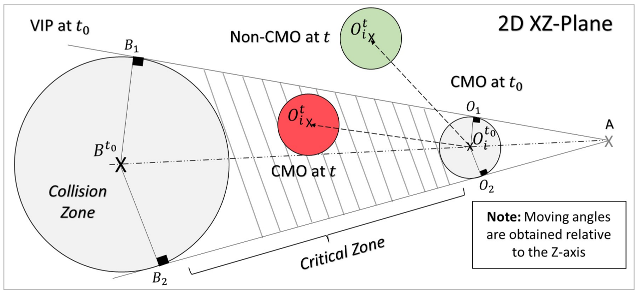

Figure A.1. Collision Prediction and CMOs Identification.

A. Collision Prediction and CMO Identification

This section explains our collision prediction method to identify CMOs. Integrating any other collision prediction algorithm with our proposed path planning algorithm is straightforward.

Our collision prediction method is a geometric-based algorithm inspired by [12]. The assumptions we make are similar to the ones we assumed in Section 3.1 regarding the circular shape and movement of the CMO and VIP, except that, to identify CMOs, we assume that the VIP's circle $'C_B$ is larger than the CMO's circle $'C_{O_i}$, which means the $r_B >> r_{O_i}$, as it is shown in Fig. A.1. This assumption is necessary for VIP safety, in particular, to ensure that all possible threats are detected. Similar to collision avoidance described in Section 3.2, the *critical zone* is defined with the tangents $\overleftrightarrow{B_1O_1}$ and $\overleftrightarrow{B_2O_2}$ of the circles $'C_B$ and $'C_{O_i}$. Then, we calculate three angels as follows:

- The angle $\angle\sigma$ between $\overline{AB^{t_0}}$ and $\overline{AB_1}$, which can be calculated as:

$$\sigma = tan^{-1}\left(\frac{1}{\sqrt{\frac{'d^2}{(r_B - r_{O_i})^2} - 1}}\right), \tag{18}$$

  where $'d$ is the Euclidean distance between the VIP and the object at t_0, which can be expressed as:

$$'d = \sqrt{(X_B^{t_0} - X_{O_i}^{t_0})^2 + (Z_B^{t_0} - Z_{O_i}^{t_0})^2} \tag{19}$$

  Note that $Y_B^{t_i}$ and $Y_{O_i}^{t_i}$ are ignored (*i.e.,* equal to 0) since we assume the camera motion at the Y-axis is negligible as it is vertical to the ground plane.

- The angle $\angle\theta$ of the displacement vector from $O_i^{t_0}$ to B^{t_0}, which can be expressed as:

$$\theta = tan^{-1}\left(\frac{X_B^{t_0} - X_{O_i}^{t_0}}{Z_B^{t_0} - Z_{O_i}^{t_0}}\right), \tag{20}$$

- The angle $\angle\rho$ of the displacement vector from $O_i^{t_0}$ to O_i^t, which can be calculated as:

$$\rho = tan^{-1}\left(\frac{X_{O_i}^{t} - X_{O_i}^{t_0}}{Z_{O_i}^{t} - Z_{O_i}^{t_0}}\right), \tag{21}$$

Based on angles σ, ρ, and θ, if $\theta - \sigma \leq \rho \leq \theta + \sigma$, means the object is moving inside the critical zone and may threaten the VIP; thus it is identified as a CMO. Otherwise, it is a non-CMO object and we ignore it.

B. Derivation of the α angle

We calculate the angle $\angle\alpha$ between $\overline{AO_i^{t_1}}$ and $\overline{AO_1}$ as shown in Fig. 2. Obviously, since $\triangle AO_i^{t_1}O_1$ and $\triangle AB^tB_1$ are similar, there is:

$$\frac{|\overline{AO_i^{t_1}}|}{|\overline{AB^t}|} = \frac{r_{O_i}}{r_B}, \tag{22}$$

$$\frac{|\overline{AO_i^{t_1}}|}{|\overline{AB^t}|} = \frac{|\overline{(AB^t + B^tO_i^{t_1})}|}{|\overline{AB^t}|} = \frac{|\overline{AB^t}| + d}{|\overline{AB^t}|} = \frac{r_{O_i}}{r_B}, \tag{23}$$

thus,

$$|\overline{AB^t}| = \frac{r_B}{r_{O_i} - r_B}d, \tag{24}$$

From the Pythagoras equation, there is:

$$|\overline{AB_1}| = \sqrt{|\overline{AB^t}|^2 - {r_B}^2}. \tag{25}$$

Inserting Eq. (24) into Eq. (25), there is:

$$|\overline{AB_1}| = r_B\sqrt{\frac{d^2}{(r_{O_i} - r_B)^2} - 1}. \tag{26}$$

For α in $\triangle AB^tB_1$, there is:

$$tan(\alpha) = \frac{r_B}{|\overline{AB_1}|} = \frac{1}{\sqrt{\frac{d^2}{(r_{O_i} - r_B)^2} - 1}}, \tag{27}$$

thus,

$$\alpha = tan^{-1}\left(\frac{1}{\sqrt{\frac{d^2}{(r_{O_i} - r_B)^2} - 1}}\right). \tag{28}$$

C. Implantation

Fig. C.3 shows the prototype's general architecture, which consists of four modules: *Vision Module* (VM), *Objection Detection and Tracking* (ODT), *CMO Identification* (CMOI), and *CMO Status Estimation* (CMOSE).

The VM captures 640x480 resolution videos at 30 frames per second using an RGB camera (with $50mm \times 28mm \times 0.9mm$ size and $5g$ weight) mounted on the VIP's

Figure C.2. Our Real-World Prototype.

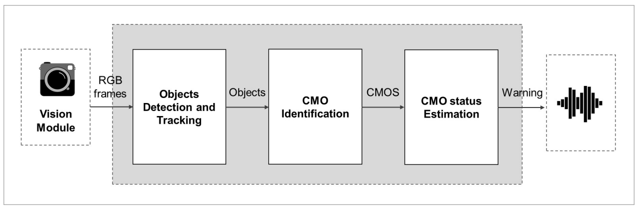

Figure C.3. The General Architecture of Our Prototype.

chest. The camera transmits videos, through a USB cable, to a laptop (in the VIP's backpack) with an Intel Core i9-9900K CPU, an NVIDIA GeForce RTX 2080 GPU, and 64GB RAM. The ODT module detects and tracks moving objects from the VM video using three modules: pre-trained CNN-based algorithms, *i.e.,* YOLOv3 [28], and Deep Simple Online Real-Time (Deep SORT)[32], in which objects' trajectories and classes are outputs, and our motion detection algorithm that identifies moving objects and compensates camera motion for accurate estimations.

The CMOI module identifies the CMOs using our collision prediction method described in Appendix A. CMOSE receives a CMO's trajectory to estimate the distance (in m) from the CMO to the VIP, using a pre-trained DisNet model [15]. It can also estimate the CMO's speed using the following equation:

$$|v_O^t| = \sqrt{(\frac{X_O^t - X_O^{t_0}}{t - t_0})^2 + (\frac{Y_O^t - Y_O^{t_0}}{t - t_0})^2 + (\frac{Z_O^t - Z_O^{t_0}}{t - t_0})^2}, \tag{29}$$

For accurate estimation, a 2D-to-3D transformation is applied to recover 3D CMO's metric position in the Camera Coordinate System[3] (CCS), using the Pinhole Model [14]. Based on the CMO's estimated distance and speed, CMOSE calculates the estimated collision time (in s) as follows:

$$'t_{cl}^t = \frac{Z_O^t}{|v_O^t|}. \tag{30}$$

The *MinD* scheme, and all the above, are implemented in Python 3.7.0. For Yolov3 and Deep Sort, we used public implementation[4] based on the TensorFlow-gpu 2.1.0.

For conducting our experiments [5], we first set up an experimental environment by identifying the radius of $'C_B$ and $'C_{O_i}$ and the critical zone (described in Appendix A) on the ground with ropes based on each object class. Objects were located at different distances (up to $100m$) and moving at varying speeds (up to $60km/h$) toward the person mimicking the VIP while holding our prototype to capture videos (*i.e.,* 140 videos with different lengths up to $32min$). Note that, for safety considerations, the person only held the prototype in scenarios involving slowly moving objects; otherwise, the vision module was installed on the ground. For analysis purposes, we manually identify the ground truth (GT) of CMOs (*i.e.,* moving inside the critical zone), CMOs' relative initial-distances to the VIP, and CMOs' speeds (*i.e.,* for cars and motorcycles, we use their own speedometers; for bicycles and pedestrians, we measured the time taken to travel a predefined distance, then calculate the speed by dividing the distance by the time taken). Note that, under each speed, the object was moving at a uniform fixed speed only to get accurate measurements (*i.e.*, GT) as advanced precise speed measurement equipment was unavailable. We evaluate the prototype when the camera was stationary and slowly moving toward objects.

Our prototype's evaluation results reveal its precise identification of CMOs, achieving an overall mean Average Precision (mAP) of 97.19% and an overall mean Average Recall (mAR) of 89.23%. Our prototype also accurately estimates VIP-CMOs distance, with an overall absolute error of only $-0.75m$ (note, the negative sign indicating that the estimated value is less than the actual value, and vice versa). This error is negligible compared to the overall average of GT distances, which is $22.57m$. Furthermore, the prototype reliably estimates CMOs' speed, with an overall average maximum error of $3.89km/h$ (the overall average of GT speeds is $18.92km/h$). The evaluation also indicates that the prototype successfully predicts CMOs' average collision times, achieving an absolute error of $-2.83s$ (the average of GT collision times is $11.05s$).

In summary, our prototype was able to accurately identify, estimate the distance to, and predict the collision time of CMOs. This enables us to build a reliable empirically parameterized simulator (as possible) to effectively evaluate our path planning algorithm without compromising health and safety, as mentioned in Section 4.

[3]It is a real-world 3D Cartesian coordinate system with the camera as the origin (0,0,0).

[4]https://github.com/theAIGuysCode/yolov3_deepsort

[5]Note that this study is approved by the Research Governance and Integrity Team in the same authors' institution.