Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-34 (2019) 176–183

# Video-guided Camera Control for Target Tracking and Following ⋆

**Jake Gemerek** * **Silvia Ferrari** * **Brian H. Wang** *
**Mark E. Campbell** *

* *Department of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY, USA (e-mail: (jrg362, ferrari, bhw45, mc288)@cornell.edu)*

**Abstract:** This paper considers the problem of controlling a nonholonomic mobile ground robot equipped with an onboard camera characterized by a bounded field-of-view, tasked with detecting and following a potentially moving human target using onboard computing and video processing in real time. Computer vision algorithms have been recently shown highly effective at object detection and classification in images obtained by vision sensors. Existing methods typically assume a stationary camera and/or use pre-recorded image sequences that do not provide a causal relationship with future images. The control method developed in this paper seeks to improve the performance of the computer vision algorithms, by planning the robot/camera trajectory relative to the moving target based on the desired size and position of the target in the image plane, without the need to estimate the target's range. The method is tested and validated using a highly realistic and interactive game programming environment, known as Unreal Engine™, that allows for closed-loop simulations of the robot-camera system. Results are further validated through physical experiments using a Clearpath™ Jackal robot equipped with a camera which is capable of following a human target for long time periods. Both simulation and experimental results show that the proposed vision-based controller is capable of stabilizing the target object size and position in the image plane for extended periods of time.



*Keywords:* Robot vision, Tracking applications, Robot navigation, Robot control

## 1. INTRODUCTION

Recent advancements in computer vision, particularly video-based object detection and classification, pave the way for future autonomous systems comprised of camera-equipped mobile robots that can decide how to obtain and process images or videos without human intervention (Wei et al. (2014)). The development of autonomous mobile cameras have recently been shown to impact a variety of applications that include automated surveillance, (Esterle et al. (2017)), intelligent cinematography (Nageli et al. (2017)), and autonomous social navigation (Chen et al. (2017)). The challenge of detecting, tracking, and following a mobile human target of interest is critical to all of the aforementioned applications. As a result, several human tracking algorithms have been developed, some of which make use of carefully designed hand-crafted features, such as Histograms of Oriented Gradients (HoG) for detection with simultaneous KLT (Kanade-Lucas-Tomasi) feature-tracking (Benfold and Reid (2011)), and optical flow-based human tracking methods using multiple cameras (Tsutsui et al. (2001)). More recent human tracking algorithms take advantage of the advancements of deep convolutional networks, and instead use convolutional features for tracking human appearance in videos (McLaughlin et al. (2017)). Human tracking in video has become one of the challenging problems at the forefront of computer vision, leading to the development of benchmark datasets (Leal-Taixé et al. (2015)). The state-of-the-art methods according to such benchmarks base their hypotheses on multiple cues, such as appearance, kinematics, and interactions (Sadeghian et al. (2017)).

Although the problem of human detection and tracking has been studied extensively over recent years, almost all of the tracking algorithms do not incorporate any type of control over the camera field of view (FoV), and instead assume the video sequence is recorded *a priori*. The few works that do actively control the camera FoV simplify the detection and tracking of the human target (Goldhoorn et al. (2014), Nageli et al. (2017)). This paper considers the problem of controlling a mobile camera with a bounded FoV that is rigidly attached to a mobile robot capable of onboard computing for real-time video processing. The camera is controlled such that a target (human) of interest is detected, tracked, and followed while moving through a complex, unstructured environment. The novel approach in this paper leverages a state-of-the-art deep learning algorithm for detecting a human target in the video (Liu et al. (2016)), whose output is processed to compute a control input command for the mobile robot without requiring a 3-dimensional position estimate or kinematic model of the target. This is advantageous since human motion is generally very difficult, if not impossible, to accurately predict.

⋆ This research is funded by the Office of Naval Research Grant N00014-17-1-2175.


10.1016/j.ifacol.2019.01.062

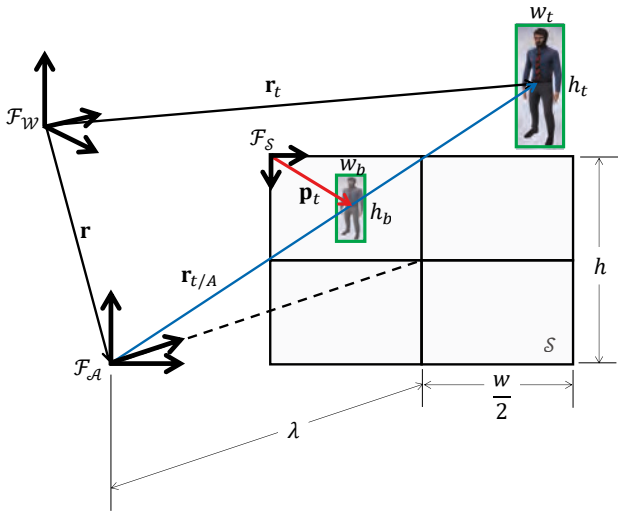


Fig. 1. The inertial, body, and image reference frames are illustrated, along with relative position vectors and dimensions. The target is shown in the 3-dimensional world being transformed, via perspective projection, onto the image plane.

This paper also proposes the use of a game development software, known as Unreal Engine™, for simulation of the vision-based control algorithm in a photo-realistic virtual space. Most robot simulation and visualization software emphasize physical accuracy and lack visual realism. Furthermore, simulation in a visually-realistic environment provides the ability to quickly and efficiently recreate and redefine an infinite set of testing conditions, while providing a deterministically repeatable test environment. This work further validates the proposed controller as well as the reliability of the Unreal Engine™as a control algorithm simulator through physical experiments which demonstrate the proposed methods are capable of real-world implementation.

## 2. PROBLEM FORMULATION

Consider a region of interest, $\mathcal{W} \subset \mathbb{R}^3$, populated with human target $\mathcal{T} \subset \mathcal{W}$, and a mobile robot $\mathcal{A} \subset \mathcal{W}$. The robot is equipped with a camera which has a bounded FoV, characterized by a focal length $\lambda \in \mathbb{R}$, a half-angle $\alpha \in \mathbb{S}^1$, and an aspect ratio $A_R$. The image plane, $\mathcal{S} = [0, w] \times [0, h]$, is the perspective projection of $\mathcal{W}$ as seen through the camera FoV, where $w$ and $h$ are the width and height of the image, respectively, which may be computed from the camera parameters $\lambda$, $\alpha$, and $A_R$, that is $w = 2\lambda \tan\alpha$, and $h = w/A_R$ (Fig. 1).

A reference frame $\mathcal{F}_\mathcal{A}$ is embedded in $\mathcal{A}$, such that the 1st axis is aligned with the camera optical axis, the 3rd axis points vertically, and the 2nd axis completes the right-hand rule. The origin of $\mathcal{F}_\mathcal{A}$ is known as the focal point of the camera, whose position $\mathbf{r}(t) \in \mathbb{R}^3$ is expressed relative to an inertial reference frame $\mathcal{F}_\mathcal{W}$, as illustrated in Fig. 1. The coordinates of the focal point position, expressed in $\mathcal{F}_\mathcal{W}$ are given as $\mathbf{r}(t) = [x(t), y(t), z]^T$, where $z$ is the constant height of the focal point. Assuming the $z$-axes of $\mathcal{F}_\mathcal{W}$ and $\mathcal{F}_\mathcal{A}$ remain parallel, the rotation matrix which maps vectors expressed in $\mathcal{F}_\mathcal{W}$ to $\mathcal{F}_\mathcal{A}$ is given as

$$\mathbf{R}_W^A = \begin{bmatrix} \cos\theta(t) & \sin\theta(t) & 0 \\ -\sin\theta(t) & \cos\theta(t) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

where $\theta(t)$ is the yaw angle of the robot.

The perspective projection $\mathbf{P}_A^S$ maps vectors in $\mathbb{R}^3$ expressed with respect to $\mathcal{F}_\mathcal{A}$, whose origin is the focal point of the camera, to vectors in $\mathbb{R}^2$ expressed with respect to $\mathcal{F}_\mathcal{S}$. Fig. 1 shows the vector $\mathbf{r}_{t/A} \in \mathcal{W}$ being mapped to the vector $\mathbf{p}_t \in \mathcal{S}$ via $\mathbf{P}_A^S$. $\mathbf{p}(t) = [x_b(t), y_b(t)]^T$ is the position vector to the center of the bounding box in the image plane. The perspective projection can be written as a scaled linear operation on $\mathbf{r}_{t/A}$ using homogeneous coordinates (Sonka et al. (2014)). The perspective projection mapping of an arbitrary vector $\mathbf{r} \in \mathcal{W}$ to the associated image plane vector $\mathbf{p} \in \mathcal{S}$ is

$$\mathbf{P}_A^S(\mathbf{r}) = \frac{1}{2}\begin{bmatrix} w \\ b \\ 0 \end{bmatrix} - \gamma \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{r} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}, \tag{2}$$

where $\gamma \in \mathbb{R}$ is a scaling parameter used to enforce the 3rd element to equal unity. Therefore, all of the required transformations have been defined which take the target position $\mathbf{r}_t$, and the robot position $\mathbf{r}$, expressed in inertial frame, and define a position vector on the image plane $\mathbf{p}_t$. This complete transformation is illustrated in Fig. 1 and may be expressed as

$$\begin{bmatrix} \mathbf{p}_t \\ 1 \end{bmatrix} = \mathbf{P}_A^S \left( \mathbf{R}_W^A (\mathbf{r}_t - \mathbf{r}) \right). \tag{3}$$

Assuming $\mathcal{A}$ is rigid, and the camera is rigidly attached to $\mathcal{A}$, the robot configuration (state) vector can be described as $\mathbf{q}(t) = [x(t)\ \ y(t)\ \ \theta(t)]^T$, which is governed by the nonholonomic unicycle kinematic model,

$$\dot{\mathbf{q}}(t) = \begin{bmatrix} \cos\theta(t) & 0 \\ \sin\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \mathbf{G}(\mathbf{q}(t))\mathbf{u}(t), \tag{4}$$

where $v(t) \in \mathbb{R}$ is the forward velocity, and $\omega(t) \in \mathbb{R}$ is the yaw rate. The control input vector is defined as $\mathbf{u}(t) = [v(t)\ \ \omega(t)]^T \in \mathcal{U}$, and $\mathcal{U} \subset \mathbb{R}^2$ is the set of admissible control inputs.

A bounding box $\mathbf{b}(t) \in \mathbb{R}^4$ associated with the target $\mathcal{T}$ is extracted from the video and projected on $\mathcal{S}$. The elements composing $\mathbf{b}(t) = [x_b(t), y_b(t), w_b(t), h_b(t)]^T$ are the coordinates of the bounding box center $\mathbf{p}_t(t) = [x_b(t), y_b(t)]^T \in \mathcal{S}$, expressed in the image frame $\mathcal{F}_\mathcal{S}$, and the width $w_b(t) \in [0, w]$ and height $h_b(t) \in [0, h]$ of the bounding box. Several computer vision algorithms exist which extract such a bounding box containing an object of interest from an image, some of which will be reviewed in the following section.

### 2.1 Control Objective

In order to maintain the target within the camera FoV, a reliable target bounding box must be consistently extracted from the video sequence. Therefore, it is desirable to maintain the target not only within the FoV, but at a reliable range for accurate image processing. The control objective is to drive the bounding box $\mathbf{b}(t)$ to a desired set point $\tilde{\mathbf{b}}$ by suitable choice of the control input vector $\mathbf{u}(t)$ over some time interval of interest $[t_0, T) \subseteq \mathbb{R}$. That is,

$$[x_b, y_b, w_b, h_b]^T \to [\tilde{x}_b, \tilde{y}_b, \tilde{w}_b, \tilde{h}_b]^T. \tag{5}$$

where $\tilde{x}_b$, $\tilde{y}_b$ are the desired (constant) coordinates of the bounding box center, and $\tilde{w}_b$, $\tilde{h}_b$ are the desired (constant) width and height of the bounding box, respectively. Typically, the set point $\tilde{\mathbf{b}}$ is chosen such that the bounding box remains in the center of the image at a sufficient scale for reliable image processing. However, since there is no control over the pitch of the camera or the relative height of the camera focal point and the target center, $y_b(t)$ is not controllable. As long as the target and robot both remain on a level surface with a reasonable relative height between the camera focal point and the target center, this uncontrollable variable will not affect the proposed algorithm's performance. A more complex model may assume the camera is mounted on a gimbal, enabling the camera to rotate with respect to the robot, which would provide a means to control $y_b(t)$, but such a model is not considered in this work.

Similarly, the width $w_b(t)$ and height $h_b(t)$ of the bounding box provide some redundant information since the robot cannot control the orientation of the target. Therefore, the width and height are combined into a single size metric $\Delta_b(t)$ of the bounding box, which is a measure of the length of the bounding box diagonal, $\Delta_b(t) = \|[w_b(t), h_b(t)]^T\|_2$. Therefore, after post-processing the extracted bounding box $\mathbf{b}(t) = [x_b(t), y_b(t), w_b(t), h_b(t)]^T$ into a useful output vector $\mathbf{y}(t) = [\Delta_b(t), x_b(t)]^T$, the control objective is reduced to

$$\mathbf{y}(t) \to \tilde{\mathbf{y}}, \tag{6}$$

where $\tilde{\mathbf{y}} = [\tilde{\Delta}_b, \tilde{x}_b]^T$, and $\tilde{\Delta}_b = \|[\tilde{w}_b, \tilde{h}_b]^T\|_2$.

## 3. METHODOLOGY

This paper presents a novel unified video processing and control approach for detecting and pursuing a human target in a complex unstructured environment, which is accomplished by stabilizing the control objective defined in the previous section. Due to recent advancements of object recognition tasks in computer vision, the presented methodology employs a state-of-the-art deep convolutional neural network (CNN) Liu et al. (2016) to detect and classify objects within the image plane. The output of the CNN is then processed to extract a bounding box $\mathbf{b}(t)$ associated with the target. This bounding box is then used to compute a control law designed to maintain the target human in a desirable position within the image for reliable future detections.

### *3.1 Target Detection and Identification*

Over recent years algorithms for multi-class object detection in images have become extremely accurate, mostly due to the use of deep CNNs Huang et al. (2017)). Three recent well-known architectures are Faster R-CNN proposed by Ren et al. (2016), the Single Shot Multibox Detector (SSD) developed by Liu et al. (2016), and the Region-based Fully Convolutional Network (R-FCN) by Dai et al. (2016). It is difficult to disambiguate the *best* architecture due the use of interchangable feature extraction and classification techniques. Furthermore, the work by Huang et al. (2017) present a comprehensive study of the speed-accuracy tradeoff between different CNN architectures and feature extractors. The object detection algorithm used in this work is chosen to be as accurate as possible while simultaneously being capable of real-time implementation on a physical robot with onboard computing. This work uses a MobileNet (Howard et al. (2017)) implementation of the SSD architecture (Liu et al. (2016)) in order to satisfy real-time resource constraints. The CNN is pre-trained on the Microsoft COCO data set of Lin et al. (2014).

The CNN takes as input an image $\mathcal{S}(t)$ at time $t$ and outputs a set of detection-confidence pairs $\mathcal{B}(t) = \{(\mathbf{b}_i(t), c_i(t))\}_{i=1}^{N_{det}(t)}$, where $\mathbf{b}_i(t)$ are bounding boxes containing objects of the same class as the target (i.e., human), and $c_i(t) \in [0, 1)$ are the associated confidence scores of the bounding box, and $N_{det}(t) \in \mathcal{N}$ is the number of detections. The target bounding box $\mathbf{b}(t)$ is then computed as the bounding box with the highest associated confidence score, when multiple detections are extracted. If no detections are extracted, i.e., $\mathcal{B}(t) = \emptyset$, the target bounding box $\mathbf{b}(t)$ is set equal to the set point bounding box $\tilde{b}$ in order to stop the robot. This processing step is expressed as

$$\mathbf{b}(t) = \begin{cases} \arg\max_{\mathbf{b}_i} \{c_i : (\mathbf{b}_i, c_i) \in \mathcal{B}(t)\} & \mathcal{B}(t) \neq \emptyset \\ \tilde{\mathbf{b}} & \mathcal{B}(t) = \emptyset \end{cases}. \tag{7}$$

This formulation guarantees that $\mathbf{b}(t)$ exists and is unique by construction. The bounding box $\mathbf{b}(t)$ is then transformed into the output vector $\mathbf{y}(t)$ which is used as the control variable in the controller to be designed in the following subsection.

### *3.2 Video-guided Camera Control Design*

The control law for tracking and following the target based on video frames obtained by the robot camera, and processed according to the previous subsections, is developed by considering properties of the perspective projection, and noting how points in three dimensions move across an image in two-dimensions while the camera is moving. Due to the properties of the perspective projection, objects which are closer to the focal point appear larger, and objects that are farther from the focal point appear smaller. This provides a natural method for controlling the size of the bounding box $\Delta_b(t)$ without requiring kinematic estimations in the 3-dimensional world. Similarly, the position of the bounding box $x_b(t)$ in the image provides a natural error signal for the yaw rate $\omega(t)$ of the robot. Because the target human may be moving, the use of integral compensation is proposed in order to reduce steady state errors that would be present if the control input were simply proportional to these error signals. Then, the proposed video-guided control input is designed using the following proportional-integral compensation

$$\mathbf{u}(t) = -\mathbf{K}_1 \Delta\mathbf{y}(t) - \mathbf{K}_2 \int_0^t \Delta\mathbf{y}(\tau) d\tau, \tag{8}$$

where $\mathbf{K}_1, \mathbf{K}_2 \succeq 0$ are diagonal gain matrices of reasonable dimension, and $\Delta\mathbf{y}(t) = \mathbf{y}(t) - \tilde{\mathbf{y}}$. The proposed control law is validated using photo-realistic simulations in highly complex environments, as well as through physical experiments in a laboratory setting.

## 4. SIMULATION ENVIRONMENT

The Unreal Engine™is a leading game development software capable of advanced open-source environment development and manipulation. These capabilities have recently been exploited by several industries outside of the game development community, including architectural visualization, film-making, and virtual reality training simulations. The use of Unreal Engine™for simulation of computer vision-based automatic control algorithms makes no sacrifice to physical accuracy, but has the advantage of a vast user community composed of artists and developers who create visually realistic environments, characters, behaviors, and objects, which may be used in simulations. This ease of access to a diverse set of environments and scenarios helps test the robustness of proposed methods in ways not feasible in real-world experiments or conventional robotics software.

Similar game development softwares have been used in previous works for generating synthetic data to train deep CNNs, such as Johnson-Roberson et al. (2017). Unreal Engine™has also been used for similar computer vision tasks by several authors, such as semantic segmentation by Qui and Yuille (2016), and simulating stereo-vision applications by Zhang et al. (2016). Furthermore, the work by Shah et al. (2017) uses Unreal Engine™to develop realistic quadrotor and vehicle simulations for autonomous vehicle simulations. This work, proposes the use of Unreal Engine™for the novel task of simulating a fully autonomous robot using visual feedback for tracking a target in the realistic virtual space. The setup and results of these simulations are presented in the following section.

## 5. SIMULATION RESULTS

The control law developed in Section 3 is tested in the visually realistic and complex subway environment using Unreal Engine™, in which a mobile robot equipped with a camera tracks and follows a potentially moving human without knowledge of the target dynamics or environment geometry. The subway environment consists of realistic lighting, a moving subway, and other moving objects.

A number of simulations are conducted in order to analyze the proposed control algorithm: (1) a step response of the velocity input $v(t)$, (2) a ramp response of the velocity input $v(t)$, (3) a step response of the yaw rate input $\omega(t)$, (4) a ramp response of the yaw rate input $\omega(t)$. The set point used for all simulations is chosen such that the bounding box stabilizes to a desired size for reliable image processing at the center of the camera FoV. The single setting for the diagonal gain matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ is manually chosen and not changed between simulations.

The first simulation tests the controller response to a step input in the size error of the target, i.e., $\Delta_b(t) - \tilde{\Delta}_b$. This is done by initially placing the target human in the center of the camera FoV, such that $x_b(t_0) = \tilde{x}_b$, and at a distance away from the robot such that $\Delta_b(t_0) < \tilde{\Delta}_b$. The bottom of Fig. 2 shows the initial configuration of the robot and target in a geometrically simplified visualization of the subway, along with the visual input to the robot at the initial time $t_0$. The set point bounding box $\tilde{\mathbf{b}}$ is illustrated

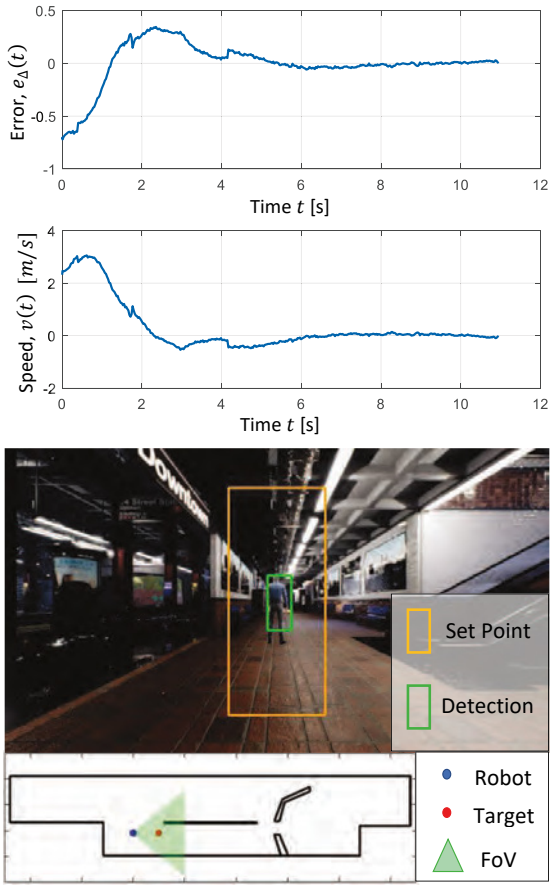


Fig. 2. Simulated step response to an initial error in the size of the target, $e_{Delta} = (\Delta_b(t) - \tilde{\Delta}_b)/\tilde{\Delta}_b$. The bottom of the figure illustrates the initial configuration of the target and robot in a geometrically simplified visualization of the subway, along with the initial visual input at the initial configuration.

as the orange bounding box and the estimated target bounding box $\mathbf{b}(t_0)$ output from the CNN is illustrated as the green bounding box. The top of Fig. 2 shows the resulting error signal $e_\Delta(t) = (\Delta_b(t) - \tilde{\Delta}_b)/\tilde{\Delta}_b$ and control input $v(t)$. The step response slightly overshoots the desired position and stabilizes in roughly five seconds.

The second simulated experiment tests the controller response to a ramp input in the size error of the target, $\Delta_b(t) - \tilde{\Delta}_b$. This is done by initially placing the target human in the center of the camera FoV, such that $x_b(t) = \tilde{x}_b$, and programming the human target to walk at a constant velocity in the direction of the initial camera optical axis. The human is programmed to walk at 1.3 m/s, which is a typical walking speed of a human. Fig. 3 illustrates the controller response as well as several snapshots throughout the simulation showing the robot-target configuration and the associated visual input. The response of the controller to such an input again stabilizes without any steady state error, due to the integral term of the control law in 8.

The next simulated experiment tests the controller response to a step input in the lateral position of the target, i.e., $x_b(t) - \tilde{x}_b$. This is done by initially placing the target human at a distance from the robot such that $\Delta_b(t_0) = \tilde{\Delta}_b$,

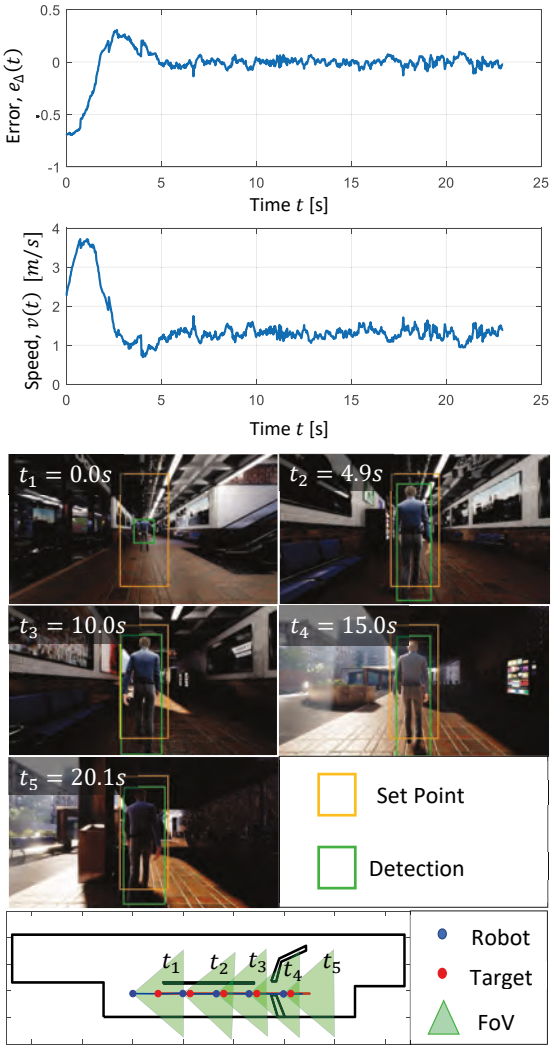


Fig. 3. Simulated ramp response of the error in the size of the target, $e_\Delta = (\Delta_b(t) - \tilde{\Delta}_b)/\tilde{\Delta}_b$. The bottom of the figure illustrates the configuration of the target and robot in a geometrically simplified visualization of the subway at a number of snapshots throughout the simulation, along with the visual input at the time of these snapshots.

but offset from the optical axis such that $x_b(t_0) < \tilde{x}_b$. The bottom of Fig. 4 shows the initial configuration of the robot and target in a geometrically simplified visualization of the subway, along with the visual input to the robot at the initial time $t_0$. The top of Fig. 2 shows the resulting error signal $e_{x_b}(t) = (x_b(t) - \tilde{x}_b)/\tilde{x}_b$ and control input $\omega(t)$. The step response slightly overshoots the desired position but quickly stabilizes about the set point configuration.

The final simulated experiment tests the controller response to a ramp input in the lateral position of the target, $x_b(t) - \tilde{x}_b$. This is done by programming the human target to walk in a circular motion centered at the camera focal point at a constant speed. The radius of the target's circular path is chosen such that $\Delta_b(t) = \tilde{\Delta}_b$. Fig 5 illustrates the controller response as well as several snapshots throughout the simulation showing the robot-target configuration and the associated visual input. The response of the controller to this ramp input very rapidly

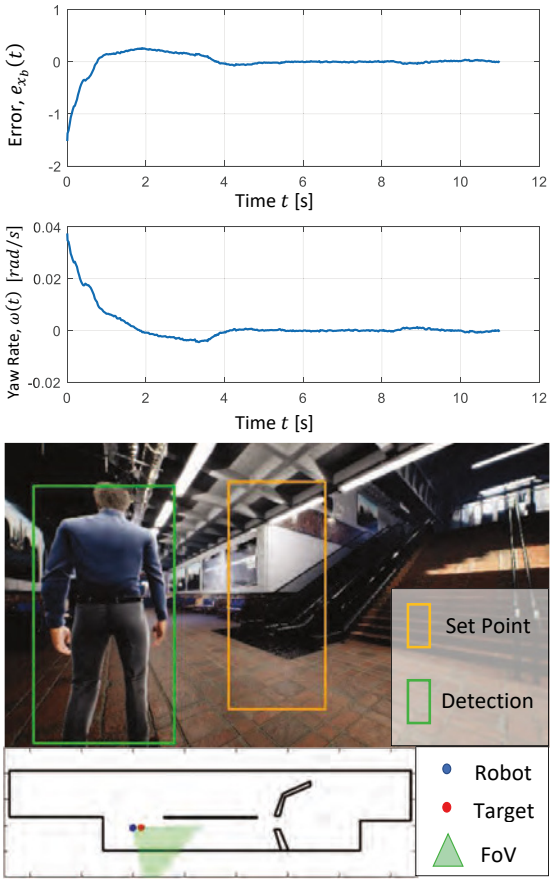


Fig. 4. Simulated step response to an initial error in the position of the target, $e_{x_b} = (x_b(t) - \tilde{x}_b)/\tilde{x}_b$. The bottom of the figure illustrates the initial configuration of the target and robot in a geometrically simplified visualization of the subway, along with the initial visual input at the initial configuration.

stabilizes without any steady state error. Some high frequency oscillations in the error signal are visible, and are caused by the periodic nature of the human walking as viewed from the side. That is, the bounding box slightly changes in shape and position due to swinging arms and legs of a walking human. Two of the spikes in the signal are caused by errors in the CNN detection algorithm, but are only present at single frames, which does not affect performance.

The four simulations performed in this study all show the controller stabilizing about the desired set point. Furthermore, the robot was programmed to follow the human through the subway environment using the proposed controller, while the target moved arbitrarily through the environment. Even in this case, where the robot was subject to small disturbances such as brief/partial occlusions, lighting variations, and change in the target motion the robot successfully stabilized about the set point. In this case, the robot was able to follow the human through the subway environment for several minutes, and possibly longer. Therefore, these simulation results suggest that, as long as the human target does not intentionally evade the robot, then the proposed controller will be capable of following the human indefinitely under reasonable conditions.

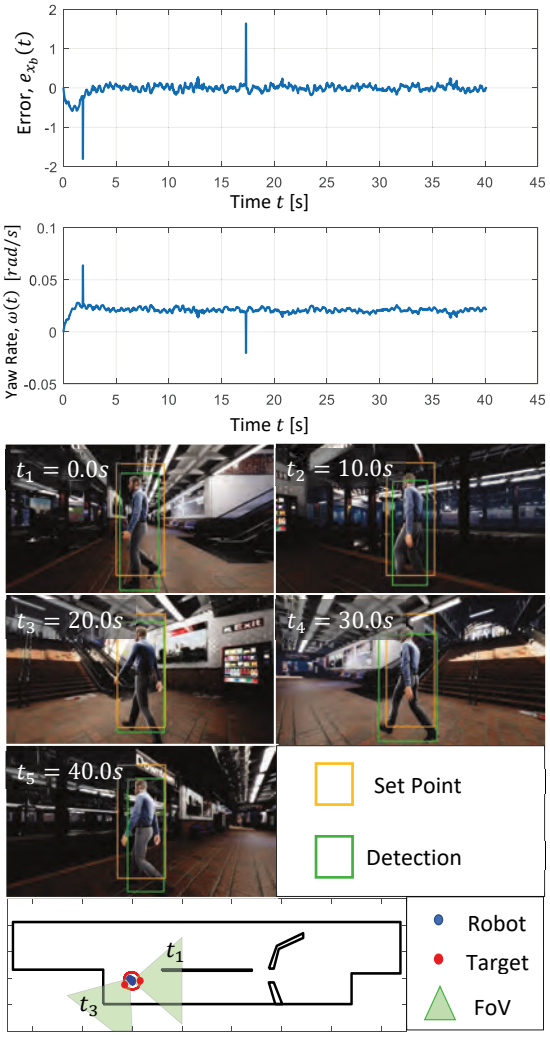


Fig. 5. Simulated ramp response of the error in the position of the target, $e_{x_b} = (x_b(t) - \tilde{x_b})/\tilde{x_b}$. The bottom of the figure illustrates the configuration of the target and robot in a geometrically simplified visualization of the subway at a number of snapshots throughout the simulation, along with the visual input at the time of these snapshots.

## 6. EXPERIMENTAL RESULTS

The experimental validation of the proposed control algorithm is done using a Clearpath$^{TM}$Jackal robot equipped with a camera and onboard computing capabilities. The robot can be accurately modeled by the nonholonomic unicycle model 4. A Vicon motion capture system is used to provide ground truth measurements of the robot and target states. It should be made clear that the Vicon data is never made available to the robot, and is only used to collect accurate pose data for results visualization. The four physical experiments presented here are exactly the same as the four simulated experiments in the previous section. That is : (1) a step response of the velocity input $v(t)$, (2) a ramp response of the velocity input $v(t)$, (3) a step response of the yaw rate input $\omega(t)$, (4) a ramp response of the yaw rate input $\omega(t)$

The first physical experiments tests the controller response to a step input in the size error of the target, i.e., $\Delta_b(t) -$

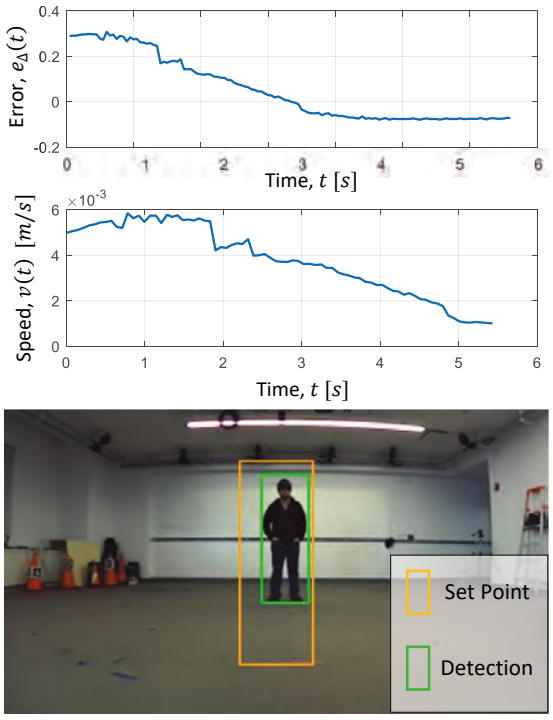


Fig. 6. Simulated step response to an initial error in the size of the target, $e_{Delta} = (\Delta_b(t) - \tilde{\Delta_b})/\tilde{\Delta_b}$. The bottom of the figure illustrates the initial visual input.

$\tilde{\Delta}_b$. This is done by initially placing the target human in the center of the camera FoV, such that $x_b(t_0) = \tilde{x}_b$, and at a distance away from the robot such that $\Delta_b(t_0) < \tilde{\Delta}_b$. The bottom of Fig. 6 shows the visual input to the robot at the initial time $t_0$. The set point bounding box $\tilde{\mathbf{b}}$ is illustrated as the orange bounding box and the estimated target bounding box $\mathbf{b}(t_0)$ output from the CNN is illustrated as the green bounding box. The top of Fig. 6 shows the resulting error signal $e_\Delta(t) = (\Delta_b(t) - \tilde{\Delta}_b)/\tilde{\Delta}_b$ and control input $v(t)$. The step response slightly overshoots the desired position then quickly stabilizes.

The second physical experiment tests the controller response to a ramp input in the size error of the target, $\Delta_b(t) - \tilde{\Delta}_b$. This is done by initially placing the target human in the center of the camera FoV, such that $x_b(t) = \tilde{x}_b$, and programming the human target to walk at a constant velocity in the direction of the initial camera optical axis. Fig. 7 illustrates the controller response as well as several snapshots throughout the simulation showing the robot-target configuration and the associated visual input. The response of the controller to such an input again stabilizes. However, due to the physical limitations of the laboratory setup (i.e., Size of the Vicon area) the human cannot walk far enough to allow the robot to fully reach steady state, but extrapolation of the available response is promising. This further illustrates the power of visually-realistic simulation in Unreal Engine$^{TM}$.

The next physical experiment tests the controller response to a step input in the lateral position of the target, i.e., $x_b(t) - \tilde{x}_b$. This is done by initially placing the target human at a distance from the robot such that $\Delta_b(t_0) = \tilde{\Delta}_b$, but offset from the optical axis such that $x_b(t_0) < \tilde{x}_b$. The

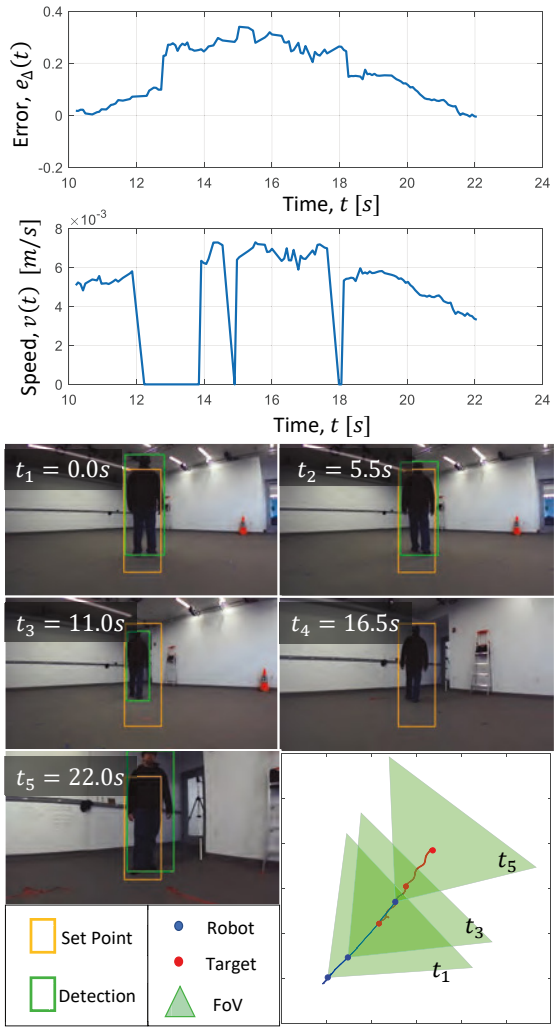


Fig. 7. Simulated ramp response of the error in the size of the target, $e_\Delta = (\Delta_b(t) - \tilde{\Delta}_b)/\tilde{\Delta}_b$. The bottom of the figure illustrates the configuration of the target and robot at a number of snapshots throughout the simulation, along with the visual input at the time of these snapshots.

bottom of Fig. 8 shows the initial visual input to the robot at $t_0$. The top of Fig. 2 shows the resulting error signal $e_{x_b}(t) = (x_b(t) - \tilde{x}_b)/\tilde{x}_b$ and control input $\omega(t)$. The step response slightly has no overshoot and quickly stabilizes about the set point configuration.

The final physical experiment tests the controller response to a ramp input in the lateral position of the target, $x_b(t) - \tilde{x}_b$. This is done by the human target walking in a circular motion centered at the camera focal point at a constant speed. Fig 9 illustrates the controller response as well as several snapshots throughout the simulation showing the robot-target configuration and the associated visual input. The response of the controller to this ramp input very rapidly stabilizes without any steady state error. These results confirm that the proposed controller as well as future computer vision-based controllers can be readily simulated in Unreal Engine$^{TM}$and then successfully implemented on physical robotic platforms.

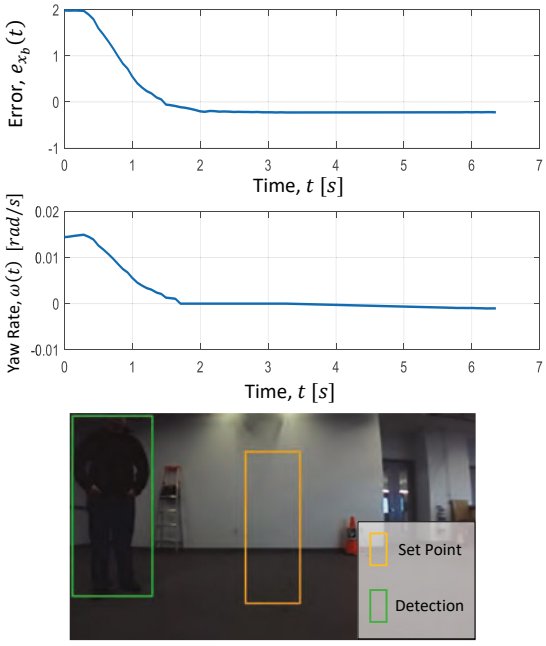


Fig. 8. Simulated step response to an initial error in the position of the target, $e_{x_b} = (x_b(t) - \tilde{x}_b)/\tilde{x}_b$. The bottom of the figure illustrates the initial visual input.

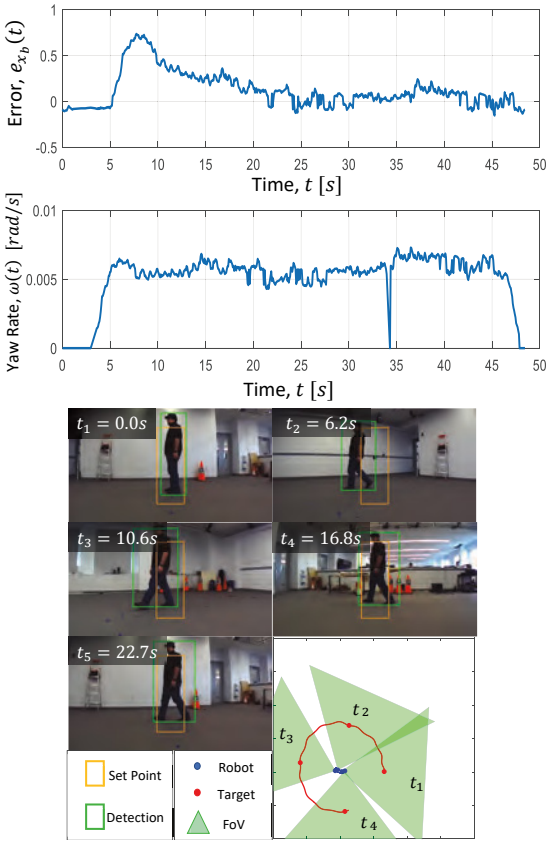


Fig. 9. Simulated ramp response of the error in the position of the target, $e_{x_b} = (x_b(t) - \tilde{x}_b)/\tilde{x}_b$. The bottom of the figure illustrates the configuration of the target and robot at a number of snapshots throughout the simulation, along with the visual input at the time of these snapshots.

## 7. CONCLUSION

This paper presents a method for mobile camera control using its video feedback in real time, in order to detect and pursue a human target. Because video frames are dependent on the camera position and orientation, the interactive and highly realistic game programming environment Unreal Engine™is used to perform virtual experiments in real time. The proposed approach relies on consistent bounding box extraction to control the camera's forward speed and yaw rate to maintain the target within its FoV and at a specified distance for accurate image processing. The simulation results show the camera tracking the human target, keeping the target within the FoV and at a reasonable distance for reliable image processing. The same control algorithm is successfully implemented on the Clearpath™Jackal robot, which also successfully follows the target and maintains it in the camera FoV. Future work includes tracking a particular human target using multiple cameras with different viewpoints in a crowded environment.

## REFERENCES


Benfold, B. and Reid, I. (2011). Stable multi-target tracking in real-time surveillance video. *Computer Vision and Pattern Recognition*.

Chen, Y.F., Everett, M., Liu, M., and How, J.P. (2017). Socially aware motion planning with deep reinforcement learning. *CoRR*, abs/1703.08862. URL `http://arxiv.org/abs/1703.08862`.

Dai, J., Li, Y., He, K., and Sun, J. (2016). R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409. URL `http://arxiv.org/abs/1605.06409`.

Esterle, L., Lewis, P., Yao, X., and McBride, R. (2017). The future of camera networks: staying smart in a chaotic world. In *International Conference on Distributed Smart Cameras*.

Goldhoorn, A., Garrell, A., Alquzar, R., and Sanfeliu, A. (2014). Continuous real time pomcp to find-and-follow people by a humanoid service robot. In *2014 IEEE-RAS International Conference on Humanoid Robots*, 741–747. doi:10.1109/HUMANOIDS.2014.7041445.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861. URL `http://arxiv.org/abs/1704.04861`.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*.

Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S.N., Rosaen, K., and Vasudevan, R. (2017). Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 746–753. doi:10.1109/ICRA.2017.7989092.

Leal-Taixé, L., Milan, A., Reid, I.D., Roth, S., and Schindler, K. (2015). Motchallenge 2015: Towards a benchmark for multi-target tracking. *CoRR*, abs/1504.01942. URL `http://arxiv.org/abs/1504.01942`.

Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C.L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312. URL `http://arxiv.org/abs/1405.0312`.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., and Berg, A.C. (2016). Ssd: Single shot multibox detector. *arXiv preprint*.

McLaughlin, N., d. Rincon, J.M., and Miller, P. (2017). Video person re-identification for wide area tracking based on recurrent neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 1–1. doi: 10.1109/TCSVT.2017.2736599.

Nageli, T., Alonso-Mora, J., Domahidi, A., Rus, D., and Hilliges, O. (2017). Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization. *IEEE Robotics and Automation Letters*, 2(3), 1696–1703. doi:10.1109/LRA.2017.2665693.

Qui, W. and Yuille, A. (2016). Unrealcv: Connecting computer vision to unreal engine. *Computing Research Repository (CoRR)*.

Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster r-cnn: Toward real-time object detection with region proposal networks. *arXiv preprint*.

Sadeghian, A., Alahi, A., and Savarese, S. (2017). Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *CoRR*, abs/1701.01909. URL `http://arxiv.org/abs/1701.01909`.

Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2017). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *CoRR*, abs/1705.05065. URL `http://arxiv.org/abs/1705.05065`.

Sonka, M., Hlavac, V., and Boyle, R. (2014). *Image processing, analysis, and machine vision*. Cengage Learning.

Tsutsui, H., Miura, J., and Shirai, Y. (2001). Optical flow-based person tracking by multiple cameras. In *International Conference on Multisensor Fusion and Integration for Intelligent Systems*.

Wei, H., Lu, W., Zhu, P., Ferrari, S., Klein, R.H., Omidshafiei, S., and How, J.P. (2014). Camera control for learning nonlinear target dynamics via bayesian nonparametric dirichlet-process gaussian-process (dp-gp) models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Zhang, Y., Qiu, W., Chen, Q., Hu, X., and alan L. Yuille (2016). Unreal stereo: A synthetic dataset for analyzing stereo vision. *Computing Research Repository (CoRR)*.