer. It pre-processes the raw sensor data into something useful to combine it in the next layer (e.g., from a video stream to an object's location, from sound to text).

C. Data Fusion and Interpretation Layer

In this layer, the pre-processed data from the individual sensors are combined to create a higher-level understanding of the scene. The combination of data increases the amount of insight collected by each sensor. For example, tracking an object from two individual cameras can be combined to map the object's trajectory in 3D space. More advanced models can be used in this layer, like a model trying to interpret what is happening (e.g., a man walks to the door, a bag left unattended for 5 minutes). However, these advanced techniques are not relevant to this research. Hence, we focus on data fusion for trajectory mapping.

D. Human Interaction Layer

The right-most layer of the system is the human interaction layer. It provides operators (e.g., security, police, caretakers) with an overview of the system's findings, like the target trajectory or suspicious behavior detected. It also eases the decision process because operators may practically act when an anomaly has been detected.

Although it is irrelevant for this research, this layer is also important for training the interpretation model(s) used in the previous layer. These models try to detect events (e.g., a bag left unattended for 5 minutes) and classify them as anomalies or normal behaviors. When a situation is falsely interpreted as an anomaly, the operator should mark it as a false positive. This information can be taken into account for the purpose of training the anomaly classifier.

IV. Experimental Design

In this section, the general workflow of the experiments are introduced and the setup of the experiments run during our testing phase is explained. A detailed explanation and implementation of the methods used in each experiment are discussed in Section V.

A. Experiment 1

This experiment is designed to answer **RQ1**, our first research question. We compare the trajectory mapping performance of two methods: triangulation and monocular depth perception. Both methods are tested on the same scene where the target walks in a zigzag pattern through the observed area. The triangulation method calculates the position of the target in 3D space using the angle towards the target from two cameras. On the other hand, the monocular depth perception method uses a single video frame from one camera and tries to add depth to the frame of the video artificially. Hence, the monocular depth perception calculates the position of the target using the angle towards the target and the artificial depth added by the approach. The angle towards the target is calculated using the camera's resolution and field of view (FOV) combined with the target's pixel coordinates in the video frame. The angle calculation and the tracking of the target are explained in more detail in Section V.

The result for both methods is a list of target positions plotted in a graph representing the top view of the monitored scene. These positions are then compared to each other and to the scene recording to determine their performance on trajectory mapping.

1) Experiment workflow: The first experiment defines two workflows one for each method used, in the online Appendix [34] the architecture in fig. 9. The workflows show where the data comes from and in what form it flows through the system. It also shows the methods for calculating the target's position in 3D space. The first workflow, namely *multi camera*, uses the triangulation approach and two cameras to determine the target's position. The second workflow, namely *single camera*, uses the monocular depth perception method and a single camera to localize the target in a 3D space.

The *multi camera* workflow starts with two cameras on the left. Each camera records a video and sends it to the target tracking process. The target tracking extracts the pixel coordinates of the target and sends them to the triangulation method. The triangulation, starting from the pixel coordinates, calculates the angle toward the targets to determine the target position in 3D space. Lastly, the target's position is plotted on a top-view map of the scene.

The *single camera* workflow starts with only one camera. First, the camera produces a single video of the scene. Then, it tracks the target and uses its coordinates to calculate the angle towards the target. Next, the monocular depth perception method adds artificial depth to the video frames. As a result, the approach uses the angle toward the target's pixel coordi-

nates and the depth to determine the position in 3D space. In the last step, the position is plotted on a top-view map of the scene.

The experiment outputs two top view maps, one produced by the multi-camera workflow using two cameras and the triangulation method, and the other produced by the single-camera workflow using the monocular depth perception method and only a single camera. Finally, we compare these two maps to each other and the actual behavior of the target in the scene.

B. Experiment 2

The second experiment is designed to answer **RQ2**, our second research question. This experiment compares the performance of triangulation and UKF methods on their ability to predict and map the trajectory of our target. The first method uses triangulation to calculate the position of the target in 3D space and the second method uses UKF to predict the target's trajectory by using a combination of real measurements and an estimation of the velocity and direction of the target. Both these methods use two cameras observing the same scene. The methods are tested in different scenarios where the target's walking pattern differs every time. Section IV-B1 discusses the experiment's workflow executed for each video.

1) Experiment workflow: In the online Appendix [34] the architecture in Figure 10 displays the workflow of experiment two. It starts with two cameras producing a video of the same observed scene. The target tracking process extracts the target's pixel coordinates for each video.

With the pixel coordinates of the target in each video, the workflow splits and uses Triangulation or UKF to determine the trajectory. Triangulation uses pixel coordinates to determine the angle toward the target from both cameras. The angles are then used to calculate the target's position in 3D space. On the other hand, UKF creates a model that tracks the target, estimating its velocity and direction. The estimation of velocity and direction is based upon real measurements, measured using an exact approach, such as Triangulation. The velocity and direction are updated at specific intervals when the model receives new measurements. Finally, based on the estimated velocity and direction of the target, the position of the target is predicted.

The experiment ends with a list with the coordinates of the target in 3D space. For both methods, these coordinates are plotted in a top-view graph. We compare the outcome of both trajectories against each other and against the actual behavior of the target in the video.

V. Experimental Setup

This section gives a detailed overview of the data obtained and used in the experiments. The experiment environment, the workflow and the implemented target tracking, angle calculating towards the target, monocular depth perception, triangulation and UKF methods are described.

A. Data

The data used in the experiments is video data manually created in a controlled environment. Two smartphones, a Samsung Note 10 plus and a Xiaomi Mi 9t pro, filmed a scene from different positions. The video resolution is 1080p, and its framerate is 60fps. The target in the video performs several patterns, like walking in a straight line, zigzagging, or stopping in the middle of a walk. The different movement patterns are chosen to challenge the prediction algorithm. Sudden direction changes make predicting movement and making a good test case hard. The videos are cut so the center of the target is always in frame for both cameras. An overview of the camera sensors and their settings can be found in table VI available in the online Appendix [34]. An overview of the videos, the pattern performed, and the duration can be found in table II. Each video number in table II has two recordings, one from camera 1 and one from camera 2, both from different positions. The first experiment uses only video number 4, while the second uses all the videos.

TABLE II
Overview of videos used in the experiments.

Video number	Walking Pattern	Duration (seconds)	Resolution	Frames per second
1	Straight	5	1080x1920	60
2	Walk-stop-walk	17	1080x1920	60
3	U-turn	10	1080x1920	60
4	Zigzag	9	1080x1920	60

B. Experiment environment

Figure 2 shows the environment setup used for both the experiments. We see cameras 1 and 2 positioned approximately 11 meters apart. Both cameras can see each other. The videos are recorded outside in a backyard of approximately 7x12 meters during the day, always around noon. The videos are, therefore, shot in a bright environment. In addition, we chose to position the cameras across each other to replicate a public monitoring scenario, like a station or a festival terrain,

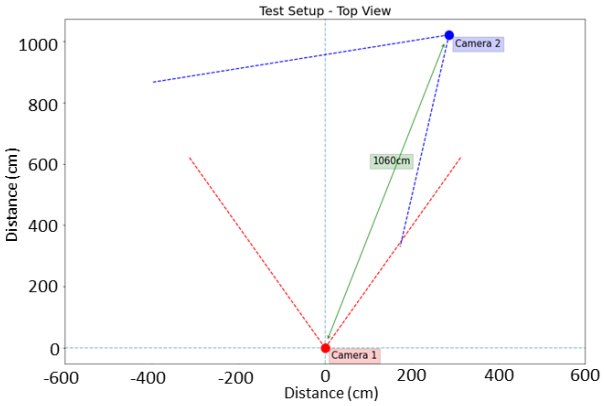

Fig. 2. Top view depiction of the test setup. The green line indicates the approximate distance between both cameras. The x-axis, y-axis, and green lines are in centimeters.

because two security cameras may overlap, recording partially the same scene.

C. Target tracking

Both experiments use target tracking. It takes as input a video shot from any of the cameras. To track the target, we first select the target by manually drawing a box around the target. Then, we create a tracker object from the OpenCV Python library, initializing it with the CSRT tracking algorithm [35]. Then, we play out the video, and for each frame, we take the centroid coordinates of the box and store them in a CSV file. We take the centroid of the box as it resembles the center of the target. Table III presents an example of the resulting tracking data.

TABLE III
EXAMPLE OF THE RESULTING CSV AFTER TRACKING THE TARGET.

index	object_id	frame_number	x-coordinate	y-coordinate
0	0	0	1883	394
1	0	1	1874	396
2	0	2	1866	397
3	0	3	1857	401
4	0	4	1847	403

D. Calculating the angle toward the target

Triangulation calculates the target position in both experiments based on the angles toward the target from both cameras. This section explains how to calculate the angle based on the target coordinates. The following equation calculates the angle of the target from the perspective of the camera:

$$\angle a = \arctan(\frac{\text{x-coordinate}}{d}) \tag{1}$$

Where $\angle a$ is the angle towards the target, with a degree range from 0 to the camera's maximum field of view (fov). The x-coordinate is the x coordinate of the target obtained from the tracking operation. The d is the camera's focus point.

Hence, if the x coordinate of the target is 0, we expect an angle of 0 degrees; if the x coordinate of the target is 1920, the camera's maximum resolution, we expect an angle equal to the maximum fov of the camera.

E. Monocular depth perception

The first experiment uses the triangulation or monocular depth perception method to determine the target's position and compares their results afterward. The depth perception experiment uses only camera 1 (camera details in table VI in the online Appendix [34]) on video number 4 because camera 1 has the widest fov and recorded the clearest video, and video 4 contains the zigzag pattern, one of the more complex patterns. As explained in section II-A, there are ten different networks shown in table I that we use to determine depth, each trained with different data sets or a different encoder. In this experiment, we parsed our video through each different network once and compared the output of each network to determine which network had the best result in determining depth. In the results, we compare the best-performing network with the triangulation method.

Before the analysis, the video must be pre-processed. The first step is to downscale the video from 60fps to 30fps to limit the number of frames we need to process. Then, each frame is converted to a 512x256 resolution due to the input format of the network. Figure 3 presents two processed frames of the same video, color-coded and interpretable by humans. Brighter is an object's color, and closer is the estimation. Darker is the color of an object, and further away is the estimation from the camera. After the pre-processing, we take each video frame and run it through the most accurate network.

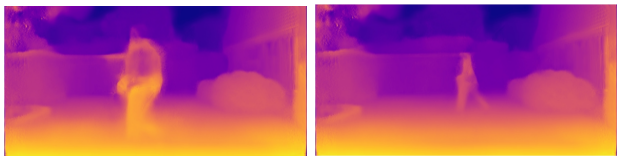

Fig. 3. Two resulting example frames of a video after being processed by the monocular depth perception network.

To calculate the target's position in 3D space, we use the angle towards the target and the estimated depth of the target. For each frame, we get the target's estimated depth by transposing the target's pixel coordinate from the 1080x1920 (60fps) video to the 512x256 (30fps) video and taking the depth value of this coordinate. The pixel coordinate brightness level gives the depth value. The coordinate resembles the center of the target. This process results in a data frame containing the angle towards the target and the estimated depth for each video frame. Plotting a point for each frame and drawing a line between those points gives the top view trajectory mapping.

F. Triangulation

Experiment 1 uses Triangulation as a multi-camera method, while Experiment 2 uses Triangulation as the baseline approach. First, it calculates the target's location in 3D space by combining data from two cameras, taking the x-coordinates from both cameras and converting them into angles using the method discussed in section V-D. Next, for each 15 angle, it calculates the average to compensate for the instability of the tracking algorithm and the difference between the approximation and the real position of the cameras, discussed in section VIII. Finally, knowing the angles, position, and orientation of the cameras, we draw two virtual lines and calculate their point of intersection using the following equations:

$$x\_intersect = \frac{((c1_y - c1_x * \tan(c1\_t)) - (c2_y - c2_x * \tan(c2\_t)))}{(\tan(c1\_t) - \tan(c2\_t))} \tag{2}$$

$$y\_intersect = (x\_intersect * tan(c1\_t)) + (c1_y - c1_x * tan(c1\_t)) \tag{3}$$

where $c1_x, c1_y$ and $c2_x, c2_y$ are the coordinates of the cameras in the top view, and $c1\_t$ and $c2\_t$ are the angle towards the target from both cameras.

The resulting $x\_intersect$ and $y\_intersect$ are the target coordinates in the top view. Iterating for each frame, plotting the resulting points, and drawing a line between the points gives us the mapping of the trajectory of the target.

G. Unscented Kalman Filter

In the second experiment, we use UKF to predict the target's position. Firstly, we track the location of the target in a top view coordinate space, getting its x and y coordinates and their respective velocity, namely $\dot{x}$ and $\dot{y}$, giving us the following state variable:

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} \end{bmatrix}^\mathsf{T} \tag{4}$$

We set the state variable's starting condition with the target's starting position and velocity to 1. We define the vector u to keep track of the target direction and the distance moved from the last point recorded.

$$\mathbf{u} = \begin{bmatrix} \Delta d & \alpha \end{bmatrix}^\mathsf{T} \tag{5}$$

The u vector aids in predicting the next state more accurately. The state transition function predicts the new position of the target. Due to the non-linearity system assumption, the state transition is a function instead of a matrix. To estimate the new coordinates, we take the current x and y and add, respectively, Δx and Δy, having the following functions:

$$\Delta x = \cos({U_a}^\mathsf{T}) * {U_{\Delta d}}^\mathsf{T} \tag{6}$$

$$\Delta y = \sin({U_a}^\mathsf{T}) * {U_{\Delta d}}^\mathsf{T} \tag{7}$$

Our sensors return the position of the target, calculated using triangulation. Therefore, our measurement function needs to return our prediction (state variable) in the form of a position (measurement space). Our state variable has a 4x1 shape, we return position 0 and 2 (x and y respectively), meaning that our measurement function looks like $\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$. The state co-variance indicates the trust we have in the measurement. Our initialization for the state co-variance matrix indicates how much we trust the initial state.

VI. Results of Experiments

In this section, we present the results of our two experiments. In Experiment 1, we compare our results of trajectory mapping using monocular depth perception (single-camera) approach against the triangulation (multi-camera)approach using video number 4 with zigzag pattern as shown in fig. 4 and fig. 5. Multi-camera setup uses cameras 1 and 2, while the single camera setup uses only camera 1. The single-camera setup (4) plots two lines; the blue line is the full trace, based on all measurement points (30fps), and the red line is a sample line (6fps) (more realistic, as discussed in Section VII). The single-camera trajectory mapping results in fig. 4 is achieved

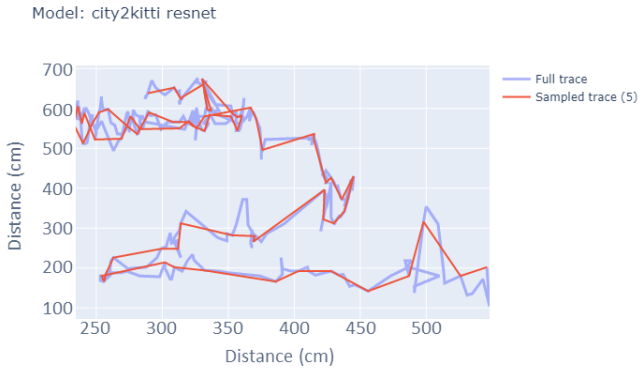

Fig. 4. Single-camera trajectory mapping using monocular depth perception.

using the *Citty 2 Kitti resnet* model. It is the best-performing model in terms of positioning compared to the other models, leading us to choose it for comparison against triangulation. In the results of the multi-camera trajectory mapping (fig. 5), we notice a sudden spike. The anomalous behavior of the spike and its implications have been addressed in section VIII. Experiment 2 predicts the target's trajectory using UKF and compares it to the baseline approach (triangulation). Figure 6 displays the results of the data fusion experiment executed in video 4, where the target walked in a zigzag pattern. As

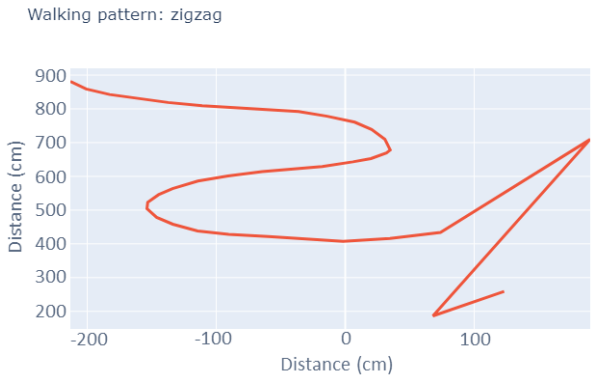

Fig. 5. Multi-camera trajectory mapping using triangulation.

we can see, the baseline and UKF are very similar. The blue hue indicates the uncertainty (co-variance) of the predicted position. Inspecting such uncertainty, we see that the hue shrinks on relatively straight paths, and when the direction changes, the hue grows. As expected, we also see that the uncertainty is high at the start because the algorithm needs time to converge. Moreover, we see a big spike around x=75 and y=425. This spike occurs when the target is in line with the two cameras. At that moment, the angle towards the target from one camera could overshoot the angle towards the target from the other camera resulting in a wrong measurement. We discuss the impact on the validity of this issue in VIII. However, we can see that when we reach this spike, the uncertainty also spikes.

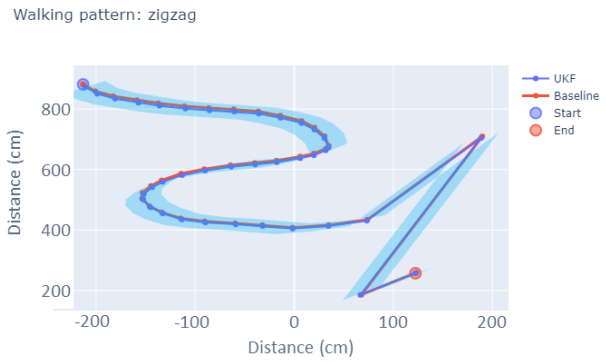

Fig. 6. Result of experiment 2 on video number 4.

VII. Discussion

In this section, we discuss the results of Experiment 1 and Experiment 2. We show the visual validation, Performance analysis and the answers to the research questions for each of them respectively.

A. Experiment 1

We compare the single and multi-camera setup results based on visual validation. The results in fig. 4 and fig. 5 are based on video 4, where the subject walks in a zigzag pattern. Both results show this pattern. The multi-camera setup shows a much smoother result than the sample single-camera result. The sudden spike in the multi-camera setup is an incorrect measurement discussed in section VIII. The single-camera setup does not have the anomalous spike as it only uses one camera. If we look at the y-axis of both results, we see that the single-camera setup, differently from the multi-camera, struggles to distinguish the depth after about 600cm. We judge the trajectory mapping performance of the single-camera setup based on the sampled trace because the full trace (at 30fps) takes too much time to compute. In table IV we have an overview of the processing time duration for all different models. The time is based on the sampled rate (the process calculates the depth every five frames), not the full trace. We can see that the processing time is always higher

TABLE IV
Results for processing duration of monocular depth perception.

Model name	Video Number	Video Duration (seconds)	Processing Duration (seconds)
City Scapes	4	9	15.2
Eigen	4	9	15.2
Kitti	4	9	15.2
City 2 Eigen	4	9	15.4
City 2 Kitti	4	9	15.2
City Scapes Resnet	4	9	23.2
Eigen Resnet	4	9	23.2
Kitti Resnet	4	9	24.2
City 2 Eigen Resnet	4	9	23.4
City 2 Kitti Resnet	4	9	23.2

than the video duration, even at the sample rate. A single frame takes about 0.30 seconds to process with a normal network and 0.43 with resnet. Hence, the maximum number of frames we can process each second is about 2-3, depending on the network. We evaluate the single-camera setup on the sampled trace with a sample rate of 5 (6fps). The experiment is executed using real video data. We compared two scenarios, one with two cameras combined using *Triangulation* to map the trajectory and one with a single camera combined using *monocular depth perception* to map the trajectory. Comparing them, we noticed that the trajectory mapping created using Triangulation is much smoother than the mapping created using monocular depth perception. We also noted that after about 600cm, the single-camera setup had trouble distinguishing the depth. However, in both cases, the pattern of the target is distinguishable. Furthermore, we calculated the time it takes to compute the depth of the video. The experiment proves that the multi-sensor data fusion approach outperforms the single-sensor system in trajectory mapping. The multi-sensor approach is much more accurate and stable than the single-sensor approach. On top of that, the maximum depth the monocular neural network can determine is limited to about 6 meters. Lastly, even if the maximum depth of the network would increase, we are left with high computation time.

B. Experiment 2

The visual validation and the difference between the baseline and UKF predictions judge the performance of UFK. Looking at the graph in fig. 6, we see that the baseline and UKF both match the zigzag pattern of the target. The baseline and UKF are also very similar. Therefore, we calculate the difference between the baseline and UKF. Table V shows the overview of the results for each video. We calculate the total difference (Total Diff. (cm)) for the entire video, the minimum and maximum difference (Min/Max/Avg Diff. (cm)) for any single point, the average difference for all points, and the difference between the last positions (Diff. at last position (cm)) given by the respective algorithms. The total difference

TABLE V
Experiment 2 results for each video.

Test Number	Video Number	Total Diff. (cm)	Min/Max/Avg Diff. (cm)	Diff. at last position (cm)
0	1	121.2	3.2/7.7/5.8	7.7
1	2	367.5	2.7/12.0/5.3	4.1
2	3	213.5	2.8/9.1/5.2	6.8
3	4	209.8	1.8/9.0/5.5	2.2

between UKF and the baseline can become relatively high as shown in table V. However, the maximal distance between two points for any video at any time is 12cm, and the average never goes above 5.8 in any video.

We executed the experiment using multiple videos with different walking patterns. We compared the baseline against the UKF algorithm for every scenario. We visually validated that the methods correctly display the target pattern for each scenario. We also calculated the difference between the baseline and UKF algorithm. We found that the average difference between the prediction and actual position is around 5.5cm. Although the total difference can get relatively high (e.g., 367.5cm in test 1), we see how the algorithm is still close to the baseline during the prediction and at the end of it.

As a result, the experimentation answers our second research question: *UKF can correctly predict the target's trajectory and stay very close to the actual trajectory.*

VIII. Threats to Validity

In Experiment 1 and Experiment 2, we used triangulation for trajectory mapping as a comparison technique for our proposed methods. However, we noticed a *spike* in the result obtained from our experiments as shown in fig. 7. In this subsection, we explain the reason for the spike and how it influences the validity of both experiments.

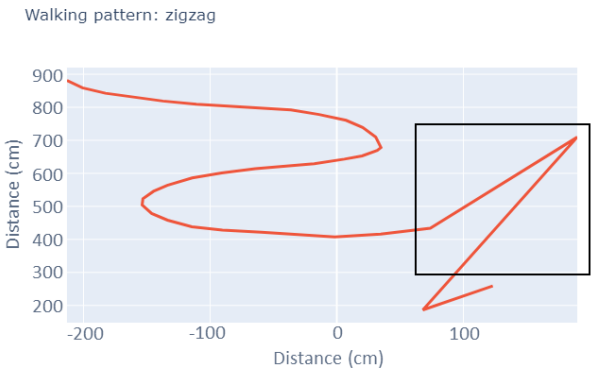

Fig. 7. Results of Experiment 1:'spike' highlighted in the black box.

The spike occurs when the target is in line with both cameras. We determined the position of the target by tracking the target from both cameras, taking the x-coordinate of the centroid of the tracking box around the target, calculating the angle towards the target from both cameras and then using those to draw virtual lines to see where they cross, as explained in Section V. As the tracking of cameras does not bring accurate and stable results, and the position of the cameras is only an approximation, we grouped angles and averaged them out to compensate this approximation. However, still in some circumstances the angle toward the target from both cameras overshoot and miss each other, as shown in fig. 8.

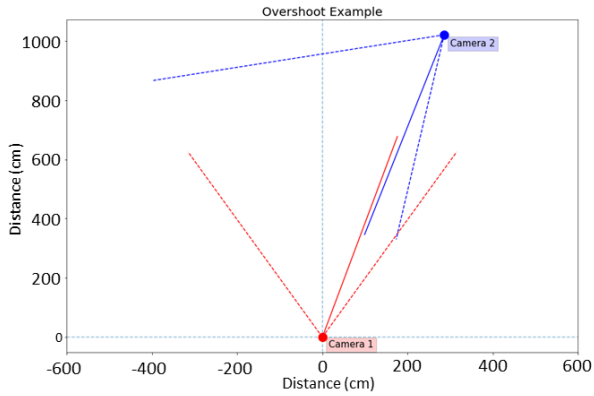

Fig. 8. Example of overshooting angles.

The black box in fig. 7 represents the angles overshooting. The lines cross but at the wrong coordinates, creating the spike. This situation does not pose any threat to the validity of our results. In the first experiment, we compared the mapped trajectory of the multi-sensor and single-sensor approaches; the spike only occurs in the multi-sensor setup when the target is in line with the cameras, and the rest of the graph is still helpful for comparison against the single sensor setup. In the second experiment, we compared our method using UKF against the one with triangulation. UKF can perfectly cope with sudden changes and can predict the trajectory of a target even with very rough non-linearity, only at the point of the spike, the uncertainty rises because there is no trust in its prediction. Although the spike does not pose any validity threats, we still proposed two possible corrections to this situation. Firstly, we corrected the spike by taking a range of coordinates instead of a single coordinate in both experiments. Secondly, we calculated where the two ranges instead of where two lines hit and took the median coordinate in our second experiment by altering the UKF algorithm and adding restrictions based on the boundaries of *human behavior*. Hence, instead of abandoning UKF predictions and bringing uncertainty to our results (like the behavior in our second experiment), we added some constraints (such as maximum speed per second) to the model to avoid anomalous behaviors.

IX. Conclusion

Currently, the third generation of surveillance systems takes advantage of cloud computing, affordable sensors, and advanced algorithms. However, there is a lack of a general architecture framework for these systems. In this paper, we proposed our architecture framework for big data analysis for multi-sensor surveillance systems, defining four layers from physical sensors to the user analytic monitoring, with the aim of designing trajectory prediction of a target within a multi-sensor surveillance system. Consequently, we defined two research questions in the field of surveillance systems and two respective experiments to answer them. The first experiment compared a multi-camera data fusion approach using triangulation against a single-camera approach using monocular depth perception on their ability to map the trajectory of a target. The second experiment used UKF as a data fusion approach to predict a target's trajectory and compared to our baseline with triangulation. To create the data for the experiments, we recorded real videos outside in a backyard. We placed two cameras across from each other so that they monitored the same scene and asked our target to perform several different walking patterns. We concluded that *the multi-camera approach works best as it has a smoother trajectory and does not have computation time limitations or depth perception limitations.* and *UKF is suitable to predict the trajectory of our target and stays close to our baseline over time.*

X. Limitations and Future Work

This research did not consider the multi-target multi-tracking problem. In reality, multiple targets need to be tracked simultaneously and the system should be able to keep track of

them all. Therefore, we would continue to address this problem in our research on trajectory prediction for surveillance systems. Furthermore, we would consider designing an object recognition method related to the location itself. For instance, a surveillance system for a train station, should recognize objects such as people, bags, suitcases and trains in that location. Moreover, we assumed that the approximate location of the sensors is known and fixed, though mobile cameras (e.g., drones, smart glasses) with no fixed location could improve the position measurement and consequently the prediction should be considered as well. Finally, another line of research will consider *multidimensional paradigms* (e.g., [36]) and *hidden relationships* (e.g., [37]) as relevant extensions of our actual framework.

ACKNOWLEDGMENT

The authors acknowledge the work of Tom van der Wielen for his invaluable work in bootstrapping this research endeavor. This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

REFERENCES

[1] T. D. Räty, "Survey on contemporary remote surveillance systems for public safety," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 40, no. 5, pp. 493–515, 2010.

[2] R. A. Ariyaluran Habeeb, F. Nasaruddin, A. Gani, I. A. Targio Hashem, E. Ahmed, and M. Imran, "Real-time big data processing for anomaly detection: A Survey," *International Journal of Information Management*, vol. 45, no. February, pp. 289–307, 2019.

[3] C. S. Regazzoni, V. Ramesh, and G. L. Foresti, "Special issue on video communications, processing, and understanding for third generation surveillance systems," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1355–1367, 2001.

[4] G. Cascavilla, D. A. Tamburri, F. Leotta, M. Mecella, and W. Van Den Heuvel, "Counter-terrorism in cyber–physical spaces: Best practices and technologies from the state of the art," *Information and Software Technology*, vol. 161, p. 107260, 2023.

[5] D. De Pascale, M. Sangiovanni, G. Cascavilla, D. A. Tamburri, and W.-J. Van Den Heuvel, "Securing cyber-physical spaces with hybrid analytics: Vision and reference architecture," in *Computer Security. ESORICS 2022 International Workshops*, 2023, pp. 398–408.

[6] D. De Pascale, G. Cascavilla, M. Sangiovanni, D. A. Tamburri, and W.-J. van den Heuvel, "Internet-of-things architectures for secure cyber–physical spaces: The visor experience report," *Journal of Software: Evolution and Process*, vol. 35, no. 7, p. e2511, 2023.

[7] X. Wang, "Intelligent multi-camera video surveillance: A review," *Pattern Recognition Letters*, vol. 34, no. 1, pp. 3–19, 2012.

[8] R. Langone, A. Cuzzocrea, and N. Skantzos, "Interpretable anomaly prediction: Predicting anomalous behavior in industry 4.0 settings via regularized logistic regression tools," *Data Knowl. Eng.*, vol. 130, p. 101850, 2020.

[9] A. Cuzzocrea, "Retrieving accurate estimates to OLAP queries over uncertain and imprecise multidimensional data streams," in *Scientific and Statistical Database Management - 23rd International Conference, SSDBM*, ser. Lecture Notes in Computer Science, vol. 6809. Springer, 2011, pp. 575–576.

[10] A. Cuzzocrea, C. K. Leung, and R. K. MacKinnon, "Mining constrained frequent itemsets from distributed uncertain data," *Future Gener. Comput. Syst.*, vol. 37, pp. 117–126, 2014.

[11] E. G. Rieffel, A. Girgensohn, D. Kimber, T. Chen, and Q. Liu, "Geometric tools for multicamera surveillance systems," *2007 1st ACM/IEEE International Conference on Distributed Smart Cameras, ICDSC*, pp. 132–139, 2007.

[12] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43, p. 46, 2004.

[13] D. Simon, *Optimal state estimation: Kalman, H Infinity, and nonlinear approaches*, 2006.

[14] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," *Signal Processing, Sensor Fusion, and Target Recognition VI*, vol. 3068, p. 182, 1997.

[15] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," *IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium, AS-SPCC 2000*, pp. 153–158, 2000.

[16] Replication-Package, "Designing multi-sensorial information systems: a dual case-study," https://doi.org/10.5281/zenodo.8308946, 2023.

[17] H. N. Hu, Q. Z. Cai, D. Wang, J. Lin, M. Sun, P. Kraehenbuehl, T. Darrell, and F. Yu, "Joint monocular 3D vehicle detection and tracking," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-October, pp. 5389–5398, 2019.

[18] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan, "Accurate monocular 3D object detection via color-embedded 3D reconstruction for autonomous driving," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-October, pp. 6850–6859, 2019.

[19] J. M. U. Vianney, S. Aich, and B. Liu, "RefinedMPL: Refined monocular PseudoLiDAR for 3D object detection in autonomous driving," *arXiv*, 2019.

[20] X. Weng and K. Kitani, "Monocular 3D object detection with pseudo-LiDAR point cloud," *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pp. 857–866, 2019.

[21] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 6602–6611, 2017.

[22] M. A. Bakr and S. Lee, "Distributed multisensor data fusion under unknown correlation and data inconsistency," *Sensors (Switzerland)*, vol. 17, no. 11, 2017.

[23] Y. Bar-Shalom and X.-R. Li, "Estimation and tracking- principles, techniques, and software," *Norwood, MA: Artech House*, 1993.

[24] T. J. Ellis and J. Black, "A multi-view surveillance system," *IEE Colloquium (Digest)*, vol. 3-10062, pp. 59–63, 2003.

[25] J. Black and T. Ellis, "Multi camera image tracking," *Image and Vision Computing*, vol. 24, no. 11, pp. 1256–1267, 2006.

[26] A. Chilgunde, P. Kumar, S. Ranganath, and H. Weimin, "Multi-Camera Target Tracking in Blind Regions of Cameras with Non-overlapping Fields of View," pp. 42.1–42.10, 2012.

[27] J. Howse, *OpenCV Computer Vision with Python*.

[28] A. Brdjanin, N. Dardagan, D. Dzigal, and A. Akagic, "Single Object Trackers in OpenCV: A Benchmark," *INISTA 2020 - 2020 International Conference on INnovations in Intelligent SysTems and Applications, Proceedings*, 2020.

[29] T. W. Mi and M. T. Yang, "Comparison of tracking techniques on 360-degree videos," *Applied Sciences (Switzerland)*, vol. 9, no. 16, 2019.

[30] A. Lukežič, T. Vojíř, L. Čehovin Zajc, J. Matas, and M. Kristan, "Discriminative Correlation Filter Tracker with Channel and Spatial Reliability," *International Journal of Computer Vision*, vol. 126, no. 7, pp. 671–688, 2017.

[31] P. Lytrivis, A. Amditis, and G. Thomaidis, "Sensor data fusion in automotive applications," 2009.

[32] A. Steinberg and C. Bowman, "Revisions to the JDL Data Fusion Model," 2001.

[33] J. Llinas, C. Bowman, G. Rogova, A. Steinberg, E. Waltz, and F. White, "Revisiting the JDL data fusion model II," *Proceedings of the Seventh International Conference on Information Fusion, FUSION 2004*, vol. 2, pp. 1218–1230, 2004.

[34] Appendix, "Designing multi-sensorial information systems: a dual case-study," https://doi.org/10.6084/m9.figshare.24073308, 2023.

[35] K. Farkhodov, S.-H. Lee, and K.-R. Kwon, "Object tracking using csrt tracker and rcnn." in *BIOIMAGING*, 2020, pp. 209–212.

[36] A. Cuzzocrea and P. Serafino, "*LCS*-hist: taming massive high-dimensional data cube compression," in *EDBT 2009, 12th International Conference on Extending Database Technology*, ser. ACM, vol. 360. ACM, 2009, pp. 768–779.

[37] Z. Wu, W. Yin, J. Cao, G. Xu, and A. Cuzzocrea, "Community detection in multi-relational social networks," in *Web Information Systems Engineering - WISE 2013 - 14th International Conference*, ser. Lecture Notes in Computer Science, vol. 8181. Springer, 2013, pp. 43–56.