A Lightweight Visual Servoing Controller Based on Gaussian Processes for Wheelchair Corridor Navigation

A. H. Abdul Hafez *

* *Department of Computer Science, College of Computer Science and Information Technology, King Faisal University, Al-Ahsa, Saudi Arabia. E-mail: aabdulhafiz@kfu.edu.sa*

Abstract: This paper presents a lightweight visual servoing framework based on Gaussian Process (GP) regression for real-time autonomous navigation of electric wheelchairs in corridor environments. The proposed system leverages Direct Visual Servoing (DVS), where global image features are mapped directly to velocity control signals using a GP model. Histogram of Oriented Gradients (HOG) features are extracted from camera images and used as inputs to the GP, which predicts angular velocity commands without requiring explicit geometric feature extraction or Jacobian computation. The GP model is trained using a dataset collected in varied indoor corridors, with control labels derived from a geometric-based control law. The system was deployed on a Raspberry Pi embedded platform and tested on a commercial wheelchair. Experimental results demonstrate that the proposed controller operates at 7 Hz, ensures robust trajectory correction under visual noise, and achieves over 90% accuracy in corridor-following scenarios, even under challenging lighting and occlusion conditions. This approach combines the adaptability of learning-based control with the simplicity and interpretability of classical visual servoing, offering a scalable and computationally efficient solution for assistive mobility applications.

Keywords: Visual Servoing, Gaussian Process Regression, Wheelchair Navigation, Embedded Control Systems, Mobility Assistance, Lightweight Controllers

1. INTRODUCTION

Autonomous navigation in indoor environments is a fundamental capability for assistive mobile robots such as powered wheelchairs. Among typical tasks, corridor following is especially critical, enabling users to safely traverse structured spaces such as hospital hallways, residential buildings, or educational campuses. However, reliable execution of this task requires a navigation controller that is accurate, robust, computationally efficient, and capable of generalizing across visual variations in the environment.

Visual Servoing (VS) has proven effective in robotics for closed-loop control using camera input. Traditional methods rely on geometric features such as lines, points, or vanishing points extracted from images Chaumette and Hutchinson (2006); Pasteau et al. (2016); Abdul Hafez et al. (2015), which are then used to define error functions relative to desired image states. However, such methods may fail in the presence of occlusions, low lighting, or noisy sensor inputs, and often require handcrafted feature extractors and tuning of Jacobian matrices. Moreover, real-world implementations demand lightweight, embedded solutions that can operate in real-time under resource constraints.

Learning-based visual control methods offer an alternative, enabling the mapping from image observations to control commands without explicit feature modeling. Prior works have applied deep learning techniques such as CNNs for wheelchair navigation Dorbala et al. (2019b), but these models are often computationally intensive and unsuitable for low-power embedded hardware.

Importantly, the corridor-following problem is inherently *dynamic*: the platform moves continuously, the image stream is time-varying, and the controller must operate in closed loop under actuation delays, motion blur, and disturbances. Although the proposed GP regressor is learned as an instantaneous mapping from visual features to an angular velocity command, the overall navigation system operates as a discrete-time feedback controller executed at the camera/control rate (7 Hz). Therefore, the proposed solution is not restricted to a stationary setting; it is designed for and evaluated under a moving (dynamic) regime in which the wheelchair state evolves over time and the control command is updated online from successive observations.

Recent surveys underscore the breadth of research on autonomous wheelchair guidance Mole et al. (2024); Atoyebi et al. (2025). Some of this work targets indoor corridor following, where image–based visual-servoing (VS) has been shown to be effective Abdul Hafez (2014); Lakmal et al. (2020). The limited literature that does address the

⋆ This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Al-Ahsa, Saudi Arabia (Grant No. KFU260463).

wheelchair often depends on costly sensors or computationally heavy stacks—e.g. LiDAR-based SLAM Wen et al. (2022a); Panah et al. (2021), map–centric planning frameworks Sorokin et al. (2022), inverse-reinforcement-learning pipelines Wang et al. (2023); Zhang et al. (2023), or large goal–conditioned vision models Shah et al. (2023). Purely vision-based alternatives typically combine semantic segmentation with cost-map fusion Wen et al. (2022b), or focus on specialised cases such as occlusion-aware local planning Seo and Kang (2023) or obstacle avoidance Tawil and Hafez (2022). Hence, a lightweight *camera-only* solution specifically tailored to wheelchair sidewalk navigation is still missing—an unmet need that the present work addresses.

While DVS has been demonstrated to be more robust than classical visual servoing methods, it is still a challenge to formulate reference features for the desired goal Pasteau et al. (2016). The challenges are due to the possible diversity of the task environment, which demands different reference features for different situations. It can therefore be difficult to implement a robotic application such as corridor following on the base of DVS, as it is challenging to generalise the approach over the different shapes of corridors and the different internal environmental conditions that may be encountered.

In contrast, in the current article we attempt to gather the strengths of both learning-based methods and DVS techniques; Machine learning-based methods appear to generalize well and to be robust but at the cost of time and computation complexity. On the other hand, the DVS was shown to be a robust, lightweight approach but at the cost of scalability.

To achieve the corridor following task, the work presented in Pasteau et al. (2016), Pasteau et al. (2013), and Narayanan et al. (2016) used a wheelchair equipped with only a camera. The median line and the vanishing point were extracted as geometric features from the acquired images and fed into the developed control law. The goal was then to drive the wheelchair in such a direction that causes the median line of the corridor to be in the middle of the image by reducing the error between the features extracted from the current image and the desired features. This approach could be affected by the occlusion, or absence of the targeted features due to noise and measurement errors which can lead to an erroneous control signal.

The corridor following task was implemented in Dorbala et al. (2019b) using a learning-based method where a ResNet-18 CNN model was trained on the ImageNet dataset and fine-tuned using the set of corridor images which were collected for this purpose. This approach outperformed the traditional one in terms of the robustness of the extracted features, however, a heavy training phase was needed, and the inference of such a deep model, in terms of time and computational complexity on a low cost embedded board, is prohibitive. This learning-based method can be categorized as a Direct Visual Servoing (DVS) scheme where the whole image is considered an input used to produce the desired control signal.

In this paper, we propose a lightweight Direct Visual Servoing (DVS) framework that utilizes Gaussian Process (GP) regression to learn a mapping from global image features to control velocities. Histogram of Oriented Gradients (HOG) descriptors are extracted from real-time camera input and fed into a GP model trained to predict angular velocity. Unlike black-box deep networks, GPs offer interpretability, principled uncertainty estimation, and efficient learning from small datasets. The proposed system is trained using images labeled with velocities derived from a geometric control law and is fully deployed on a Raspberry Pi-powered electric wheelchair.

Extensive experiments conducted in real corridor environments demonstrate that the controller generalizes well across various indoor conditions, achieves accurate navigation with over 90% R^2 performance even under noise, and maintains 7 Hz control frequency on embedded hardware. The system is resilient to initialization errors and external disturbances and outperforms both classical geometric controllers and prior deep learning-based approaches in terms of efficiency and robustness. Moreover, the controller is evaluated in a dynamic closed-loop regime (including runs with a moving wheelchair and motion-induced visual artifacts), consistent with practical operation of assistive platforms Uğur et al. (2024).

The main contributions of this work are:

- A novel GP-based Direct Visual Servoing architecture for real-time wheelchair navigation in corridors.
- A lightweight implementation suitable for embedded systems, validated on a commercial wheelchair.
- A robustness evaluation against image noise, motion blur, and corridor appearance variation.
- A detailed comparison with traditional geometric control and CNN-based models.

The remainder of the paper reviews the Direct Visual Servoing in Section 2.1, while the wheelchair model and the corridor following tasks are introduced in Sections 2.2 and 2.3. Section 3 details the proposed approach. Section 4 presents the feature selection and comparison process, while results from the conducted experiments are illustrated in Section 5.

2. BACKGROUND AND WHEELCHAIR SYSTEM MODELLING

2.1 GP-based Direct Visual Servoing

The visual controller generates a vector of velocity signals u_c which is sent to the robot's joints to maneuver the end-effector toward the desired pose during a time step Chaumette and Hutchinson (2006); Abdul Hafez and Jawahar (2006). This signal is ideally designed in such a way that it minimizes the error $\rho(x, x_d)$ between the features x extracted from current image and the desired one x_d extracted from the reference (desired) image. Ideally, we wish to achieve the velocity vector that minimises $\rho(x, x_d)$ If the feature vector x is a global descriptor, where the image is fed into the control law directly, the VS can be referred to as Direct Visual Servoing (DVS).

The proposed approach explores using a Gaussian Process (GP) to generate the desired control signal. We call our method as GP-based DVS method. The GP-based visual controller uses a global feature vector x as an input to the GP which produces the required control signal as an

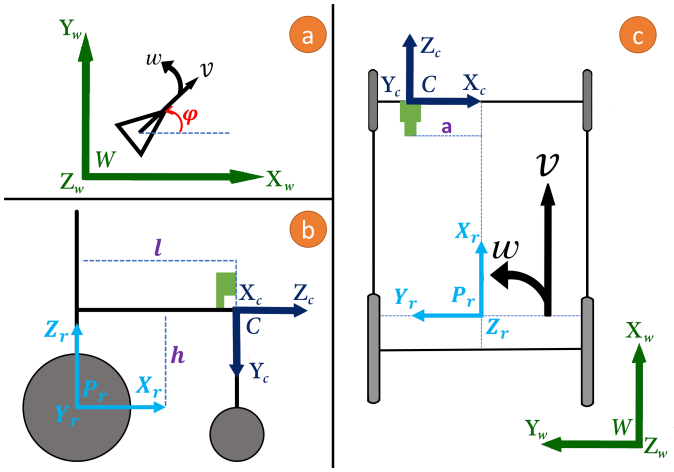


Fig. 1. Wheelchair modeling related figures with the camera C, robot P_r, and world W frames (a) The robot model (b) Side view of the wheelchair (c) Top view of the wheelchair.

output signal of the GP model. Generally, we look at a visual controller from the form:

$$u_c = \mathcal{GP}(x) \tag{1}$$

where, u_c is 6-dimensional velocity vector and $\mathcal{GP}$ indicates a 6-vector of independent Gaussian Process regressors each of which has 1-dimensional output, such that a GP-based regressor is needed for each velocity component

$$u_c = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \mathcal{GP}(x) = \begin{bmatrix} \mathrm{GP}_{v_x}(x) \\ \mathrm{GP}_{v_y}(x) \\ \mathrm{GP}_{v_z}(x) \\ \mathrm{GP}_{\omega_x}(x) \\ \mathrm{GP}_{\omega_y}(x) \\ \mathrm{GP}_{\omega_z}(x) \end{bmatrix} \tag{2}$$

In the previous equation, each of the components of u_c represents a velocity component where v refers to the translation velocity and ω to the orientation velocity.

Each of the GP components in (2) needs to be trained separately. The training dataset is represented as $\{x_i, y_i\}_{i=1}^{n}$, where x_i are the feature vector represents the ith training image and y_i are the observation values of the velocity components corresponding to the input image.

In our work, the values y_i are calculated using a geometric-based control law assuming a true values geometric features that are manually annotated. More details on training data collection and preparation are given in the section 3.2.

2.2 The Wheelchair Model

In this section, we present the kinematic model of our wheelchair. We assume that our wheelchair is a wheeled robot with six wheels, moving on a horizontal plate, see Figure 1. It is actuated using the two large rear wheels by a differential controller and two DC motors. Two passive front wheels and two small wheels behind the wheelchair are also used to provide support. In fact, a wheelchair can be modeled as a unicycle robot Gulati and Kuipers (2008)Andaluz et al. (2015) thus matching non-holonomic constraints Pasteau et al. (2016), because it has two driven wheels which are controlled independently by two direct current motors and four caster wheels to maintain stability Andaluz et al. (2015). The two control variables related to the wheelchair are the velocity v along its forward/backward direction and the angular (steering) velocity ω that perform the wheelchair rotation and facilitate changes in direction.

Figure 1 depicts the different Cartesian frames considered in this modeling task. $F_W(W,\ x_W,\ y_W,\ z_W)$ represents the world frame and $F_r(R,\ x_r,\ y_r,\ z_r)$ is a frame of the wheelchair attached to the middle of the segment formed by the centers of the two deferentially actuated wheels. We define $F_c(C,\ x_c,\ y_c,\ z_c)$ as the camera frame that is rigidly fixed to the wheelchair, where C represents the camera optical center Pasteau et al. (2016).

We are interested in finding the transformation that relates F_r and F_c.

To transfer a vector P_r expressed with respect to the robot frame (as an example) to the camera frame, we write

$$P_c = {}^cR_r P_r + {}^ct_r \tag{3}$$

where P_c represents the vector P with respect to camera frame and P_r represent the same vector with respect to robot frame. We mount the camera on the wheelchair such that it is at a height h from the segment connecting the centers of the two actuated wheels. Hence, the translation vector ${}^ct_r$ between F_r and F_c is ${}^ct_r = [w\ h\ -l]^T$.

The velocity screw transformation matrix ${}^cW_r$ which links the camera velocity to the robot velocity is given by

$${}^cW_r = \begin{bmatrix} {}^cR_r & [{}^ct_r]_\times {}^cR_r \\ 0_{3x3} & {}^cR_r \end{bmatrix} \tag{4}$$

where ${}^cR_r$ is the rotation matrix that models the fixed orientation of the robot frame relatively to the camera frame, given by

$${}^cR_r = \begin{bmatrix} \sin\theta & -\cos\theta & 0 \\ 0 & 0 & -1 \\ \cos\theta & \sin\theta & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \tag{5}$$

We can use ${}^cR_r$ for a camera with any orientation related to y axis, in our case we have only one camera with $\theta = 0°$.

In the global coordinate frame, the velocity of the unicycle robot shown in Figure 1 is represented as follows Gulati and Kuipers (2008)

$$\begin{aligned} v_x &= v\cos\varphi \\ v_y &= v\sin\varphi \\ \omega_z &= \omega \end{aligned} \tag{6}$$

where $u = (v, \omega)$ is the control input. Equation (8) describes the kinematic model of the robot with respect to the global coordinate frame and can be expressed compactly as

$${}^gV_r = J_r u, \tag{7}$$

where u is the robot velocity. By substituting the kinematics equation (6) into (7), we have

$${}^gV_r = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 \\ \sin\varphi & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{8}$$

The camera velocity can then be written as

$$u_c = {}^cW_r{}^rJ_r u, \tag{9}$$

2.3 Corridor Following Task Definition

Successful completion of the corridor following task is defined as enabling the robot to navigate down the middle of a corridor without colliding with the surrounding walls. With an image-based solution, VS can deliver the right control signal to the motors of the wheelchair. Features are extracted from images that are collected through the camera. Reducing the error between those features and the desired features minimizes the difference between the current position of the wheelchair and the desired one . One challenge is to find the features which lead to robust navigation, several different options were investigated in section 4 for this purpose.

The above definition of the corridor following task can be formulated as a visual servoing task. Our aim in this proposed work is therefor to employ the GP-based DVS controllor for a robust corridor following task against the visual input information acquired from the camera sensor. In the current corridor following task, for the sake of simplicity, only the angular velocity ω around axis y is considered to be calculated through the control law, while the translational velocity is considered as a constant value. Our proposal aims to develop a GP-based controller that is dependent only on the feature vector x extracted from the current image I without explicitly calculating the Jacobian matrix. Simplifying the control scheme presented in Eq. (2) to a single degree of freedom that is the angular velocity ω_y around axis y results with

$$\omega_y = \mathrm{GP}_{\omega_y}(x), \tag{10}$$

where x is the input feature vector that represents the current image. One may notice that for more advanced tasks like the corridor following task with an obstacle avoidance task, two GPs will be trained and used to produce both the rotational and translational velocities.

Dynamic closed-loop system. During execution, the controller runs on an image stream and updates the command at each sampling instant k with sampling period $T_s \approx 0.143$ s (7 Hz). Let the wheelchair pose be $s_k = [x_k,\, y_k,\, \varphi_k]^T$ and the measured image at time k be I_k, from which the feature vector x_k is extracted. The proposed controller computes the steering command as

$$\omega_k = \mathrm{GP}_{\omega_y}(x_k), \qquad x_k = \phi(I_k), \tag{11}$$

while the forward speed υ is kept constant in this work. The resulting closed-loop evolution follows the standard unicycle kinematics in discrete time:

$$\begin{aligned} x_{k+1} &= x_k + T_s\, \upsilon \cos(\varphi_k), \\ y_{k+1} &= y_k + T_s\, \upsilon \sin(\varphi_k), \\ \varphi_{k+1} &= \varphi_k + T_s\, \omega_k. \end{aligned} \tag{12}$$

Hence, the proposed approach explicitly targets the dynamic regime through iterative perception–control updates, and its robustness is evaluated under motion-induced effects (e.g., motion blur) and external disturbances in Section 5.

3. GP-BASED DVS FOR CORRIDOR FOLLOWING TASK

The proposed approach for the completion of the corridor following task is explained in detail throughout this section. The approach mainly consists of two phases: The training phase and Testing Phase, as shown in Figures 2. In the training phase, random corridor images are collected and their geometric features are extracted. After that, the corresponding velocity of each image in the training set is calculated using the control law mentioned in Pasteau et al. (2016). Finally, a GP-Model is trained on the images and their corresponding velocities which resulted in a trained GP-Model. In the testing phase, the model resulting from training is tested using random images captured from the camera attached to the wheelchair, these images are not used in the training process. The features of each test image are extracted and sent as input to the model. The model outputs are velocity values corresponding to each input image, as described in (10). So, every input image has its appropriate angular velocity.

3.1 GP Regressor as a DVS controller

GPs can be defined as a collection of random variables that have a joint Gaussian distribution. A GP can be completely specified as a normal distribution $\mathcal{N}_{GP}$ such as:

$$f(x) = \mathcal{N}_{GP}(m(x), k(x, x')) \tag{13}$$

where $m(x)$ and $k(x, x')$ are mean and covariance functions of the GP respectively. Also, x and x' are the feature vectors that represent two different input images respectively. Using the Squared Exponential (SE) kernel function, the covariance is

$$k(x, x') = \sigma_\omega^2 \exp\left(-\frac{1}{2l^2}\left(x - x'\right)^2\right) + \sigma_n^2 \delta \tag{14}$$

where σ_ω^2 and σ_n^2 are signal variance and Gaussian noise variance respectively, l is length scale and δ is a Kronecker delta which equals one when $x = x'$ and zero otherwise. Here σ_ω^2, σ_n^2 and l are the hyperparameters of the covariance function.

In our work, the task is to estimate the angular velocity command $\widehat{\omega}(x)$ to be produced by the visual controller by using image features as an input. Visual information are noisy in nature, so we do not have access to the function values themselves but only noisy versions of them, and we assume that

$$\omega_y = \widehat{\omega}(x) + \epsilon \tag{15}$$

where ω_y is the noisy observations of the angular velocity around y-axis, and ϵ denotes Gaussian noise with variance σ_n^2. The function values to be predicted by the GP function are denoted by $\widehat{\omega}(x)$. The covariance of the prior of ω_y is

$$\mathrm{cov}(\omega_y) = K(X, X) + \sigma_n^2 I \tag{16}$$

where $K(X, X)$ denotes $n \times n$ covariance matrix evaluated at all pairs of n training images. Then, the joint distribution of observed velocity values (ω_y) and the predicted velocities function $\widehat{\omega}^*$ at test images will be

$$\begin{bmatrix} \omega_y \\ \widehat{\omega}^* \end{bmatrix} = (0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X^*) \\ K(X^*, X) & K(X^*, X^*) \end{bmatrix}) \tag{17}$$

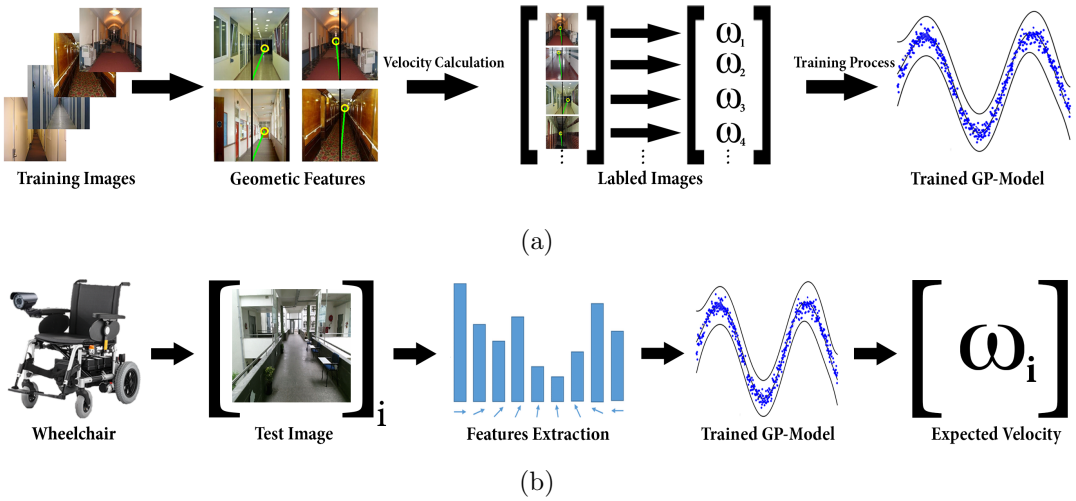


Fig. 2. Training Phase of The Proposed GP-bases DVS Approach in (a). Testing Phase of The Proposed GP-bases DVS Approach in (b).

where $K(X, X^*)$ denotes $n \times n^*$ matrix of covariances evaluated at all pairs of n training images and n^* testing images.

The predictive distribution of our model is written in terms of mean and covariance function as

$$\widehat{\omega}^* | X, \omega_y, X^* = \mathcal{N}_{GP}(\mu(\widehat{\omega}^*), \mathrm{cov}(\widehat{\omega}^*)) \tag{18}$$

where $\mu(\widehat{\omega}^*)$ and $\mathrm{cov}(\widehat{\omega}^*)$ are mean and covariance functions of the predictive distribution respectively,and can be written in compact form as

$$\mu(\widehat{\omega}^*) = k_*^T (K + \sigma_n^2 I)^{-1} \omega_y \tag{19}$$

$$\mathrm{cov}(\widehat{\omega}^*) = k(x^*, x^*) - k_*^T (K + \sigma_n^2 I)^{-1} k_* \tag{20}$$

where $k_* = K(x^*, X)$ is a vector of covariances between one test image x^* and n training images X. Also, $K = K(X, X)$, and $k(x^*, x^*)$ are the prior covariances.

It is worth noting that the hyperparameters $(\sigma_\omega^2,\ \sigma_n^2)$ and l are optimized in the training phase of the GP via maximum likelihood estimation using SLSQP algorithm which stands for sequential least square programming. The SLSQP was implemented as part of the Scikit-learn Python library. Regrading the initialization values of the hyperparameters, we noticed that changing the initial values did not greatly affect the convergence process of the optimization function in terms of finding the optimum solution in the current application.

As depicted in equation (2), there are six GP components, each of which correspond to one of the six velocity components. we are interested only in the angular velocity ω_y around axis y.

3.2 Training Data And Labeling

Calculating the labels of the gathered data is essential, as it ensures that the training data provided to the GP is sufficiently informative. The labels (angular velocities) of the dataset was calculated through the approach introduced in Pasteau et al. (2016), where the vanishing point and the median line were detected by a human annotator; some examples are given in Figure 3. After the detection step, the angular velocity was calculated for each input image by the control law proposed in Pasteau et al. (2016). The labels for each image is calculated and inserted into a one vector y.

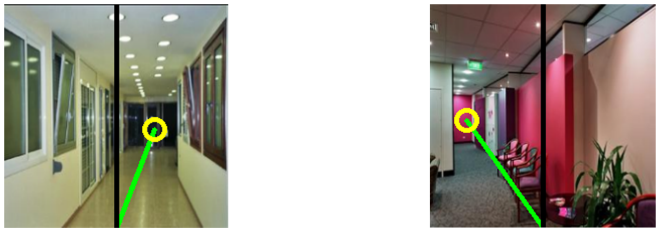


Fig. 3. Features-extracted dataset samples used for training the GP model, where the yellow circle refers to vanishing point position in the image, the black line is the middle line that splits the image vertically to two equal parts, and the green line is the line from the median line to the vanishing point.

However, a subset of the images was unreliable images in terms of the possibility to detect the vanishing point in the image. They do not show really a corridor in the scene. Their labels could not be estimated using the traditional method in Pasteau et al. (2016) similar to other images. However, a human annotator predicts an approximation for the desired direction of the velocity ω for these images. The experiments in the result section, see Figure 9-A, show how the trained model was able to predicts the correct velocity during the navigation. This is one of the strong capabilities that our proposal shows. It is also worth mentioning that no images from the experimental scenes, in Section 5.4, are used in the training dataset.

Our dataset contains images of corridors from different viewpoints and also different architectural features, gathered using the open-access datasets published in Bonarini et al. (2006) and Yang et al. (2016) in addition to the images collected in Dorbala et al. (2019b).

Table 1. The number of images in each subset

Test Set	Number of Images
JPEG Compression	672
Mild Gaussian	629
Strong Gaussian	655
Motion Blur	654
Clean	2610

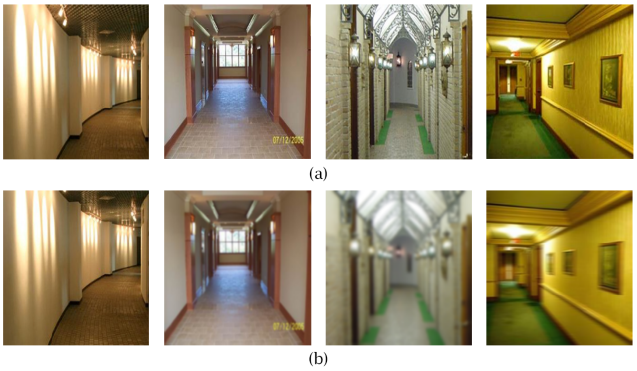


Fig. 4. Some examples from the collected dataset (a) Four images from the Clean subset (b) Four images from the noisy subset corresponding to images in the first row where the kind of noise that applied from left to right is: JPEG Compression, Mild Gaussian, Strong Gaussian and Motion Blur.

The gathered data consisted of 2610 images. Those images were duplicated and randomly corrupted with noise with one of four different noise sources which are as follows: JPEG Compression, Motion Blur, Mild and Strong Gaussian. To accomplish this task, we looped over the 2610 images, and for each image, we picked one of the four available noise sources randomly and applied it to the current image. JPEG compression was one of the noise types because the images are in PNG format which is not the normal case because of the large storage space required, so, this compression can simulate a more realistic situation. Motion Blur simulates the blur in images caused by fast robot movement. Finally, the Mild and Strong Gaussian noises are to show how our model will react to different levels of blur caused by inappropriate changes in the focus of the camera.

As a result, the whole dataset becomes equal to 5220 images. The details about this dataset are shown in table 1 and some examples are shown in Figure 4.

4. ANALYSIS OF THE DIFFERENT FEATURES

In this section we introduce various feature extraction techniques to be used in our GP model. Then, we apply evaluation metrics to these techniques to identify the best one to be used with our model.

4.1 Introduction of Evaluation Metrics

To identify the best feature extraction method to be used, three different metrics were applied. The applied metrics are: i. Mean-Squared-Error (MSE), ii. the R^2 metric, and iii. Feature Extraction Time and Angular Velocity Calculation Time based on the trained GPs.

Table 2. R^2 percentage and MSE measurements for HOG experiments

Features Per Cell	Cell Size	R^2 ((%)	MSE
9	8×8	0.00	0.9557
9	16×16	0.00	0.9557
9	32×32	75.34	0.2332
128	8×8	0.00	0.9557
128	16×16	0.00	0.9557
128	32×32	77.52	0.2126

The MSE is defined as follows:

$$MSE = \frac{1}{n}\sum_{k=1}^{n}(\omega_k - \widehat{\omega_k^*})^2$$

where ω represents the observed labels of angular velocities, $\widehat{\omega_k^*}$ represents the predicted velocities and n^* is the number of the test cases. A lower MSE valueindicates a closer fit by the model.

R^2 measures the robustness of our model, and is defined as the ratio of the Sum Squared Error (SSE) between the predicted values and the velocity labels values and the SSE between the predicted values and the mean of the velocity labels

$$R^2(\omega, \widehat{\omega}) = 1 - \sum_{k=1}^{n}\frac{(\omega_k - \widehat{\omega_k^*})^2}{(\omega_k - \bar{\omega}_k)^2}$$

where $\bar{\omega}$ is the mean of the velocity labels values related to the test images. The values of R^2 are in $(-\infty, 1)$.

For the time consumption in terms of feature extraction and the calculation of ω, we are seeking for the approach that has the best (lowest) time consumption and at the same time gives the best values in the previous two measures i.e. MSE and R^2.

4.2 Evaluation Different Feature Extraction Techniques

In the following, experiments on the mentioned feature extraction techniques are conducted in which the considered feature vector is used as an input to the GP. Using our training dataset The resulted MSE and R^2 values from the different experiments using different features are compared in the following.

Raw pixel values In this method, we do not apply any type of feature extractions, instead, we take the values of pixels directly as an input feature to the GP regressor. As depicted in Table 3, the MSE resulted by this method is equal to 0.9557 and the R^2 is equal to 0.00

Handcrafted Features HOG features were extracted from the training images and applied as an input to the GP regressor. The experiment was conducted for two different length values of the HOG feature. In other words, 9 and 128 pins per HOG feature were used. For each of these two cases, three different values for the HOG cell size were used as depicted in the second column in Table 2. The table also shows that the best results (the highest R^2 and lowest MSE) were obtained with the cell size 32×32.

Deep Features The deep features were extracted from different layers of the "ResNet18" CNN model. This model consists of 5 blocks where each of which consists of one or

Table 3. The best R^2 percentage and MSE values for HOG, ResNet18 and Raw pixel values methods

Features Extractor	R^2 **(%)**	MSE
Raw Pixels	0.00	0.9557
HOG	77.52	0.2126
ResNet18	41.96	0.5489

Table 4. The consumed time by the feature extraction methods in terms of the time needed for the feature extraction and angular velocity computed by the trained GPs.

Features	T_F **(s)**	T_V **(s)**	**Total (s)**
Raw Pixels	0	0.9879	0.9879
HOG	0.1511	0.0010	0.1521
ResNet18	0.5152	0.1627	0.6779

more convolutional layers. "ResNet18" that we used in our experiments was trained on the "ImageNet" dataset and fine-tuned on the corridors dataset found in Dorbala et al. (2019b). In our experiments we extracted deep features from the fourth convolutional layer of the third, fourth and fifth blocks with sizes of $28 \times 28 \times 128$, $14 \times 14 \times 256$ and $7 \times 7 \times 512$, respectively. The result of the experiments, fifth block gives the best result among other blocks with MSE value of 0.549.

4.3 Comparison Between Feature Extraction Techniques

By making a comparison between the best results of the Raw, HOG and ResNet18 experiments, HOG features gave the lowest MSE value between the other methods in addition to the results of R^2 and MSE in Table 3 where also HOG gave the best score among the other features. In terms of the consumed time, HOG consumed much less time than the ResNet18 features. For the Raw Pixel Values method, we assumed that no time is consumed in the descriptor extraction step because there are no processes applied to the data before it is fed into the GPs, but for the angular velocity step, it consumes more time than HOG and ResNet18.

4.4 Analysis of The Performance With Noisy Images

In section 4.1, an analysis of different feature extraction methods was conducted on a "Clean" subset of images. In this subsection we are going to make further experiments based on the extended dataset which consists of the "Clean" and "Noisy" subsets.
Our dataset consists of 5220 images with 90% and 10% of the data used for training and testing respectively. Our model was trained on the training images and tested on the test images from each of the four mentioned noisy subsets separately in addition to a test made on images from the "Clean" subset and a test made on the test images from the Full data set (Clean and Noisy). These experiments aimed to establish the effect of adding different types of noise into the images. The feature extraction methods that were used in this section are the same as the methods that achieved the best performance in the previous experiments i.e HOG (9 and 128 features per cell, 32×32 cell size) and ResNet18 (Layer from the fifth block). The results of these experiments are shown in Table 5. Using HOG features has produced higher R^2 values compared with those produced using the ResNet18 features. In addition, the experiment on the Full dataset, that contains both the clean and noisy data, has shown a little improvement in terms of the R^2 value. To avoid the increment in the computational complexity resulting from increasing the length of the feature to 128, we restrict the feature size to 9 pins.

Table 5. The R^2(%) for HoG and ResNet18 tested on six different sets. The HOG features outperofms the one extracted from deep ResNet18 when tested on the Full (noisy and clean) dataset. However, the 9-pins HOG feature is selected as it has higher computational speed.

Test Set	**HOG (9)**	**HOG (128)**	**ResNet18**
JPEG Compression	98.52	95.04	99.88
Mild Gaussian	94.19	93.45	45.54
Strong Gaussian	83.74	83.90	31.89
Motion Blur	86.83	83.58	20.77
Clean	91.02	90.88	56.24
Full	90.70	90.26	54.23

4.5 Conclusions Regarding Feature Selection

In this section, we have presented a description of our collected dataset, and three different feature extraction methods are explored. These methods are; HOG features, different layers from ResNet CNN, and raw images. The results demonstrated that HOG features have achieved the best performance in comparison with the other two methods. In addition, it is shown in Table 2 that the length of the HOG feature vector has little impact compared to the cell size. Using the size of 32×32 leads to superior performance in the R^2 value compared to the other smaller cell sizes. However, the performance has been improved a little when the length of the feature vector is increased from 9 to 128.

For the second set of experiments (Section 4.4), it was shown that using the extended dataset (Clean and Noisy) in the training phase lead to better results using HOG in terms of R^2 and also a lower MSE

5. EXPERIMENTAL EVALUATION

In this section we introduce the performance evaluation metrics used in the analysis. Then, the model performance is analysed on the original set of images and the noisy versions. Next, the accuracy of the GP model is analysed by back-projecting the estimated velocity to get the vanishing point estimate corresponding to the estimated velocity. The back-projection of the velocity is done through the geometric formulation of the problem using the TVS method. After that, we present experiments made on a real wheelchair to measure the robustness of our model and observe how the wheelchair reacts to interventions by a human. It is worth to mention that this back-projection process was first proposed and used in our previous work Dorbala et al. (2019a).

5.1 Performance Analysis Metrics

In addition to R^2 and MSE metrics that were introduced in (Section 4.1), The Precision performance evaluation metrics are used to measure the performance of our model. If the sign of the predicted velocity matches that of the observed labels, the prediction is considered to be correct, while the velocity values are classified into positive and negative according to the sign of its value. Precision is defined as the number of true positive predictions divided by all positive predictions by the model and it is written as:

$$Precision = \frac{(TP)}{(TP + FP)} \tag{21}$$

where TP is true positive predictions and FP is false positive predictions. We also used the False Prediction Score (FPS) Metric which is defined as the number of false predictions over all predictions of the model, and can be written as:

$$FPS = \frac{(FP + FN)}{(TP + FP + TN + FN)} \tag{22}$$

where TN and FN are true and false negative predictions respectively.

5.2 Analysis of The Performance With Normal And Noisy Images

In this experiment, the GP training and testing was conducted first on the normal "clean" dataset in order to observe the performance of the proposed control law in the ideal state. This was then followed by training and testing on the whole dataset i.e. Normal and noisy images to analyze the performance of the GP-based DVS model on the different noise types. HOG parameters for the both situations were set into 32×32 pixel per cell represented by 9 bins (features) for each. HOG parameters are empirically set to give the highest performance. The GP-Model was tested on four different sets of the normal dataset. Then, a specific type of noise was applied to each set of them. Noise types were as follows: Motion Blur, Strong Gaussian, Mild Gaussian, and JPEG Compression. After that, the GP-Model was tested on the noisy sets, the caparison of ω values obtained from the GP-model tested on clean and noisy sets are shown in Figure 5. The figure shows that the model's performance on normal and noisy images is roughly the same, which infers the stability and robustness of the model to noisy images.

5.3 Analysis of GP-Model Accuracy

In order to observe how well our proposed GP-Model performs, two different experiments were conducted. In both experiments, the proposed model's data is compared to that of the ground truth and the results were visualized in sections 5.3.1 and 5.3.2 respectively.

Comparing the accuracy of the estimated angular velocities In order to evaluate the accuracy of the velocities estimated using the proposed model, we compare them to the one computed using the classical geometric control law. The latter is supposed to be the most possible accurate estimate subject to the accurate extracted feature, i.e. are extracted manually. For that, we conducted an experiment to obtain both the estimated angular velocities generated from the GP-Model and the velocities obtained using the geometric method when both applied on the same set of corridor images. The result of this experiment is shown in Figure 6. The direction of the motion, i.e. left or right is correctly estimated in all cases. In addition, the magnitude of the velocity is comparable in most of the cases.

Comparing the robustness of the estimation through vanishing points x-coordinates Another way to test the robustness of the proposed approach is to test the accuracy of vanishing points locations resulted from the proposed GP-model by comparing them with the ones resulted from the classical approach. For that, we have arbitrarily selected a set of images and applied the GP-model approach on that set. Then the resulted x-coordinates of the vanishing points resulted from the model were obtained. After that the same process was repeated for the classical approach. Finally the resulted vanishing points coordinates from both approached were compared and the result were shown in Figure 7. It is implied from the figure that, the vanishing points coordinated resulted from applied a random set of corridor images is really close to that resulted from applying classical approach on the same set of images, which shows the reliability and robustness of the proposed model. The vanishing point x-coordinates are estimated by back-projecting the velocity through the geometric control using the method proposed in Dorbala et al. (2019a).

5.4 Experiments With Real Wheelchair Setup

Wheelchair experiments were conducted for the developed visual GP-based controller to achieve the corridor following task at the Hasan Kalyoncu University campus. An electric wheelchair from "Volta" was used. A single Raspberry PI equipped with a camera module was used for the complete computations including image acquisition to velocity calculation. It is worth mentioning that no other computer or computation device were used and that our trained GP model was deployed on the Raspberry PI. The angular velocity was sent to the "Sabertooth" motor driver in order to differentially drive the wheelchair motors. The method works at about 7Hz frequency, corresponding to 143ms for executing the full motion cycle.

Start from arbitrary position and external intervention The aim of this experiment is to explore the ability of our trained model to align and drive the wheelchair through the corridor starting from an arbitrary position that is not aligned with the corridor. We have conducted two kinds of experiments in which the wheelchair starts from an arbitrary position. The wheelchair is stationary in the first one while it is initially moving through the corridor in the second one.

These two trials explicitly represent *dynamic closed-loop operation*: the second trial starts while the wheelchair is already in motion, so both perception and control evolve online under time-varying visual input.

In the first experiment, the wheelchair has been turned to different angles to the left and to the right. For the largest value of turning angle the corridor is not visible in the camera field of view. In this experiment both the angular velocity and the vanishing point of the image are measured

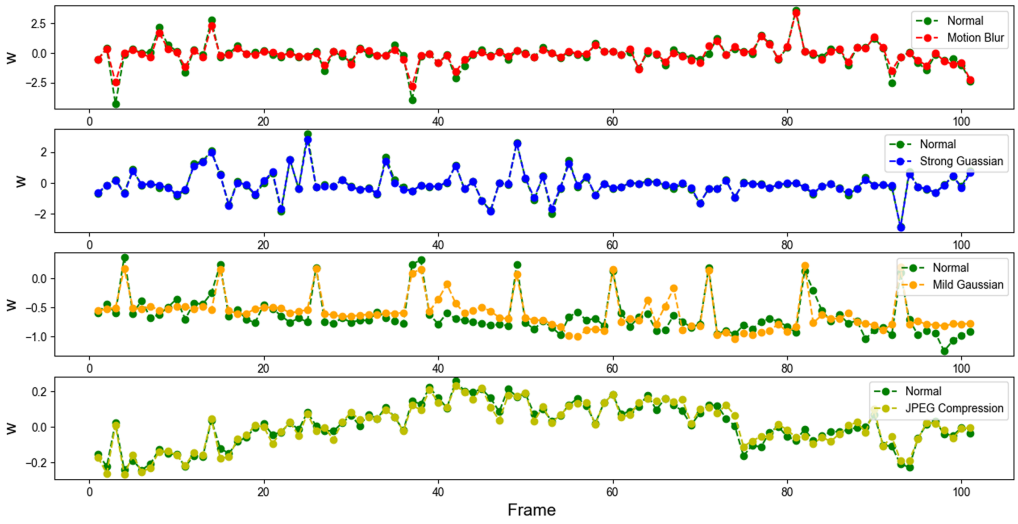


Fig. 5. The angular velocities (rad/s) computed by the GP-Model when noisy images (Motion Blur, Strong Gaussian, Mid Gaussian, and JPEG Compression from top to down) are applied as an input compared to the clean images input. Green dots refer to angular velocities resulting when testing the model on the clean dataset.

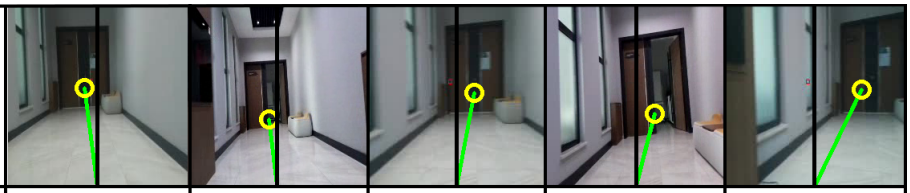
Test Image					
GP-Model	-0.1557	-0.2275	1.2028	0.3509	0.6678
Ground Truth	-0.1775	-0.1797	2.5416	0.4365	2.5112

Fig. 6. The estimated angular velocities obtained from both GP-Model Approach and Classical Approach when a sample corridor images are applied as an input. The green line represents the middle vanishing line, the black line is the line at the center of the image, and the yellow circle refers to the vanishing point position in the image. The positive ω values represent clockwise motion i.e. the vanishing point is at the right of the image, and the negative ω values represent anti-clockwise motion i.e. the vanishing point is at the left side of the image.

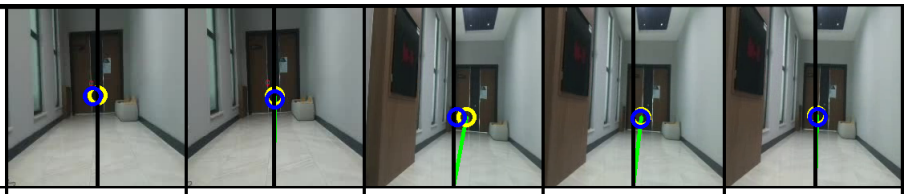
Image					
GP-Model Xv	0.0714	0.1098	0.1878	0.7786	0.0803
Ground Truth Xv	0.0751	0.1071	0.1071	0.7216	0.0893

Fig. 7. Comparing the x-coordinates obtained from GP-Model approach and the ones obtained from the classical approach. The yellow circles visualize vanishing points resulted form Classical Approach whereas blue circles represent vanishing points resulted from GP-model approach. The vanishing point x-coordinates are estimated by back-projecting the velocity through the geometric control using the method proposed in Dorbala et al. (2019a).

Table 6. The precision values obtained from running the wheelchair four runs through the corridor.

Video	Video 1	Video 2	Video 3	Video 4
Precision	1	1	0.9340	0.9252

(both values are zeros when the wheelchair is aligned with the corridor). As shown in Figure 8(A), the x-coordinate of the vanishing point and the angular velocity jump to a high value when the wheelchair is shifted from the center of the corridor and smoothly returns to the home value at which point the wheelchair is aligned with the corridor. In Figure 8(B), the turning was repeated twice, one time turning to the left and the other to the right. As the displacement of the wheelchair is larger in the second case, the controller oscillates a little during convergence.

To demonstrate the robustness of our trained model and its ability to align with and follow the center of the corridor, regardless of its initial position, two different experiments were conducted. In the first the wheelchair was stationary, while in the second the wheelchair was moving. In both experiments the wheelchair starts from an arbitrary position.

In the first experiment, the wheelchair was pushed to the left of the corridor until the end of the corridor was not visible to the camera field of view. Both the x-coordinate of the vanishing point and the angular velocity of the wheelchair were measured for each frame. As shown in Figure 8(A), when the wheelchair was pushed to the left we observed that the x -coordinate of the vanishing point varied significantly before it returned to the point where the wheelchair aligned exactly with the center of the corridor. At the same time, the angular velocity of the wheelchair varied in the opposite direction of the applied intervention in order to ensure that the wheelchair resisted the intervention and moved in the opposite direction until it eventually stayed in line with the center of the corridor. In Figure 8(B), the wheelchair was pushed two times; first to the left and then to the right of the corridor. We observed the simultaneous variations in x-coordinates of the vanishing points and angular velocity of the wheelchair in opposite directions, as in the previous case. We perceived some fluctuations until the wheelchair was aligned with the corridor.

In the second experiment, a large part of the main structure of the corridor was initially out of the camera view field. The wheelchair was able to correct its direction and align with the corridor. As depicted in Figure 9, the first shot marked with the letter A shows the image captured by the camera at the initial moment. The negative value of the corresponding angular velocity at the position (A) works to correct the direction of the wheelchair. The correction takes 0.3 seconds, the progress of it is shown in the next images i.e. B, C, and D. It is clear that the wheelchair has become aligned with the corridor at the position of image C.

Precision analysis over multiple runs The precision performance metric is evaluated over experiments conducted by running the wheelchair four different times through the corridor. Four different video samples are captured by the Raspberry Pi camera attached on the Wheelchair to measure the robustness of our model. The results of the experiment are depicted in Table 6. It can be noticed that the precision is always close to or equal to 1.

Table 7. Comparison to the work presented in Dorbala et al. (2019b).

	GP Approach	CNN Approach
Setup	GP , Rasp. Pi 4	CNN, Laptop
Exec. Time/ Epoch	0.143	1.8
Op. Freq. (Hz)	7	0.6

5.5 Comparison to the State-of-the-art

As mentioned in the introduction section, the most recent and related works to the visual wheelchair corridor navigation task were presented in Pasteau et al. (2016),Pasteau et al. (2013), Narayanan et al. (2016), and Dorbala et al. (2019b). It is worth mentioning that Pasteau et al. (2016), Pasteau et al. (2013), and Narayanan et al. (2016) present different overlapping parts from the same work, indeed our comparison will be restricted to the results shown in Pasteau et al. (2016) which is the most representative publication of that work. We also compare with Dorbala et al. (2019b) as it is the most recent work to view the corridor following as a visual task.

The work in Pasteau et al. (2016) uses the vanishing point as a visual geometric feature, referred to here as the Traditional Viosual Servoing TVS. The design of the control law requires the feature to be always available in the camera field of view, otherwise the controller will be disabled. Figure 10 shows the velocity calculated for different input images in both works, the one presented in Pasteau et al. (2016) and the one presented in this paper. Our GP-based controller is able to produce a reasonable velocity in the absence of the corridor end while the one presented inPasteau et al. (2016) could not produce a velocity command as the controller is disabled in this case.

The work presented in Dorbala et al. (2019b) was able to produce a velocity command and to continue operation in the absence of a full corridor image in the camera view. However, the current approach is more time efficient than that described in Dorbala et al. (2019b). Table 7 shows the execution time and operating frequency for the current approach and that described in Dorbala et al. (2019b). In our approach, we applied a GP model on a Raspberry Pi 4 with 8GB RAM while in Dorbala et al. (2019b) the is CNN implemented using a laptop. Our approach was able to achieve a computation speed 11 times more than the one reported in Dorbala et al. (2019b).

6. CONCLUSIONS

A wheelchair corridor following task is demonstrated in this work using a GP-based visual controller. The GP used the global HOG features as an input and was trained to produce the velocity signal as an output. The conducted experiments demonstrated the robustness and real-time performance of our approach in different real-world scenarios. For example, the wheelchair was able to return to its home position, the center of the corridor, even after multiple interventions were applied. In this work, we tested three types of features and selected HOG. In future work,

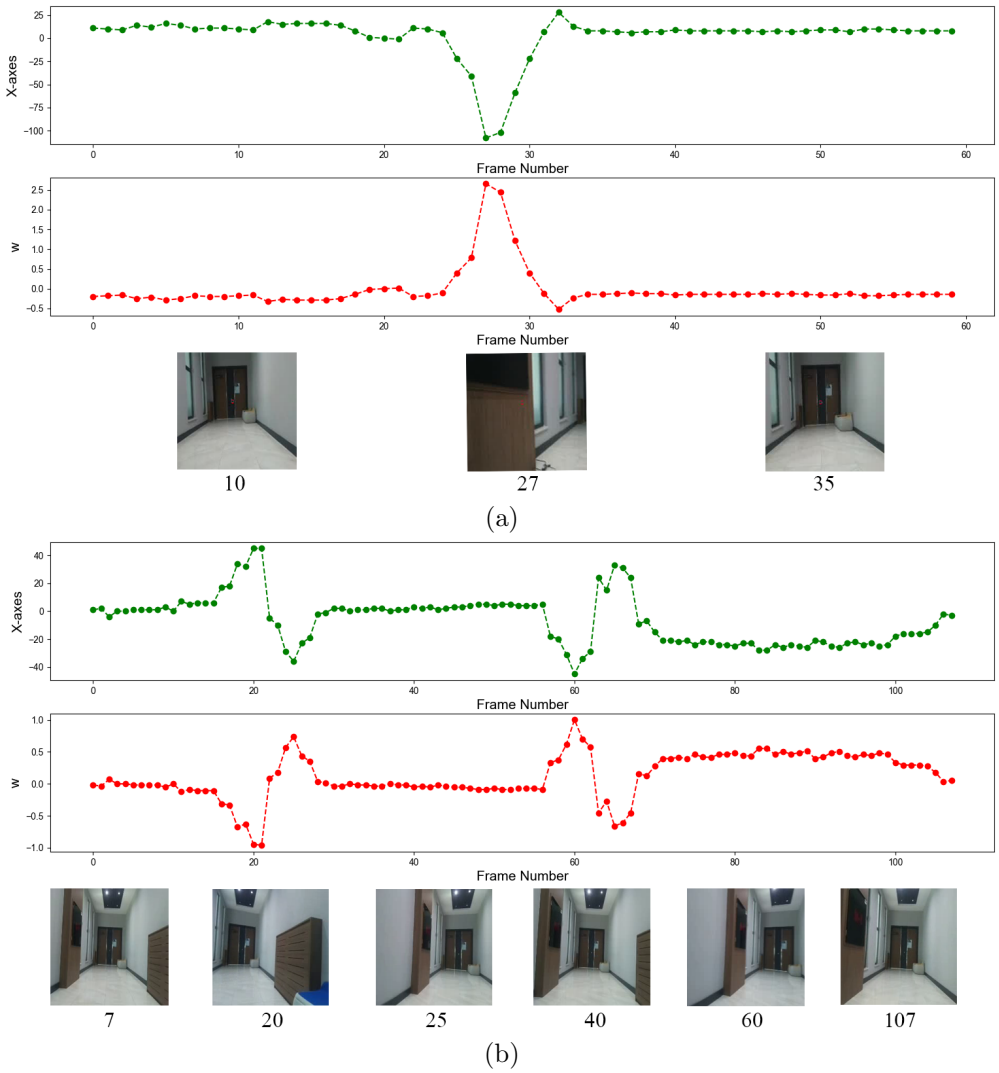


Fig. 8. The upper two figures show a curve of the X-axis of the vanishing points and angular velocities ω along with frame numbers captures by the camera. The images with names 10, 27, 35 show frame number that is captured at that moment. The lower two figures show a curve of the X-axis of the vanishing points and angular velocities ω (rad/s) along with frame numbers captures by the camera. The images with names 7, 20, 25, 40, 60 107 show frame number that is captured at that moment.

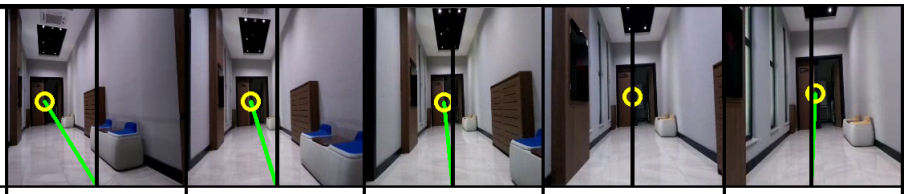
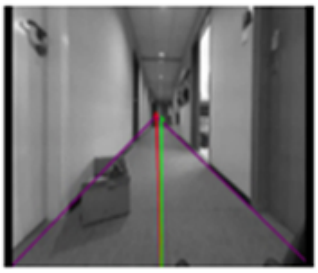
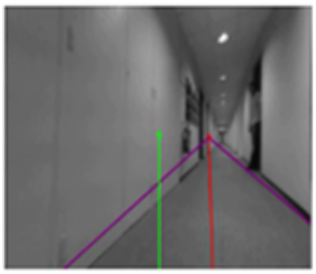
Image					
Angular Velocity w	-0.7397	-1.0723	-0.1849	0.0957	0.2629
Vanishing Point Xv	153.4	172.04	122.3	106.6	97.27

Fig. 9. The upper figure shows two rows of values corresponding to random images. The first row is the angular velocities ω predictions of each image. The second raw gives pixel value of x-coordinates of the vanishing point for each image. Yellow circle refers to vanishing point position on the image, and black line marks the center of the image. We notice that, the further the vanishing point from the center of the image, the higher the absolute angular velocity the model predicts. Furthermore, when the point is located at the left of the center of the image, the angular velocity prediction becomes negative and vice versa, which forces the wheelchair to navigate directly toward the vanishing point.

Our Approach			
	Frame 1	Frame 2	Frame 3
Frame			
Angular Velocity	0.5431	0.0464	0.0014

(a)

TVS Approach			
	Frame 1	Frame 2	Frame 3
Frame			
Angular Velocity	-0.0326	NULL	0.4762

(b)

Fig. 10. Comparison to the tradition visual serving VS work presented inPasteau et al. (2016). Our approach is in (a) and the TVS' one is in (b).

we aim for a CNN architecture that can be trained to perform better than HOG with fine-tuned filters. We believe that such trainable features will produce the highest performance scores.

Experiments on ResNet18 have shown that the last block has achieved the best performance. Further experiments will be conducted considering the first block which is expected to have more general features. We believe that they should fit better with the GPs. Also, other CNN models can be examined. A more simplified REsNet or Alex365 CNN architecture will be trained and investigated to maximize the performance of the solution.

As future work, we will extend the analysis to incorporate richer dynamic effects (e.g., actuator dynamics and delay compensation) and investigate uncertainty-aware speed adaptation under degraded visual conditions Uğur et al. (2024).

ACKNOWLEDGMENT

The author would like to acknowledge Mr. Ammar Tello and Dr. Peter Green (CARA Syria Program), as well as Mr. Ismail Haj Osman (TÜBİTAK, Project No. 117E173), for their contributions to the early experiments of this work. The Titan Xp GPU used in this research was generously donated by NVIDIA Corporation.

REFERENCES

Abdul Hafez, A.H. (2014). Visual servo control by optimizing hybrid objective function with visibility and path

constraints. *Journal of Control Engineering and Applied Informatics*, 16(2), 120–129.

Abdul Hafez, A.H. and Jawahar, C.V. (2006). Probabilistic integration of 2d and 3d cues for visual servoing. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, 1–6. IEEE.

Abdul Hafez, A.H., Nelakanti, A.K., and Jawahar, C. (2015). Path planning for visual servoing and navigation using convex optimization. *International journal of robotics and automation*, 30(3), 299–307.

Andaluz, V.H., Canseco, P., Varela, J., Ortiz, J.S., Pérez, M.G., Morales, V., Robertí, F., and Carelli, R. (2015). Modeling and control of a wheelchair considering center of mass lateral displacements. In *Intelligent Robotics and Applications*, 254–270. Springer.

Atoyebi, O., Wister, A., Mattie, J., Beadle, J., Gutman, G., Chaudhury, H., Sparrey, C.J., Jones, O.Y., Mortenson, W.B., O'Dea, E., et al. (2025). Power assist add-ons for adult manual wheelchair users: A scoping review. *Assistive Technology*, 1–12.

Bonarini, A., Burgard, W., Fontana, G., Matteucci, M., Sorrenti, D.G., and Tardos, J.D. (2006). Rawseeds: Robotics advancement through web-publishing of sensorial and elaborated extensive data sets. In *In proceedings of IROS*, volume 6, 93.

Chaumette, F. and Hutchinson, S. (2006). Visual servo control. i. basic approaches. *IEEE Robotics & Automation Magazine*, 13(4), 82–90.

Dorbala, V.S., Abdul Hafez, A., and Jawahar, C. (2019a). A deep learning approach for robust corridor following. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3712–3718. doi: 10.1109/IROS40897.2019.8967914.

Dorbala, V.S., Hafez, A.A., and Jawahar, C. (2019b). A deep learning approach for robust corridor following from an arbitrary pose. In *2019 27th Signal Processing and Communications Applications Conference (SIU)*, 1–4. IEEE.

Gulati, S. and Kuipers, B. (2008). High performance control for graceful motion of an intelligent wheelchair. In *2008 IEEE International Conference on Robotics and Automation*, 3932–3938. IEEE.

Lakmal, I.T., Perera, K.N., Sarathchandra, H.H.Y., and Premachandra, C. (2020). Slam-based autonomous indoor navigation system for electric wheelchairs. In *2020 International Conference on Image Processing and Robotics (ICIP)*, 1–6. IEEE.

Mole, A., Gurav, S., Shinde, S., Bhagwat, Y., and Patil, M.B. (2024). Survey on smart wheelchairs. *Journal of Electronics and Telecommunication System Engineering*, 43–50.

Narayanan, V.K., Pasteau, F., Marchal, M., Krupa, A., and Babel, M. (2016). Vision-based adaptive assistance and haptic guidance for safe wheelchair corridor following. *Computer vision and image understanding*, 149, 171–185.

Panah, A., Motameni, H., and Ebrahimnejad, A. (2021). An efficient computational hybrid filter to the slam problem for an autonomous wheeled mobile robot. *International Journal of Control, Automation and Systems*, 19, 3533–3542.

Pasteau, F., Babel, M., and Sekkal, R. (2013). Corridor following wheelchair by visual servoing. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 590–595. IEEE.

Pasteau, F., Narayanan, V.K., Babel, M., and Chaumette, F. (2016). A visual servoing approach for autonomous corridor following and doorway passing in a wheelchair. *Robotics and Autonomous Systems*, 75, 28–40.

Seo, D. and Kang, J. (2023). Collision-avoided tracking control of uav using velocity-adaptive 3d local path planning. *International Journal of Control, Automation and Systems*, 21(1), 231–243.

Shah, D., Sridhar, A., Bhorkar, A., Hirose, N., and Levine, S. (2023). Gnm: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 7226–7233. IEEE.

Sorokin, M., Tan, J., Liu, C.K., and Ha, S. (2022). Learning to navigate sidewalks in outdoor environments. *IEEE Robotics and Automation Letters*, 7(2), 3906–3913.

Tawil, Y. and Hafez, A.A. (2022). Deep learning obstacle detection and avoidance for powered wheelchair. In *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 1–6. doi: 10.1109/ASYU56188.2022.9925493.

Uğur, E., Kara, T., Abdulhafez, A., and Osman, I.H. (2024). Design and analysis of a novel sidewalk following visual controller for an autonomous wheelchair. *Advances in Electrical & Computer Engineering*, 24(1).

Wang, T., Dhiman, V., and Atanasov, N. (2023). Inverse reinforcement learning for autonomous navigation via differentiable semantic mapping and planning. *Autonomous Robots*, 1–22.

Wen, M., Dai, Y., Chen, T., Zhao, C., Zhang, J., and Wang, D. (2022a). A robust sidewalk navigation method for mobile robots based on sparse semantic point cloud. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7841–7846. IEEE.

Wen, M., Zhang, J., Chen, T., Peng, G., Chia, T., and Ma, Y. (2022b). Vision based sidewalk navigation for last-mile delivery robot. In *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 249–254. IEEE.

Yang, S., Maturana, D., and Scherer, S. (2016). Real-time 3d scene layout from a single image using convolutional neural networks. In *2016 IEEE international conference on robotics and automation (ICRA)*, 2183–2189. IEEE.

Zhang, T., Liu, Z., Pu, Z., Yi, J., Liang, Y., and Zhang, D. (2023). Robot subgoal-guided navigation in dynamic crowded environments with hierarchical deep reinforcement learning. *International Journal of Control, Automation and Systems*, 1–13.