Perception-aware Visual Odometry

S.H.J. (Stephan) Visschers

MSc Report

Committee:
Prof.dr.ir. G.J.M. Krijnen
H.W. Wopereis, MSc
Dr.ir. G.A. Folkertsma
Dr.ir. L.J. Spreeuwers

January 2019

003RAM2019
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

Visual Odometry (VO) is the practice of motion estimation of a mobile robot from a series of images, which is an appropriate localization tool in a GPS-denied environment. Visual odometry relies on the extraction and matching of distinct features within the recorded images to determine the robot pose. Consequently, when a view lacks the necessary information to estimate the current pose of the robot, the localization process loses its tracking, which is generally required to control the robot.

A potential solution to prevent the lack of necessary information is to implement a gaze control algorithm that actively actuates the camera towards a more favourable gaze. In this project, the tracking problem has been investigated by preliminary experiments and theoretical analysis, to analyse the conditions which facilitate tracking loss. A control algorithm has been designed in which the setpoints of the physical system are determined by a virtual mechanical model driven by the feature distribution in camera frame and the state of the system. The influence of the features on the system is continuously adapted to take into account the undesired conditions found in the analysis step.

The resulting change in behaviour of the motion of the camera unit is investigated using a series of simulation experiments, which test and demonstrate the potential benefit of the designed gaze control algorithm and its individual aspects.

These simulations demonstrate that the different aspects of the designed gaze control algorithm work as intended, to a limited extent.

The obtained simulation results increase the plausibility of the efficacy of the proposed solution. However, a definitive confirmation has not been obtained due to a lack of relevant physical data.

Contents

1 Introduction

Visual Odometry (VO) is an incremental, online estimation process of a vehicles pose by analyzing the image sequences captured by a camera.[1] A persistent problem in state-of-the-art visual odometry algorithms is their dependency on feature-rich environments [13][1][22][25]. Features are pieces of information in an image, such as points, lines, edges or corners.[1] When the amount of features in the field-of-view of the camera diminishes, so does the reliability of the pose estimation process. In an unknown environment, it is impossible to predict whether an unvisited area has enough features to maintain robustness.

The current state-of-the-art solutions to the feature reliability problem can be divided into four categories:

The first category is the manipulation of the camera angle to obtain a view with more information. Peretroukhin et al.[19] studied the effect of varying the yaw angle of a stereo camera on the visual odometry motion estimation error. Kelly and Sukhatme[14] studied the effect of changing the pitch angle of a stereo camera on the visual odometry motion estimation error. Their results showed that an oblique angle for both the yaw (w.r.t. the forward direction) and the pitch (w.r.t. the horizontal line) resulted in an increased accuracy during pose estimation. Correa and Soto[8] use a topological map of the environment and a particle-filter based optimization function to direct the camera towards more information-rich parts of the environment. Manderson et al.[15] use an offline-trained online classifier to iteratively determine a discretized change in gaze of a pan-tilt camera towards feature-rich regions, to improve their SLAM system. Their work demonstrated that increasing the effectiveness of SLAM can be achieved by actively updating the camera gaze.

The second category can be seen as an extension of the pan-tilt concept: the application of an omnidirectional camera, which covers the different possible views simultaneously. Omnidirectional cameras have been applied to aid visual odometry frequently. For example, Bunschoten and Kröse[6] successfully applied a form of visual odometry on a dataset created by an omnidirectional camera. Corke et al.[7] compared two approaches to visual odometry from an omnidirectional image sequence, namely an optical flow method and a shape-from-motion technique. Tardif et al.[26] applied a custom monocular SLAM system to an omnidirectional camera. Their SLAM system decouples the estimation of camera rotation and translation. The translation is estimated using the 3D points determined by the SLAM algorithm, while the rotation is estimated using the epipolar constraint. Similarly, Scaramuzza and Siegwart[23] applied a homography-based tracker that extracts features most likely belonging to the ground-plane to estimate translation, while using an appearance-based algorithm to estimate the rotation. Zhang et al.[28] evaluate the effect of the camera's field-of-view on the accuracy and robustness of visual odometry. They conclude that the benefit of a larger field-of-view on visual odometry is not straightforward, but depends on the situation. An increase in robustness from a larger FOV comes at a cost of a decrease in accuracy. Additionally, their experimental data indicate that cameras with a large field-of-view are beneficial for indoor situations, while this benefit decreases as the scale of the scene grows, because the angular resolution is smaller than that of a perspective camera. Additionally, the use of omnidirectional cameras comes at a cost of increased computational complexity.[1] These issues limit the applicability of omnidirectional cameras.

The third category is the manipulation of the path planning of a robot that takes into account the visual information of the environment, in order to minimize the propagation of pose uncertainty. Costante et al.[9] use texture information of the environment to adapt the path-planning process of a downwards-looking drone. Using this adaptive planning, the drone attempts to

avoid feature-poor areas. The implementation of Costante et al. has the camera fixed in a downward looking orientation, which limits the potential improvement of VO reliability using texture-aware path planning, because the camera and the drone cannot move separately.

The last category is the implementation of software strategies to minimize tracking loss. Straub et al.[25] apply binary descriptors to features to enable quick nearest-neighbour search to attempt to recover tracking. Alternatively, SLAM methods can be applied. For example, Mur-Artal et al.[17] use a bag-of-words model and RANSAC to compare the current frame to relocalization candidates in the created map. Software solutions can incorporate intelligent solutions to specific situations. However, the effectiveness of these solutions are limited by physical limitations. It can only change the usage of available information, but not change the amount of information obtained.

This report presents a novel approach in the first category to improve the reliability of pose estimation based on existing Visual Odometry (VO) algorithms, by implementing an active gaze control algorithm. The gaze of a camera is defined in this context as the direction the camera is looking towards and the field-of-view related to that direction. The gaze will be continuously altered based on a dynamical model. By altering the gaze, areas that contain insufficient information for sustained reliable VO can be avoided.

The proposed solution to the feature-dependent reliability issues of pose estimation, and the issues present in the solutions of others, is the application of a pan-tilt actuated camera unit. The camera unit consists of a stereo camera and an on-board IMU. The actuators will be controlled using a virtual dynamical model that can be tuned to adapt the behaviour of the gaze control system. This virtual model incorporates virtual springs, damping and inertia that govern the motion of the camera unit.

In order to increase the efficacy of the proposed solution, several steps in research were explored: first, which phenomena occur in visual odometry and how these affect the reliability of the visual odometry process is investigated. Second, considering the phenomena found in the previous step, how these can be taken into account while designing a controller with the goal of avoiding any unfavourable gazes. And last, how the reliability of the created controller algorithm compares to other approaches to the described problem. Additionally, any limitations of the created algorithm that are encountered during this research will be documented.

2 Problem Description & Analysis

The goal of this research is to improve the reliability of visual odometry by means of gaze control. This chapter will analyse potential pitfalls of visual odometry algorithms related to their feature-dependent reliability, based on the theoretical structure of the algorithm and data from preliminary experiments. Some of these pitfalls are software specific and therefore any results are dependent on the specific software used. The visual odometry software used for this study is ROVIO. ROVIO[3, 2] is short for "RObust Visual Inertial Odometry" and the algorithm is developed by the Autonomous Systems Lab of the ETH Zurich. ROVIO is selected as the VO algorithm that will be applied during this research, because it is the feature-based method with the best trade-off between computational complexity, robustness and accuracy.

2.1 ROVIO

ROVIO uses an Iterated Extended Kalman Filter (IEKF) to fuse inertial measurements with visual data from one or more cameras. Section 2.1.1 explains the concept of the IEKF.

The following steps are executed by ROVIO (see also figure 2.1) to keep track of landmarks (extracted features) across frames:[2]

1. For each landmark (blue dot), a patch feature (large blue square) is extracted.
2. Then, for each landmark, the image coordinates (yellow dot) and the corresponding covariance (yellow ellipse) are estimated using the filter's process model driven by the input from the IMU.
3. A number of candidate points (yellow dots) for the new frame are selected based on the magnitude of the uncertainty.
4. Each candidate is iteratively updated (black arrows). This update integrates patch intensity errors with the estimated motion from step 2. The best, valid candidate points are selected using outlier detection and quality checks (green vs. red squares).

Using the matching of features between frames, the IEKF can apply the epipolar constraint to estimate the camera motion.

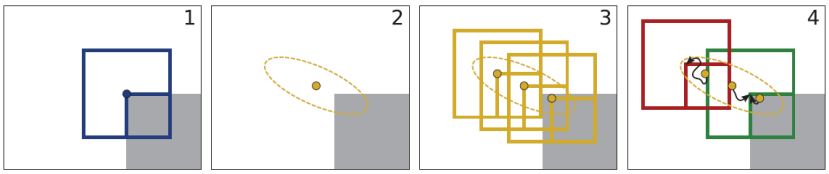


Figure 2.1: Each graphic corresponds to a step in the landmark tracking process of ROVIO.[2]

2.1.1 Iterated Extended Kalman Filter

The Kalman Filter is a two-phase recursive process, see figure 2.2, that uses measurement data observed over time, which can include noise, to create the best estimate of the system state. This method assumes models are available of the system and the observation method. In general, system and observation noise are modelled as Gaussian noise. The first phase consists of a prediction of the state in the next time step based on the system model applied to the previous state estimation, and a prediction of the corresponding error covariance. Then, in the correction phase, the predictions are corrected by using the latest measurement data, and the observation model.[20]

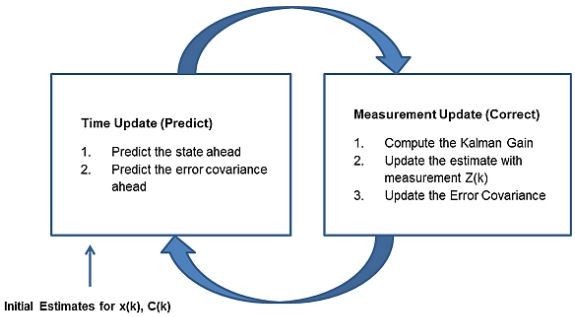


Figure 2.2: Kalman Filter process[20]

The models of the system and the observation method in the Kalman Filter are designed as linear models. However, in practice this is only true in rare, special cases. Therefore, the Extended Kalman Filter (EKF) also allows non-linear state-transition and measurement models, and applies the first-order Taylor expansion around the latest predicted state and covariance to locally approximate the non-linear model as a linear process.[12]

The Iterated Extended Kalman Filter (IEKF) expands on the concept of the Extended Kalman Filter by recursively improving the linearisation of the EKF. This results in an improved state estimation at the cost of an increased computational cost.

2.2 VI-sensor

The input data to ROVIO is captured using a Skybotix Visual-Inertial (VI) sensor[18], which was provided for this research. The sensor package consists of a stereo camera set-up, consisting of the Aptina MT9V034 CMOS sensors with Lensagon BM2820 lenses[11], complemented with an on-board ADIS16488 IMU. The signals from these sensors are time-synchronized.

2.3 Potential pitfalls

The IEKF uses both the data from the stereo camera and the IMU to create a prediction of the system state, which contains the camera pose. Therefore, if the quality of these data streams degrade significantly, the uncertainty of the state prediction grows exponentially and the estimated pose will drift. This degradation can occur in situations where feature quantity or quality are poor, or if the IMU-data contains a lot of noise. Vice versa, correspondence between the data streams can also falter if the majority of the features in the image data are in motion themselves. However, non-static scenes are outside the scope of this research.

Degradation of feature quality can occur when the distance increases significantly. Then the distance estimation uncertainty grows and the effect of the camera motion on the position of the feature in the image becomes less distinguishable. As a consequence, the calculation of the camera translation becomes less reliable.

To test this, recordings were made with the VI-sensor of the Horst tower building while walking sideways for a distance in the order of several meters (≈ 8) and returning to the starting point (within an error margin of 0.5 m), from four different distances with respect to the Horst tower: 10, 20, 30 and 60 meters ground distance approximately. This experiment shows that features at larger distances exhibit a larger final error (see table 2.1). The result of the motion at a distance of 30 meters is an outlier to this trend because of a large rotational error, see figure 2.3. However, the lighting conditions for these recordings were not consistent due to shadows, and the motion included unwanted vibrations. Both of these parasitic disturbances influenced the result, to an unknown degree.

distance to Horst (m)	10	20	30	60
final Euclidean error (m)	0.2	1.2	12.8	6.4

Table 2.1: Final errors - distance experiment

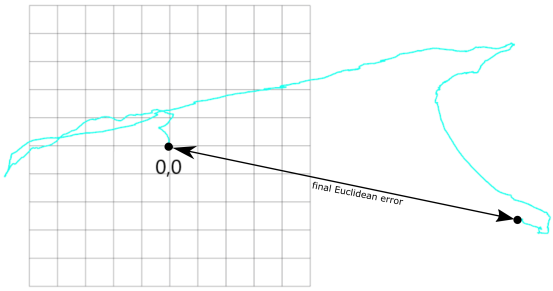


Figure 2.3: Path estimated by ROVIO from the Achterhorst-30m dataset (grid size is 1 m)

Additionally, a scene with features at various distances was captured to test how the algorithm reacts to this variation of distances in one scene. In the walkway between the Carré and Nanolab buildings on the UT campus, a forwards and backwards trajectory was traversed (by walking) with the VI-sensor facing forward. The algorithm exhibited a preference for features that appeared at a closer distance (between 1 and 50 meters). It chose features that appeared on the buildings on the sides instead of the bridge between the buildings or the top of the Horst tower that is visible in the background (see figure 2.4). However, these results can have been affected by the standard autofocus settings of the VI-sensor.

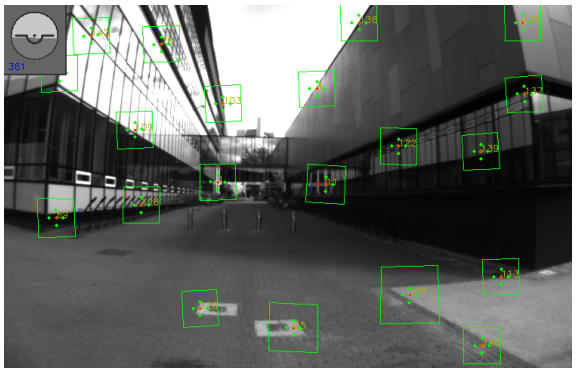


Figure 2.4: Feature preference of ROVIO exhibited in walkway

Another visible effect demonstrated by an indoor experiment in the hallway of the Carré building, was that when the camera moved backwards, the algorithm showed a preference for holding on to the already existing features in favour of the new features that are added around the edges. This creates a clustering effect. This effect ended when a number of features were moving out of frame due to a rotational motion. See figure 2.5.

Another hypothetical pitfall that was tested is if there is a minimum distance beyond which motion cannot be reconstructed by ROVIO. It appears that this minimum distance for which ROVIO is able to recover camera motion is marginal. This was tested by moving the VI-sensor towards a wall until contact was established between the VI casing and the wall, and after a brief idle moment, moving backwards to return to the starting point. While the VI was in contact with the wall, ROVIO was still able to extract features, because of the autofocus capability of the VI-sensor. It is important to take into account that even though ROVIO is able to detect features at such a close range, these features will move quickly out of the frame if the sensor moves sideways. The final Euclidean error of this closeby experiment is 0.4 m. On a trajectory

Figure 2.5: Clustering effect obtained from indoor experiment. 3 frames are shown in chronological order while the camera is moving backwards. The sequential frame number is shown in left corner.

of approximately 4 m this is an error of 10%, which is significant. This decrease in accuracy is most likely due to the shift of the focal point by the autofocus capability. During the wall approach it can be seen that after this transition phase, the algorithm has to extract a whole new set of features. This results in a brief moment during this transition where there is a lack of correspondence of features in subsequent frames.

Motion blur might form an obstacle for the feature matching step, as features are distorted. However, a quick test involving rapid motions showed that ROVIO is able to handle fast motions, as it retains the ability to distinguish features, which is also visible in figure 2.8. The path of these rapid motions, as calculated by ROVIO (expressed in Ψ_w, where the origin is defined as the initial pose) is displayed in figure 2.6. The linear velocities of the VI-sensor (expressed in Ψ_c) during this blur test are plotted in figure 2.7. These figures show that even though the VI-sensor experiences velocities up to $3\,\mathrm{m\,s^{-1}}$, there is no significant drift from the lateral motion visible in the recovered path (the perpendicular motion was to stop the recording). Figure 2.9 is a screenshot from the output of ROVIO. This image shows that definite blurring occurs during these fast motions. But even for this blur, ROVIO is able to extract the relative motion. Thus, motion blur appears to have no significant effect on the visual odometry process.

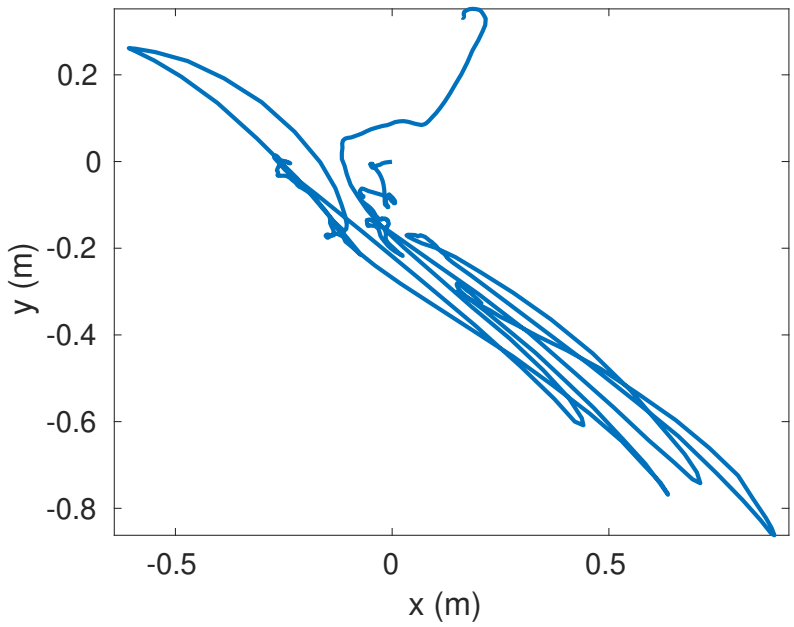


Figure 2.6: 2D position of VI-sensor during motion blur test. The sideways motion displays no extreme drift. (The perpendicular motion is to stop the recording.)

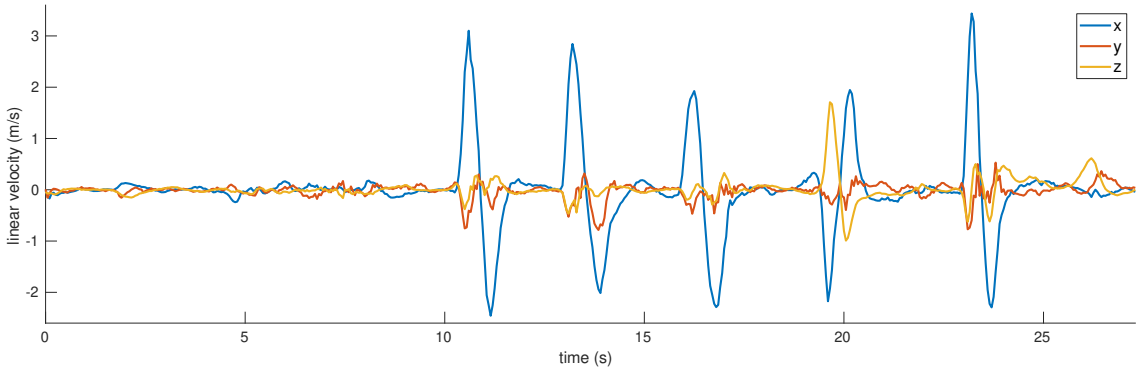


Figure 2.7: Linear velocity of VI-sensor vs. time during motion blur test.

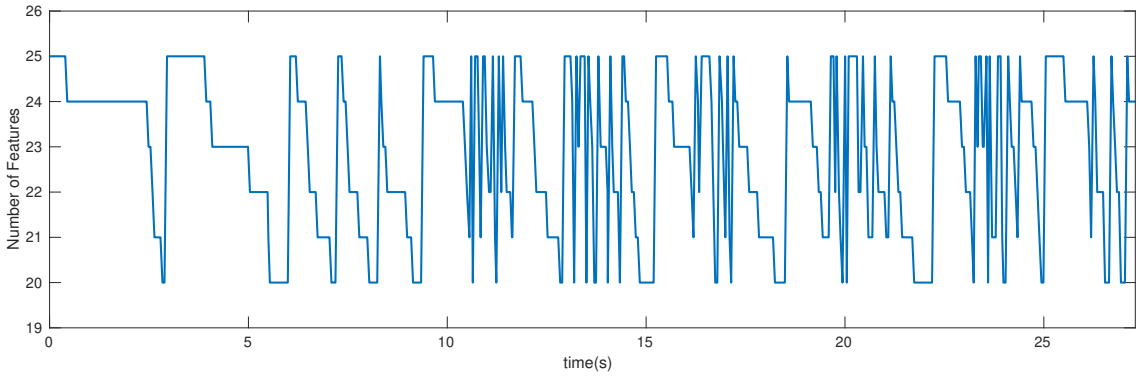


Figure 2.8: Number of features vs. time during motion blur test.

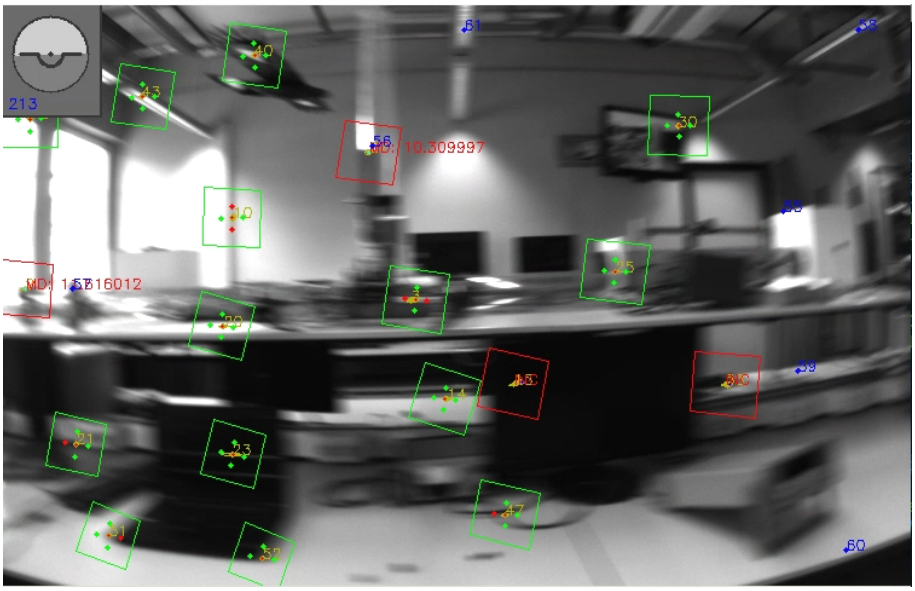


Figure 2.9: Screenshot output ROVIO with motion blur.

Another significant effect is the loss of information from either the IMU or the camera. In the case of the IMU, a loss of information can occur when there is a lot of noise, i.e. vibrations. The effect of that was tested by moving the camera while shaking it. The result is visible in figure 2.10. It can be seen that due to the shaking, the estimated trajectory quickly diverges from the actual trajectory (recorded with a motion-capture system). This divergence can be due to both introduced noise in the IMU data and/or an induced motion blur in the video data.

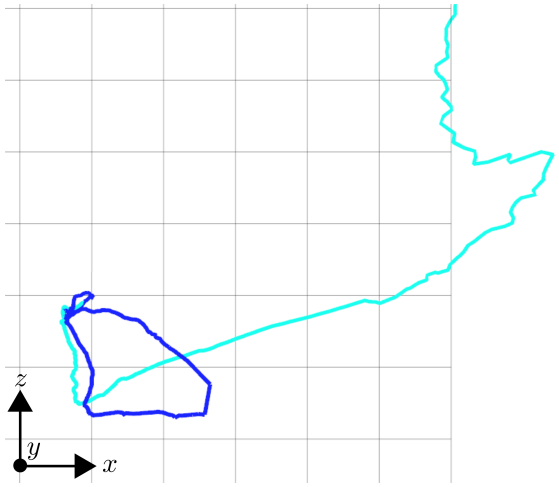


Figure 2.10: Actual trajectory from motion-capture system (darkblue) vs. estimated trajectory (light-blue), while vibrations were introduced.

In the case of the camera, a loss of information can occur when there are no features visible in frame. This effect was evaluated by moving the VI-sensor along a hallway on a cart. The VI-sensor was oriented perpendicular to the cart. Between the start and the end of the trajectory, an approximate constant lateral velocity was manually maintained from $t \approx 8\,\mathrm{s}$ to $34\,\mathrm{s}$. In figure 2.11 the number of features taken into account by ROVIO and the estimated lateral velocity are displayed. It is visible that from the second patch without features, the estimated velocity completely diverges from the approximate constant velocity. It is also visible that when the featureless patches end, the algorithm attempts recovering the tracking. However, the algorithm completely overcompensates, resulting in the final velocity being an unrealistic value for a hand-driven cart. Moreover, the carts actual velocity was reduced to zero in the end phase, which ROVIO was not able to comprehend. This discrepancy shows the dependency of the visual odometry algorithm on the presence of features.

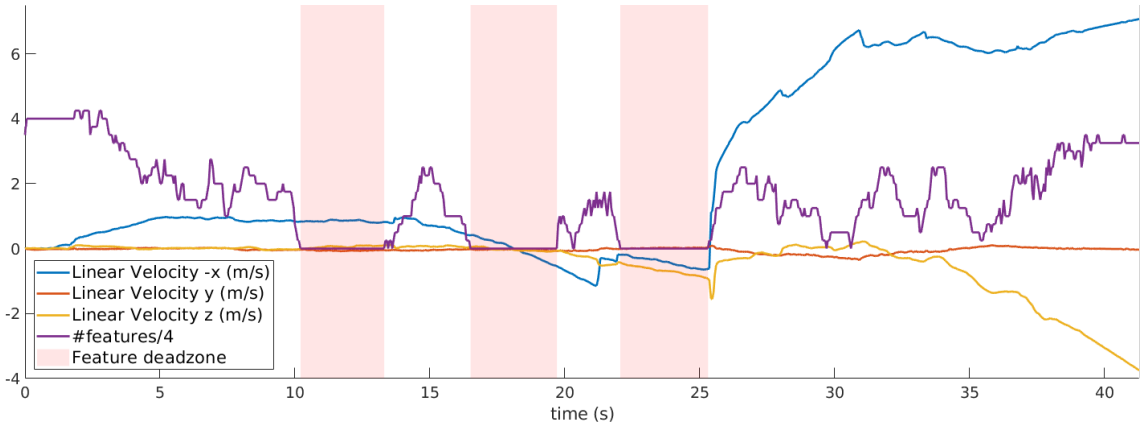


Figure 2.11: Number of used features and estimated linear velocities during hallway experiment. Note that the $-x$ direction w.r.t the camera frame is plotted since that was the forward direction for the cart. The shaded areas are the areas where no features were extracted by ROVIO, i.e. white patches of wall.

3 Modelling & Control Design

This chapter will elaborate on the design of the gaze control algorithm.

The information flow pipeline in the visual odometry process is as displayed in figure 3.1.

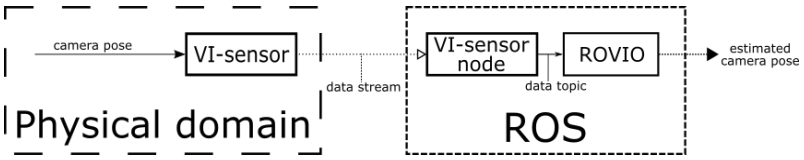


Figure 3.1: Information flow pipeline - visual odometry

The purpose of the the gaze control node is to improve the information content of the data that is used as an input for the visual odometry algorithm. To do this, a closed loop is introduced, see figure 3.2. In the physical domain, a pair of servomotors is included to enable pan-tilt actuation of the camera pose (see figure 3.3a). In the software domain of ROS, ROVIO is adapted to add an extra output which relays the location of extracted features with respect to the camera frame Ψ_c. Then this new information is used in the designed gaze control node to adjust the setpoints of the pan-tilt actuator towards a more favourable pose.

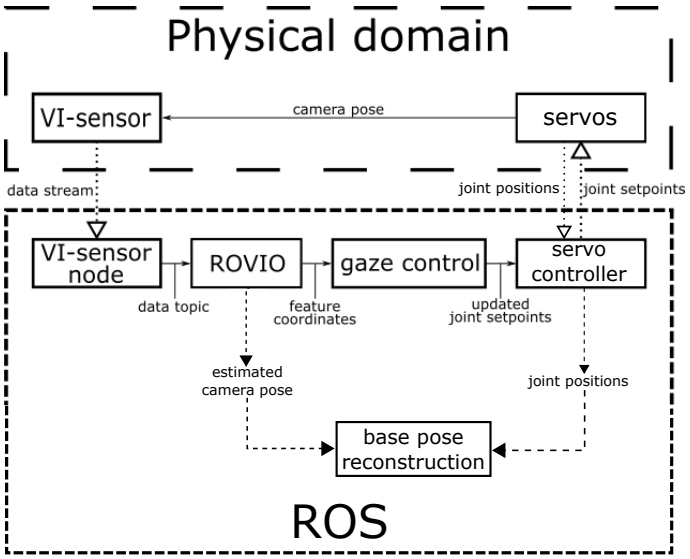


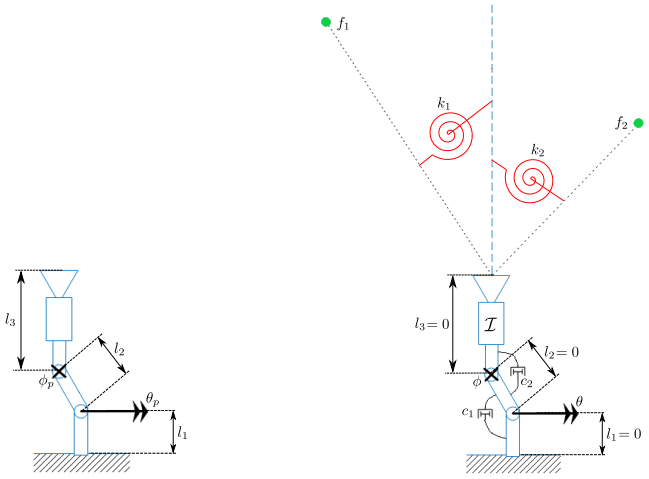
Figure 3.2: Manipulated information flow pipeline - visual odometry

The setpoints of the pan-tilt actuator are manipulated by means of a virtual, dynamical model which simulates inertias, springs and damping to achieve a desired motion (see figure 3.3b). In this model, the virtual angles ϕ and θ form the setpoints for the physical servo angles ϕ_p and θ_p. How the virtual model is constructed is explained in the rest of the chapter.

Figure 3.2 also shows that feedback from the servos combined with the estimated camera pose from ROVIO is used to reconstruct the pose of the base. (see section 3.5)

3.1 Virtual model

The basis of the virtual, dynamical model is the creation of virtual springs between the features and the camera frame. These virtual springs create a resultant force vector that is inherently directed towards a higher concentration of features. The force exerted by the virtual springs is transformed into a continuous motion by applying this virtual force on a virtual inertia $\mathcal{I}$, while



(a) Physical system **(b)** Virtual model

Figure 3.3: Schematic representations

simultaneously ensuring that the acceleration is limited and finite. To ensure that this motion converges, virtual friction ($c_i, i = 1,2$) is included on the joints. A schematic representation of this model is displayed in figure 3.3b, in which the features are represented by the green circles, the virtual springs by the red springs that are attached to the center line of the VI-sensor and the rays from the camera origin through the feature locations. Additionally, dampers representing friction are modelled on the joints. The joint axes are oriented perpendicular to each other.

Although the physical links have to constitute a finite length ($l_i > 0, i = 1\ldots3$), these lengths actually contribute little to this model, as the link lengths are generally orders of magnitude smaller than the distance to the features ($l_i \ll d$). Therefore, these lengths are neglected ($l_i = 0$), which simplify the model of the robot arm to a single, actuated 2-DOF joint. However, the link lengths have to be taken into account for the base pose reconstruction.

As the pan-tilt model only has two rotational degrees-of-freedom, the Euclidean distance of a feature with respect to the camera does not provide a useful contribution to the spring force. Therefore, the virtual springs are modelled as torsional springs, which depend on the angle with respect to the center line of the camera, instead of their position from the camera origin.

3.2 Rigid body model

The state of a rigid body is the current configuration captured in the homogeneous matrix H_k^w. The change in state can be derived from the definition of the twist:

$$\tilde{T}_k^{k,w} = H_w^k \dot{H}_k^w \tag{3.1}$$

Rewriting yields

$$\frac{\mathrm{d}}{\mathrm{d}t} H_k^w(t) = H_k^w(t) \cdot \tilde{T}_k^{k,w}(t) \tag{3.2}$$

Using this equation, the state can be continuously updated by integrating the twist $T_k^{k,w}$. This twist can be found by using Newton's second law of motion. Newton's second law of motion applied to 6-dimensional rigid bodies can be described using screw theory[24, p. 32]:

$$\left(\dot{\mathcal{P}}^k\right)^\top = \mathcal{I}^k \cdot \dot{T}_k^{k,w} = \mathrm{ad}_{T_k^{k,w}}^\top \left(\mathcal{P}^k\right)^\top + \left(W^k\right)^\top \tag{3.3}$$

with

$$\mathrm{ad}_{T_k^{k,w}} = \begin{pmatrix} \tilde{\omega}_k^{k,w} & 0_{3\times3} \\ \tilde{v}_k^{k,w} & \tilde{\omega}_k^{k,w} \end{pmatrix}$$

where:

- $\mathcal{P}^k$ = 6-dimensional momentum expressed in frame Ψ_k.
- $\mathcal{I}^k$ = inertia matrix expressed in frame Ψ_k.
- $T_k^{k,w}$ = twist from Ψ_k relative to Ψ_w, expressed in Ψ_k.
- W^k = wrench expressed in frame Ψ_k.
- $\mathrm{ad}_{T_k^{k,w}}$ = the adjoint matrix of the twist $T_k^{k,w}$.
- $\tilde{\omega}_k^{k,w}$ = the skew-symmetric matrix of the rotational velocity vector from Ψ_k relative to Ψ_w, expressed in Ψ_k, which is contained in the twist $T_k^{k,w}$.
- $\tilde{v}_k^{k,w}$ = the skew-symmetric matrix of the linear velocity vector from Ψ_k relative to Ψ_w, expressed in Ψ_k, which is contained in the twist $T_k^{k,w}$.

This equation describes the motion of a rigid body in three-dimensional space in a body-fixed frame Ψ_k. If Ψ_k is aligned with the principal axes of the body, $\mathcal{I}^k$ becomes a diagonal matrix.

To be able to use equation (3.3) as a component in a larger construction, it is convenient to transform the twists and wrenches to a common inertial reference frame Ψ_w. In order to do that, the Adjoint of the homogeneous transform matrix H_k^w can be used:

$$T_k^{w,w} = \mathrm{Ad}_{H_k^w} T_k^{k,w} \qquad \left(W^k\right)^\top = \mathrm{Ad}_{H_k^w}^\top \left(W^w\right)^\top \tag{3.4}$$

where the Adjoint has the form:

$$\mathrm{Ad}_{H_k^w} = \begin{pmatrix} R_k^w & 0_{3\times3} \\ \tilde{p}_k^w R_k^w & R_k^w \end{pmatrix} \tag{3.5}$$

As the model in figure 3.3b can only rotate about two axes, four of the six degrees-of-freedom of free space have to be constrained. These four degrees-of-freedom are the roll rotation (around z-axis of camera frame), and the x, y and z-translations. The effect of a joint is to constrain the relative motion between two frames in such a way that only motion in the direction of the degrees-of-freedom can be sustained. The forces in the other directions have to be counteracted to reduce them to zero as quick as possible. A straightforward approach to this is to model a spring-damper system between the two frames for the constrained degrees of freedom. The first step in modelling this interconnection is to calculate the relative twist between two frames:

$$T_A^{w,B} = T_A^{w,w} - T_B^{w,w} \tag{3.6}$$

Using this relative twist, reaction forces can be calculated using the constitutive relations of a spring and damper:

$$\left(W_{\text{spring}}^w\right)^\top = K_{\text{cstr}} \cdot \Delta\vec{x} = K \cdot \int T_A^{w,B} \mathrm{d}t \tag{3.7}$$

$$\left(W_{\text{damper}}^w\right)^\top = R_{\text{cstr}} \cdot T_A^{w,B} \tag{3.8}$$

where K_{cstr} and R_{cstr} are diagonal matrices containing a high spring constant and a high damping factor for the constrained degrees-of-freedom, creating a stiff response. These reaction forces can be summed and applied back to the rigid body in the direction opposite to the twist, counteracting the relative motion.

Similarly, the two degrees-of-freedom are subjected to viscous friction through an equation similar to eq. 3.8, where $R_{fric} \ll R_{cstr}$:

$$\left(W_{\text{friction}}^w\right)^\top = R_{\text{fric}} \cdot T_A^{w,B} \tag{3.9}$$

3.3 Virtual springs

Motion of the robot arm is obtained by modelling virtual torsional springs between the feature locations and the camera frame Ψ_c. The torque exerted by these springs on the camera is easiest expressed in the local frame, Ψ_c, because the z-axis of that frame coincides with the center of the obtained camera image.

These torsional springs will be subjected to three different penalties to compensate for unwanted effects.

The following sections will explain how the spring torque of the virtual springs is calculated and how the different penalties are constructed and subsequently applied to this spring torque.

3.3.1 Torsional Spring Torque of a Single Feature

The angular error around $\hat{y}^c$, $\epsilon_{\hat{y}^c}$, and the angular error around $\hat{x}^c$, $\epsilon_{\hat{x}^c}$, can both be determined using the arctangent function. A feature point with its respective projections on the three Cartesian axes of Ψ_c form a set of triangles (see figure 3.4), which can be used to determine the respective angular errors. However, because a positive rotation around $\hat{x}^c$ results in a motion towards the negative $\hat{y}^c$-axis (in a right-handed coordinate system), y_f^c should actually be reflected in the xz-plane, to obtain the proper opposite side of the triangle for consistent conversion.

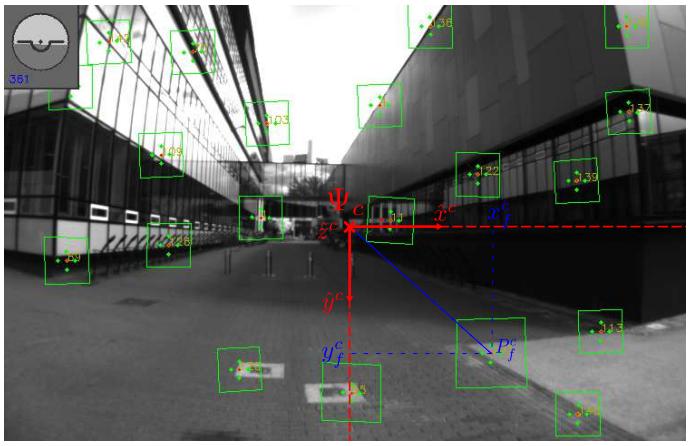


Figure 3.4: Camera frame

Then using the 4 quadrant arctangent function `atan2`, the angular errors can be obtained:

$$\epsilon_{\hat{x}^c} = \texttt{atan2}\left(\frac{-y_f^c}{z_f^c}\right) \qquad \epsilon_{\hat{y}^c} = \texttt{atan2}\left(\frac{x_f^c}{z_f^c}\right) \tag{3.10}$$

$$W^c = K_s(\cdot)\begin{pmatrix}\epsilon_{\hat{x}^c}\\ \epsilon_{\hat{y}^c}\\ 0_{4\times 1}\end{pmatrix} \tag{3.11}$$

where K_s is a 6×6 diagonal matrix.

Field-of-view

The camera will have a limited field-of-view. This field-of-view has to be taken into account in simulation. This limitation can be simulated by requiring that a feature is within a certain angular range; the effect of a feature is only calculated when this requirement is satisfied.

3.3.2 Penalties to the Spring Torque

Three penalties will be applied to the spring torque: a penalty α to compensate for the distance of a feature w.r.t. Ψ_c , a penalty β that takes into account the physical limits of the robot arm, and a penalty γ that creates an initialization phase for each feature. The reasoning behind each and their construction will be explained below. The torsional spring stiffness matrix K_s will then be calculated as:

$$K_s(\alpha,\beta,\gamma) = \begin{bmatrix} \alpha\beta_{1,1}\gamma k_1 & 0 & \\ 0 & \alpha\beta_{2,2}\gamma k_2 & 0_{2\times 4} \\ & 0_{4\times 6} & \end{bmatrix} \tag{3.12}$$

Position estimation quality degrades as features move away, as the ratio between the stereo base-line and the distance becomes insignificant. Also, if features move too close to the camera, pose estimation quality degrades because features remain visible in the camera's field-of-view for a shorter amount of time. To take this into account in the virtual spring model, the spring constant K is penalized with respect to the Euclidean distance d from the camera (in meters) with the first penalty α (see also figure 3.5):

$$\alpha(d\{m\}) = \begin{cases} d & 0 \le d < 1 \\ 1 & 1 \le d < 20 \\ 1 - 0.045 \cdot (d-20) & 20 \le d \ge 40 \\ 0.1 & d > 40 \end{cases} \tag{3.13}$$

This penalty is chosen such that it emphasizes features in an intermediate operating range (not too close-by, not too far away), but does not entirely exclude the features outside of this range. From the experiment in section 2.3, the effect of the distance was clearly visible from a distance of 30 m, so from a distance of 20 m a steep decline in weight starts until a distance of 40 m, after which a small weight remains for further distances in order to not completely disable their influence, for the specific situation that only distant features remain.

The lower limit is a choice. The distance to a feature and the velocity of the camera determine how quick a feature will move out of view. By implementing a lower limit, close features, which will relatively move out of view quickly, are not taken into account as much as features that remain in view for a longer period of time. The lower limit is chosen to be 1 meter.

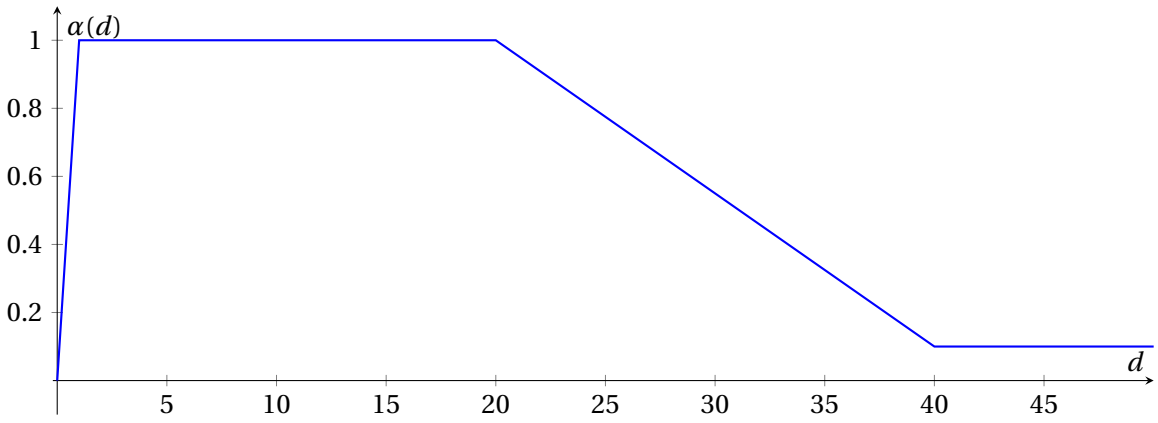


Figure 3.5: Distance penalty α

Limits to the Robot Arm

In order to take into account the physical limits that are inherent in the design of the robot arm, a second weighting factor β is introduced that inversely scales the spring force for each feature as a function of its angular distance from the physical limits. β is a diagonal 2×2 matrix, where the diagonal elements represent the weighting factor for each joint. To calculate the absolute angle of a feature w.r.t. Ψ_w a new set of compound angles $\vec{\eta}$ has to be introduced:

$$\vec{\eta} = \begin{pmatrix} \eta_{\hat{x}^c} \\ \eta_{\hat{y}^c} \end{pmatrix} = \begin{bmatrix} \phi + \epsilon_{\hat{x}^c} \\ \theta + \epsilon_{\hat{y}^c} \end{bmatrix} \tag{3.14}$$

To err on the side of caution, a virtual limit of $\frac{1}{2}\pi$ rad from the neutral position for each joint is chosen. To penalize angles closer to the limits more than in the center, a polynomial shape for $\beta_{i,i}$ is taken. This makes $\beta_{i,i}$:

$$\beta_{i,i}(\eta_i) = \begin{cases} 1 - \left(\frac{\eta_i}{\frac{1}{2}\pi}\right)^6 & |\eta_i| < \frac{1}{2}\pi \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

with $i = 1,2$.

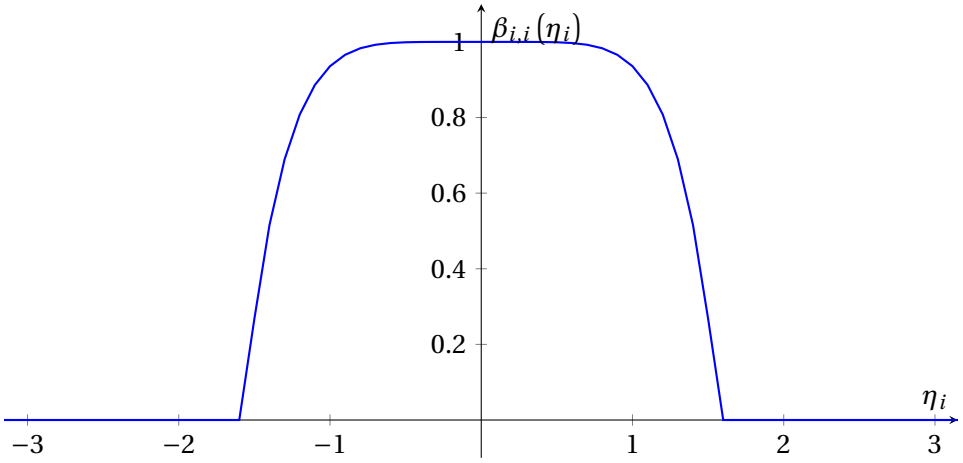


Figure 3.6: Physical limit penalty for one joint $\beta_{i,i}$

Feature initialization

The field-of-view of the camera combined with the fact that the features are connected by virtual torsional springs leads to an undesired side-effect: as soon as a feature enters the field-of-view, the force exerted by the virtual spring starts at its maximum value. In other words, each time the camera moves, a chain of step-responses is obtained. This can lead to unstable behaviour. To limit this undesirable behaviour, an initialization phase for each spring is introduced. This means that when a feature enters the field-of-view, it is assigned an age t_f, and a third spring torque penalty, dubbed γ, is linearly increased until a certain maturation age t_m is reached, where γ will reach and maintain a value of 1. Thus, this makes γ:

$$\gamma = \begin{cases} \frac{1}{t_m} t_f & t_f < t_m \\ 1 & t_f \geq t_m \end{cases} \tag{3.16}$$

3.4 Calculation of servo setpoint from camera pose

The camera pose is encoded in the H_c^w matrix. The orientation of the camera is expressed in a rotation matrix R_c^w included in H_c^w:

$$R_c^w = H_c^w[1:3, 1:3] \tag{3.17}$$

A rotation matrix can be constructed from Euler angles. The definition of Euler angles is to describe any three-dimensional rotation by composing 3 elemental rotations. The order of these rotations is important, because it determines the final orientation that is obtained. There are multiple combinations of rotation orders possible. In this research, the $X-Y'-Z''$[1] order of rotations is chosen, because of their intuitive representation with respect to the camera frame, and the fact that the Z'' axis will not be actuated. Euler angles in this order are also called intrinsic rotations of the Tait-Bryan angles or the Cardan angles.

The rotation matrix for this order of rotations is constructed with the following formula:

$$R_c^w = X(\phi)Y(\theta)Z(\psi) \tag{3.18}$$

$$= \begin{bmatrix} \cos(\theta)\cos(\psi) & -\cos(\theta)\sin(\psi) & \sin(\theta) \\ \cos(\phi)\sin(\psi)+\cos(\psi)\sin(\phi)\sin(\theta) & \cos(\phi)\cos(\psi)-\sin(\phi)\sin(\theta)\sin(\psi) & -\cos(\theta)\sin(\phi) \\ \sin(\phi)\sin(\psi)-\cos(\phi)\cos(\psi)\sin(\theta) & \cos(\psi)\sin(\phi)+\cos(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{3.19}$$

Using reverse engineering, the values of the angles can be reconstructed:

1. $R_{13} = \sin(\theta)$, thus $\theta_1 = \sin^{-1}(R_{13})$, but $\theta_2 = \pi - \theta_1 = \pi - \sin^{-1}(R_{13})$ is also a valid solution. However, only θ_1 is relevant, since θ_2 is per definition outside of the reachable half-sphere.

2. Note that $-\frac{R_{23}}{R_{33}} = \tan(\phi)$. This fact can be used to solve for ϕ: $\phi = \texttt{atan2}(R_{23}, R_{33})$. However, the sign of $\cos(\theta)$ affects the sign of R_{23} and R_{33}. This can be consistently taken into account by including $\cos(\theta)$ in the calculation of ϕ (as long as $\cos(\theta) \neq 0$):

$$\phi = \texttt{atan2}\left(\frac{-R_{23}}{\cos(\theta)}, \frac{R_{33}}{\cos(\theta)}\right) \tag{3.20}$$

3. In the specific case of gimbal lock ($\theta = \pm\frac{1}{2}\pi$), $\cos(\theta) = 0$, which means R_{32} cannot be used. In the case $\theta = +\frac{1}{2}\pi$, consider $R_{21} = \cos(\phi)\sin(\psi) + \cos(\psi)\sin(\phi)$. Through the trigonometric angle sum identity: $R_{21} = \sin(\phi+\psi)$. Similarly, $R_{31} = -\cos(\phi+\psi)$, $R_{22} = \cos(\phi+\psi) = -R_{31}$ and $R_{32} = \sin(\phi+\psi) = R_{21}$. These equations can then be used to find the relation between ϕ and ψ:

$$(\phi+\psi) = \texttt{atan2}(R_{21}, R_{22}) \tag{3.21}$$

$$\phi = -\psi + \texttt{atan2}(R_{21}, R_{22}) \tag{3.22}$$

   Since there is an infinite number of solutions in the case of gimbal lock, ψ can be chosen to be zero, from which ϕ can be calculated.

4. In case $\theta = -\frac{1}{2}\pi$, in a similar fashion, the relation between ϕ and ψ can be derived as:

$$(\phi-\psi) = \texttt{atan2}(-R_{21}, -R_{22}) \tag{3.23}$$

$$\phi = \psi + \texttt{atan2}(-R_{21}, -R_{22}) \tag{3.24}$$

   And again, ψ can be chosen to be zero, from which ϕ can be derived.

[1] Y' indicates here the y-axis of the rotated frame.

3.5 Base position calculation

The base pose H_b^w at any moment is related to the camera pose H_c^w at that moment by:

$$H_b^w = H_c^w H_b^c \tag{3.25}$$

The inverse of H_b^c, H_c^b, can be derived from the servo angles q_i, and is an application of forward kinematics, for which Brockett's exponential formula[24, p. 38] can be used:

$$H_c^b(q_1, q_2) = e^{\hat{\hat{T}}_1^{0,0} q_1} e^{\hat{\hat{T}}_2^{0,1} q_2} H_c^b(0) \tag{3.26}$$

where:

$$\begin{aligned}
\omega_1 &= \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^\top \\
r_1 &= \begin{bmatrix} 0 & 0 & l_1 \end{bmatrix}^\top \\
\hat{\hat{T}}_1^{0,0} &= \begin{bmatrix} 0 & 1 & 0 & l_1 & 0 & 0 \end{bmatrix}^\top \\
\omega_2 &= \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^\top \\
r_2 &= \begin{bmatrix} 0 & 0 & l_1 + l_2 \end{bmatrix}^\top \\
\hat{\hat{T}}_2^{0,1} &= \begin{bmatrix} 1 & 0 & 0 & 0 & -(l_1 + l_2) & 0 \end{bmatrix}^\top \\
H_c^b(0) &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_1 + l_2 + l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

4 Simulations

4.1 Simulation model

To aid the design process of the virtual model, a simulation of this model is made in 20-sim, using bond graphs[4, 5].

> *"Bond graphs are a domain-independent graphical description of dynamic behaviour of physical systems."*[5]

The complete model is shown in figure 4.1. This model consists of a rigid body submodel for the camera, a joint model and the `multispring` submodel. The camera submodel is displayed in figure 4.2, and shows a conversion (**MTF**) from the local camera frame Ψ_c to the inertial frame Ψ_w using the Adjoint matrix of H_w^c. Equations 3.4 and 3.5 are contained in this block, as well as equations 3.1 and 3.2. In the local frame, an inertia element I_c simulating the left-hand side of equation 3.3 and a modulated gyristor **MGR** simulating the ficticious forces, i.e. the first term from the right-hand side of equation 3.3. The last term $\left(W^k\right)^\top$ represents external forces, which in this case are the torques applied by the virtual spring via `p_spring`.

The joint model is displayed in figure 4.3, which contains the constraint equations 3.7 and 3.8 applied in the local frame Ψ_c. For simulation purposes, these constraint equations are split up into scalar relations. That way, the constraints can be calculated in a different order, which prevents differential causality for the camera inertia. The joint model connects the camera frame to the base frame Ψ_b. The values for the constraint parameters are $K_{\text{cstr}} = 1000\,\text{N}\,\text{m}^{-1}, R_{\text{cstr}} = 10\,\text{N}\,\text{s}\,\text{m}^{-1}$.

The `multispring` submodel contains all calculations explained in 3.3. The `H2Euler` contains the servo setpoint calculation explained in 3.4. The `viewline` submodel is for visualization purposes.

The `Features` submodel contains the locations of each simulated feature in world frame Ψ_w: P_f^w. For equation 3.10 are the feature coordinates expressed in the camera frame required P_f^c. To obtain P_f^c, the feature point coordinates need to be transformed from world frame to camera frame, using the homogeneous matrix:

$$P_f^c = H_w^c \cdot P_f^w \tag{4.1}$$

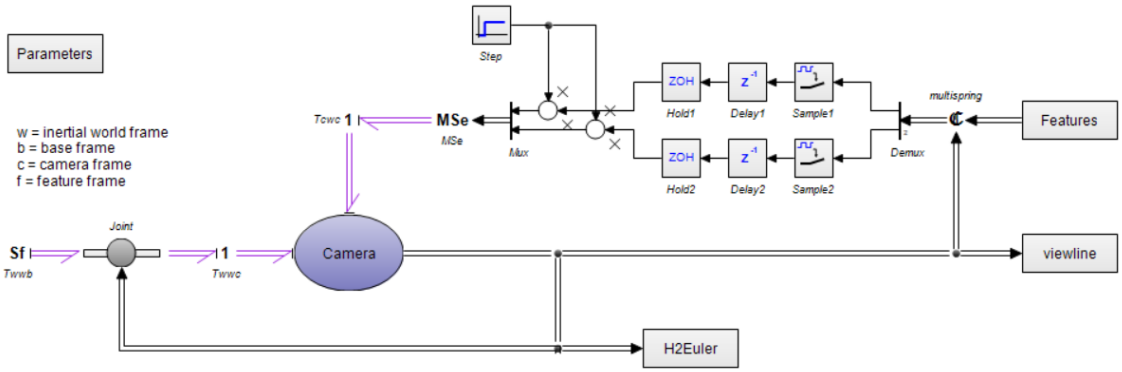


Figure 4.1: Complete bond graph model.

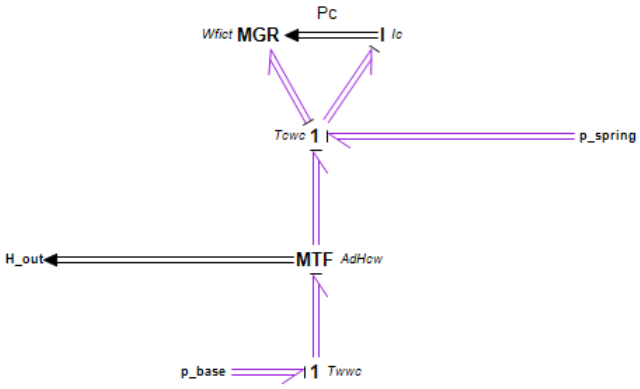


Figure 4.2: Rigid body bond graph model of the camera.

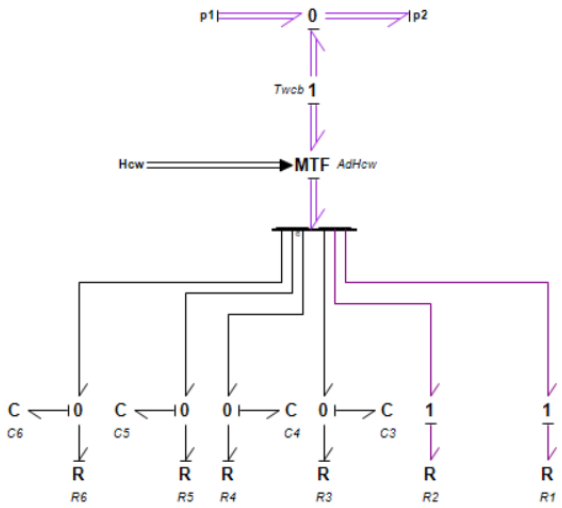


Figure 4.3: Joint bond graph model.

4.2 Tuning

The basic model without the scaling factors and the field-of-view implemented is essentially a mass-spring-damper system (figure 4.4 in the rotational domain. For the sake of simplicity, the first iteration is tested with a single feature. That is analogous to the system in figure 4.4, but with $F = 0$ (free vibration) and the initial displacement in a non-equilibrium position (initial condition is non-zero). This analogy can be used to determine the preferred tuning of the system parameters.

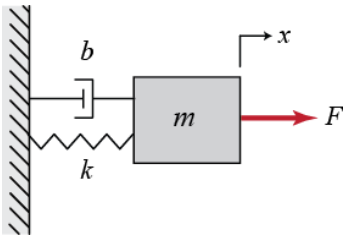


Figure 4.4: Mass-Spring-Damper system[16]

The mass in the three-dimensional rotational system is the inertia matrix $\mathcal{I}_k$. This inertia matrix is modelled similar to the physical VI-sensor by modelling it as a solid cuboid[27], combined with the parallel-axis theorem to account for the physical length of the robot arm, with the properties:

- width = 0.145 m
- height = 0.065 m
- depth = 0.06 m
- mass = 0.253 kg
- distance axis 1: $l_1 + l_2 = 0.1165$ m
- distance axis 2: $l_1 = 0.052$ m

For a mass-spring-damper system with critical damping ($c^2 = 4km$), the free response to a non-zero initial condition x_0 (i.e. a step) is[10]:

$$x(t) = x_0 e^{-\alpha t} + (\alpha x_0)\, t e^{-\alpha t} \tag{4.2}$$

$$\text{with: } \alpha = \frac{c}{2m} \tag{4.3}$$

$$\dot{x}(t) = -\alpha x_0 e^{-\alpha t} + (\alpha x_0)\left(e^{-\alpha t} - \alpha t e^{-\alpha t}\right) \tag{4.4}$$

$$= -\alpha^2 x_0 t e^{-\alpha t} \tag{4.5}$$

$$\ddot{x}(t) = -\alpha^2 x_0 e^{-\alpha t} + \alpha^3 x_0 t e^{-\alpha t} \tag{4.6}$$

$$\ddot{x}\left(t_{v_{\max}}\right) = 0 \rightarrow t_{v_{\max}} = \frac{1}{\alpha} = \frac{2m}{c} \tag{4.7}$$

$$\dot{x}\left(t_{v_{\max}}\right) = -\alpha x_0 e^{-1} = -\frac{c x_0}{2m} e^{-1} = v_{\max} \tag{4.8}$$

Assuming the maximum allowed initial position (i.e. $x_0 = \frac{1}{2}\pi$), the maximum rotational velocity to be 59 `rpm` = 6.18 rad s^{-1}, and the inertia of the camera, the damping coefficient can be determined as:

$$c_i = v_{\max} \frac{2\mathcal{I}_k[i,i]}{x_0 e^{-1}} \tag{4.9}$$

with $i = 1, 2$ for the corresponding degree-of-freedom.

Then, using that c_i, the corresponding k_i can be calculated through the critical damping relation:

$$k_i = \frac{c^2}{4\mathcal{I}_k[i,i]} \tag{4.10}$$

To ensure that the response becomes underdamped due to an accumulation of k_i for an increasing number of features, a total spring constant is defined, which takes the value of k_i. The individual spring constants $k_{f,i}$ are then a fraction of that total spring constant:

$$k_{f,i} = \frac{k_i}{n} \tag{4.11}$$

where n is the maximum amount of features that can affect the camera. In the 20-sim simulation this n is a predefined number. In physical experiments this is also a predefined number, because ROVIO has an internal parameter that determines the maximum amount of features that will be taken into account at any moment. The effect of this normalization process is that the response can be at most critically damped, but will be overdamped in most instances.

Using the parameters found in this section, the response to various situations is simulated.

4.2.1 Single point response

The first situation that is tested is the simplest non-trivial situation possible: a single non-zero point in three-dimensional space. This feature is chosen to be:

$$p_f^w = \begin{pmatrix} 4.02 \\ 2.58 \\ 3.5 \end{pmatrix} \tag{4.12}$$

such that the location of this feature is just within the field-of-view of the camera. (Even though the field-of-view functionality is not enabled in this specific scenario, taking this into account improves the comparability with subsequent scenarios.)

The spring force is multiplied with a unit step that starts at a time $t = 0.2\,\text{s}$, to show the step response clearly and allow an initialization period for the integration method.

The step responses are simulated using a 4th-order Runge Kutta integration method with a step size of 0.001 s. This simulation is considered a decent approximation of continuous-time, because of the small step size. The simulated step responses are visible in figures 4.5, 4.6 and 4.7. Figure 4.5 shows an overdamped step response with a settling time of approximately 0.5 s. Figure 4.6 shows that the designed damping and spring parameters satisfy the velocity limit of the Dynamixel servo. Figure 4.7 shows a convergence of the camera angles towards the feature. These angles are calculated using the theory of section 3.4. The final tilt angle equals the expected value of $\arctan\left(\frac{2.59}{3.5}\right) \approx 0.637\,\text{rad}$. However, the final pan angle is $\theta(\infty) \approx 0.747\,\text{rad}$ which is lower than the expected value of $\arctan\left(\frac{4.03}{3.5}\right) \approx 0.856\,\text{rad}$. This is most likely due to the fact that moving both angles simultaneously requires less energy than moving each angle individually. When this hypothesis is tested by placing the feature at: $p_f^w = \begin{pmatrix} 4.03 & 0 & 3.5 \end{pmatrix}^\top$, the final pan angle indeed becomes $\theta(\infty) \approx 0.856\,\text{rad}$.

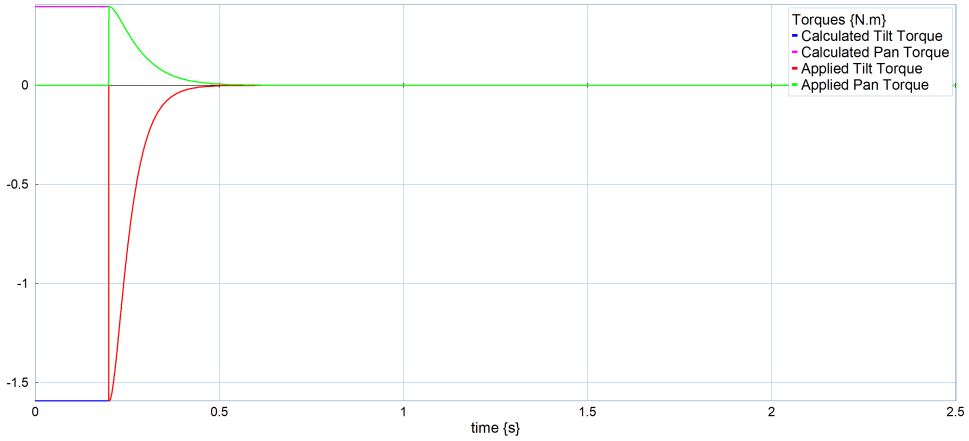


Figure 4.5: Simulated torque response in continuous-time to a single feature.

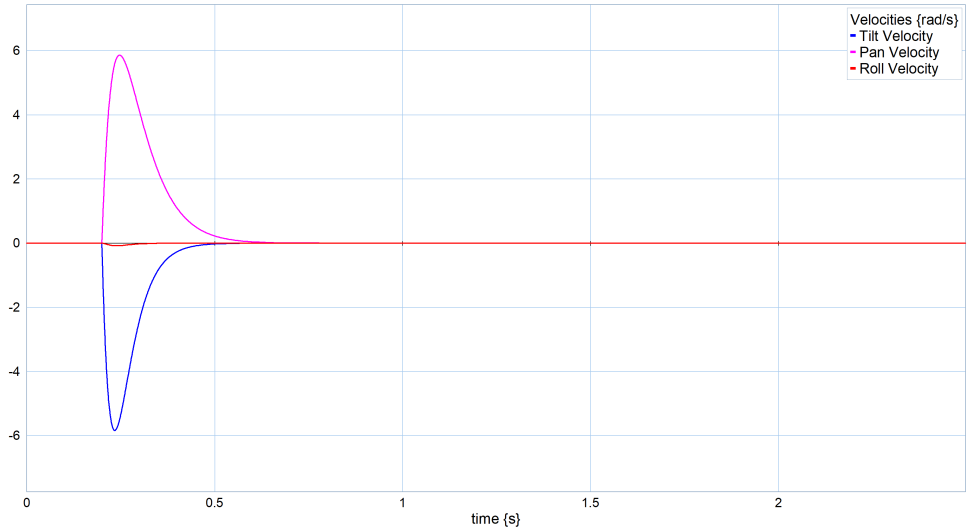


Figure 4.6: Simulated velocity response in continuous-time to a single feature.

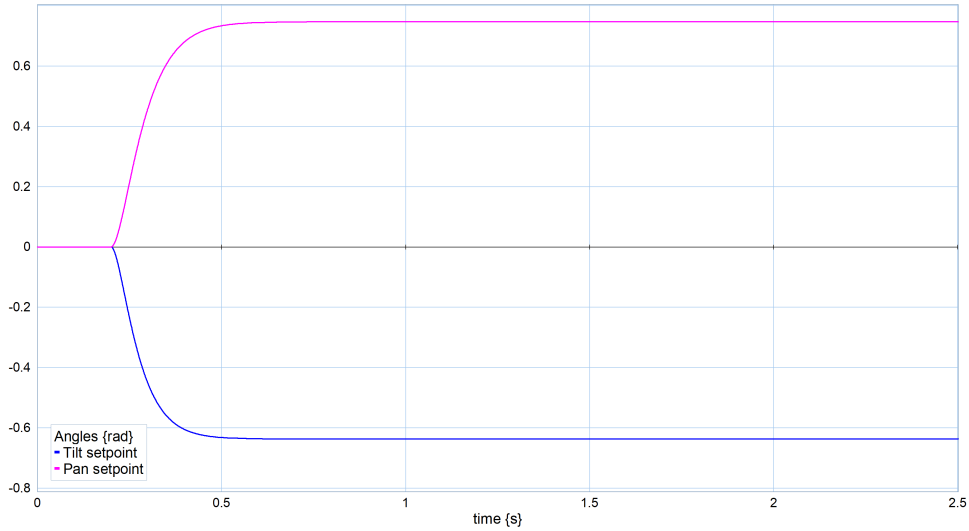


Figure 4.7: Simulated camera angles response in continuous-time to a single feature.

Continuous-time simulation in the practical set-up is unrealistic, since the ROVIO algorithm runs with a frequency of 20 Hz in standard configuration. To increase the correspondence between the 20-sim simulation and the practical set-up, a sample-and-hold mechanism is introduced in all following models to discretize the calculation of the virtual spring torque. Additionally, in this discrete-time domain, a unit delay is included to account for the processing time ROVIO and the gaze-control algorithms need to calculate a new iteration of spring torques. The step responses from this new model are visible in figures 4.8, 4.9 and 4.10. Figure 4.8 shows the torque (in N m) that is calculated by the rotational spring model and the torque that is subsequently applied to the rigid body after the discretization process. Figure 4.9 shows the camera velocities as a result of the applied torques. For each sample, a (sub-)step response is visible. Figure 4.10 shows the changing Cardan angles of the camera frame. The figures indicate that due to the discretization process, the system becomes underdamped.

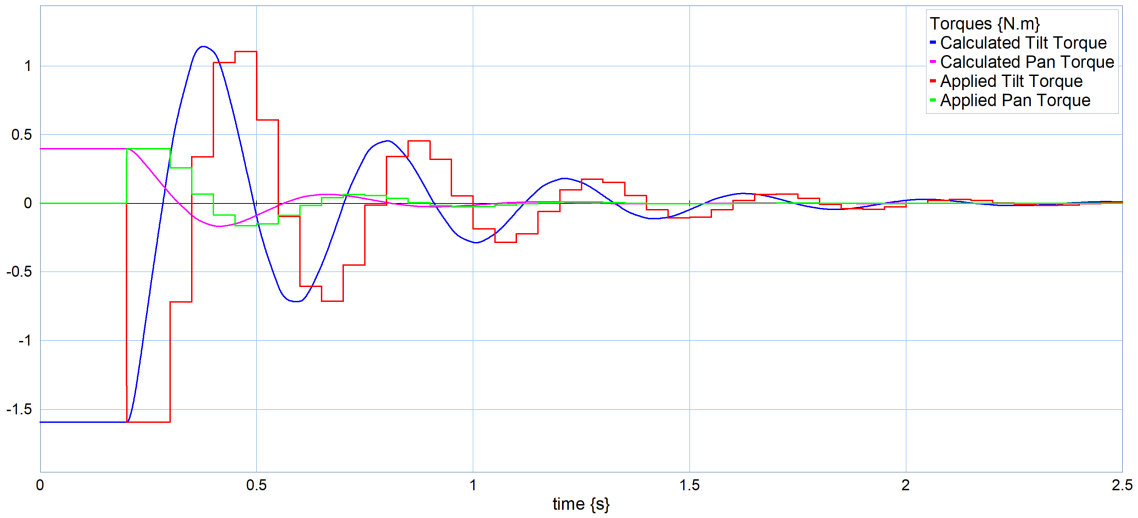


Figure 4.8: Simulated torque response to a single feature, using the continuous damping value.

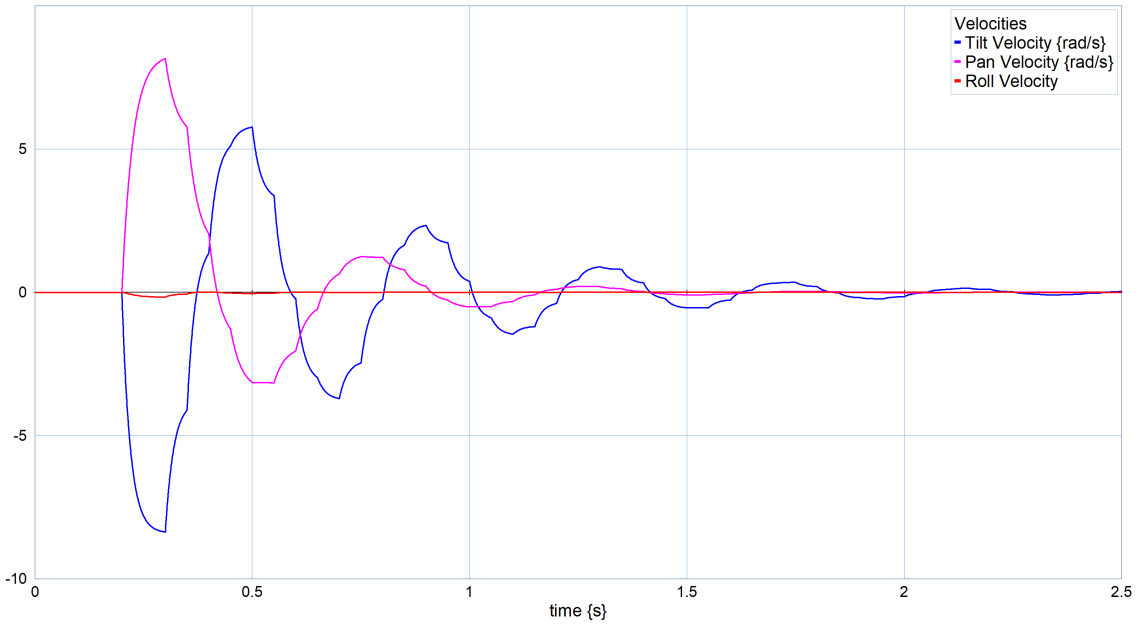


Figure 4.9: Simulated velocity response to a single feature, using the continuous damping value.

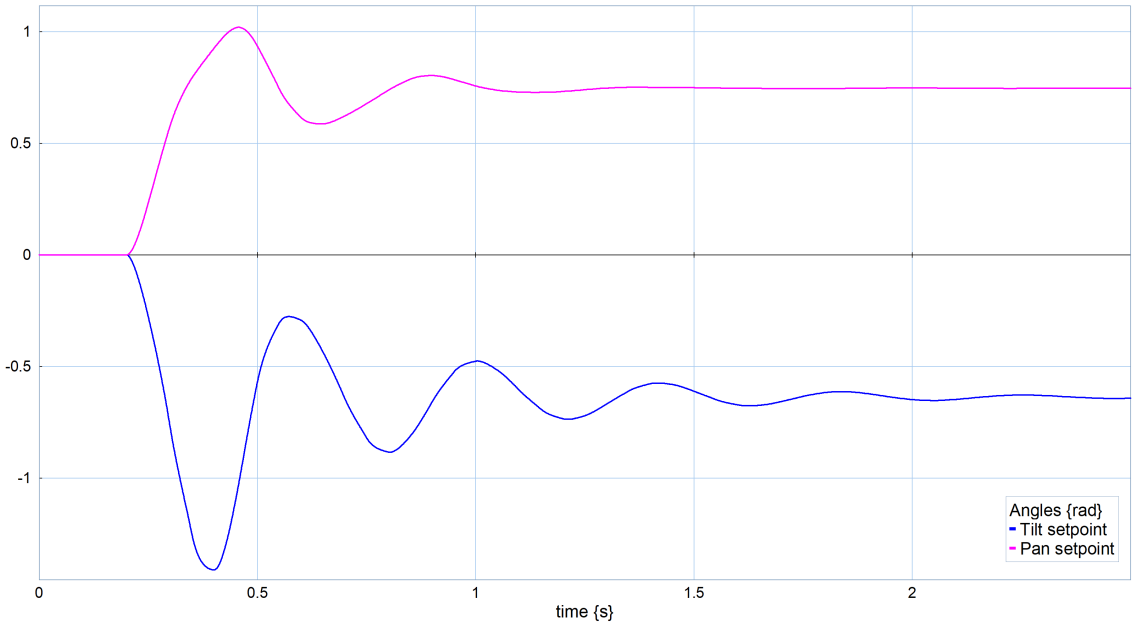


Figure 4.10: Simulated camera angles in response to a single feature, using the continuous damping value.

To compensate for the decreased damping effect of the discretization process, the damping factors are increased. The spring constant is left unchanged to not effect the speed of the step response. The increased damping factors are empirically determined to be:

$$c_{\text{comp}} = \begin{bmatrix} 3 \\ 2.3 \end{bmatrix} \tag{4.13}$$

The effect of this compensation factor is visible in the responses in figures 4.11, 4.12 and 4.13. These figures show that the step response appears critically damped again.

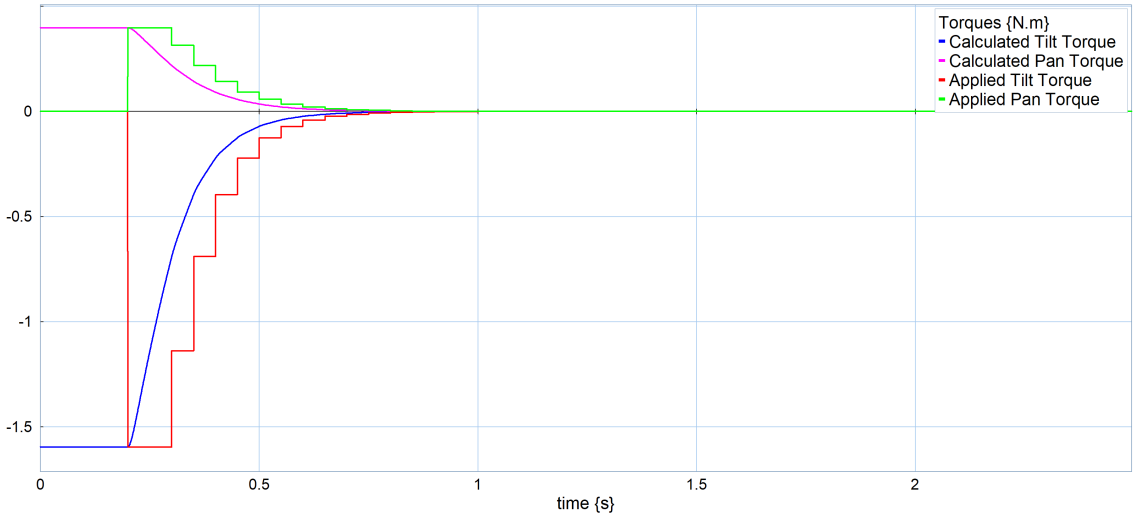


Figure 4.11: Simulated torque response to a single feature, using the compensated damping value.

The discretization process in combination with the damping compensation factor is used in all subsequent simulations.

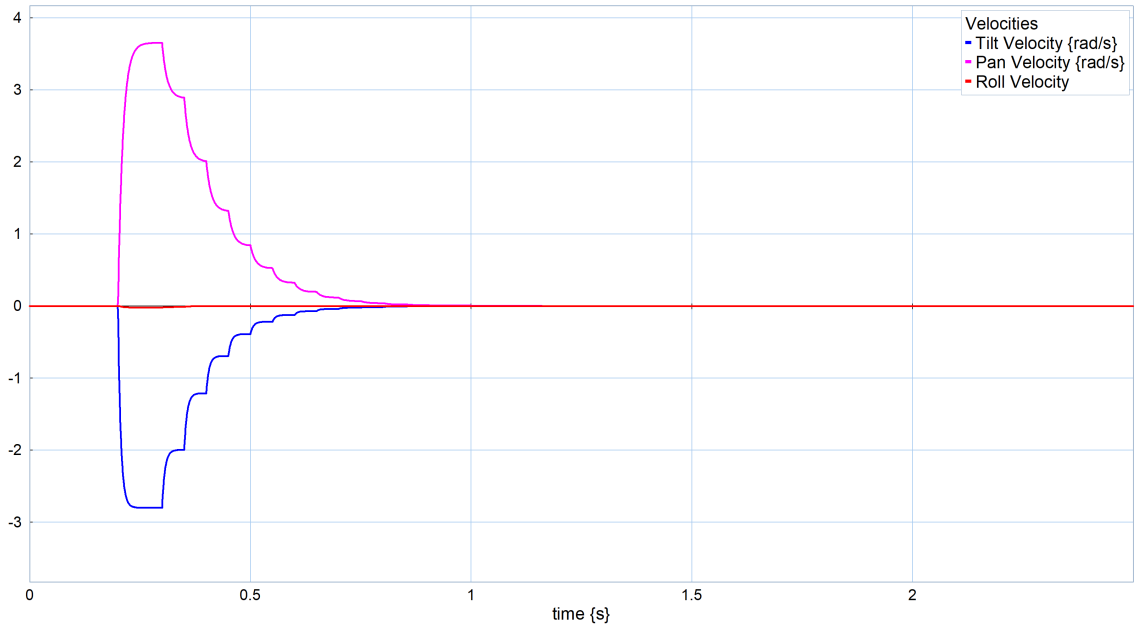


Figure 4.12: Simulated velocity response to a single feature, using the compensated damping value.

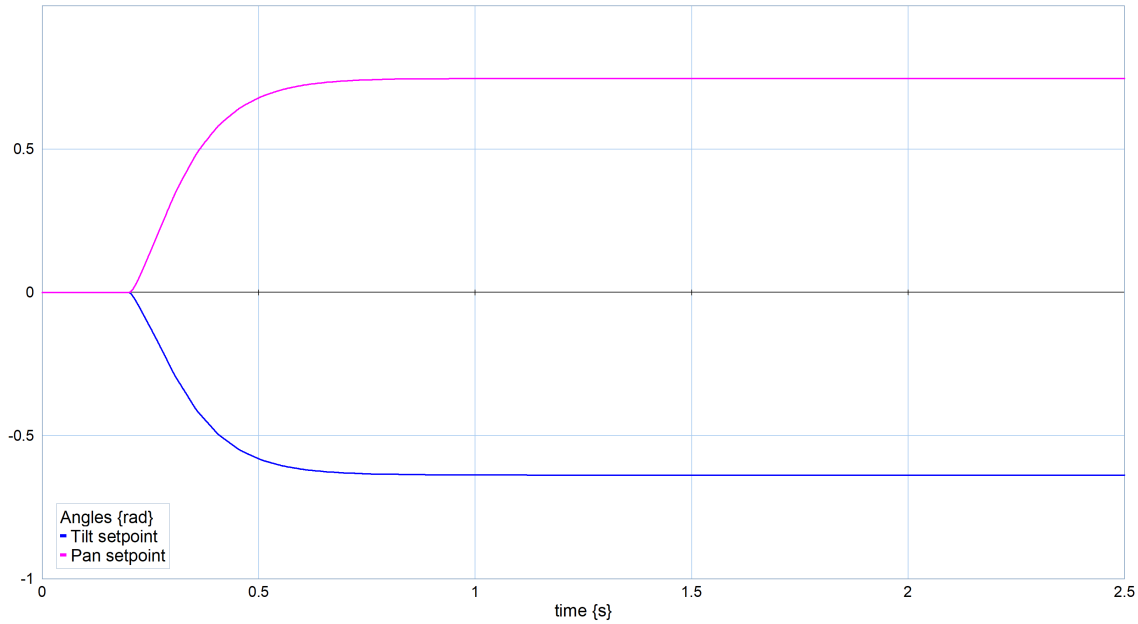


Figure 4.13: Simulated camera angles in response to a single feature, using the compensated damping value.

4.2.2 Asymmetric feature distribution

To test the response between asymmetric distributed features, a situation is created with a single feature at $p_f^w = \begin{pmatrix} 0 & 0 & 3 \end{pmatrix}^\top$ and a cluster of 10 features within a sphere of radius $r = 0.5$ around the point $p^w = \begin{pmatrix} 3 & 0 & 3 \end{pmatrix}^\top$.

This is visible from figure 4.16, as the pan final angle is approximately equal to the pan angle of the center of the cluster. The actual value is a little bit lower as the single feature still acts on the camera. It is actually $\frac{10}{11}$ of the pan angle of the cluster, which is as expected.

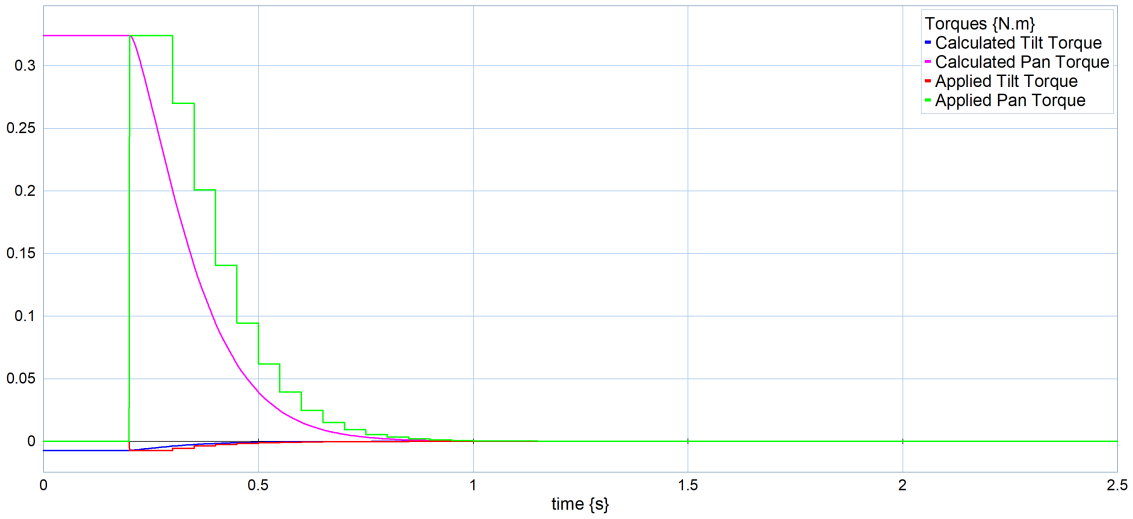


Figure 4.14: Simulated torque response to an asymmetric distribution of features.

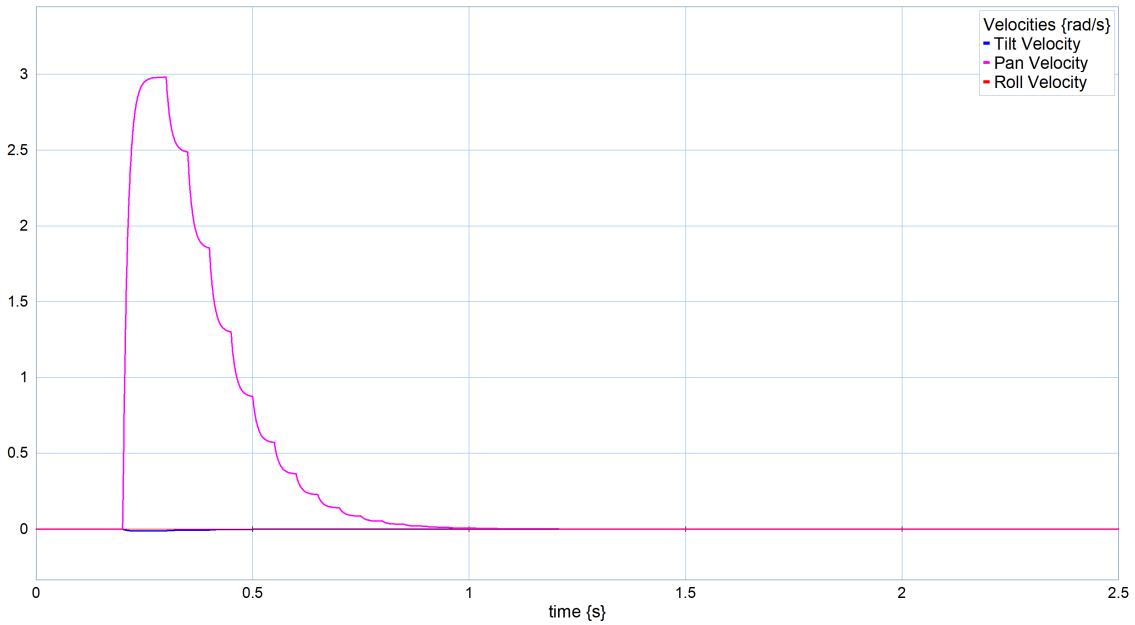


Figure 4.15: Simulated velocity response to an asymmetric distribution of features.

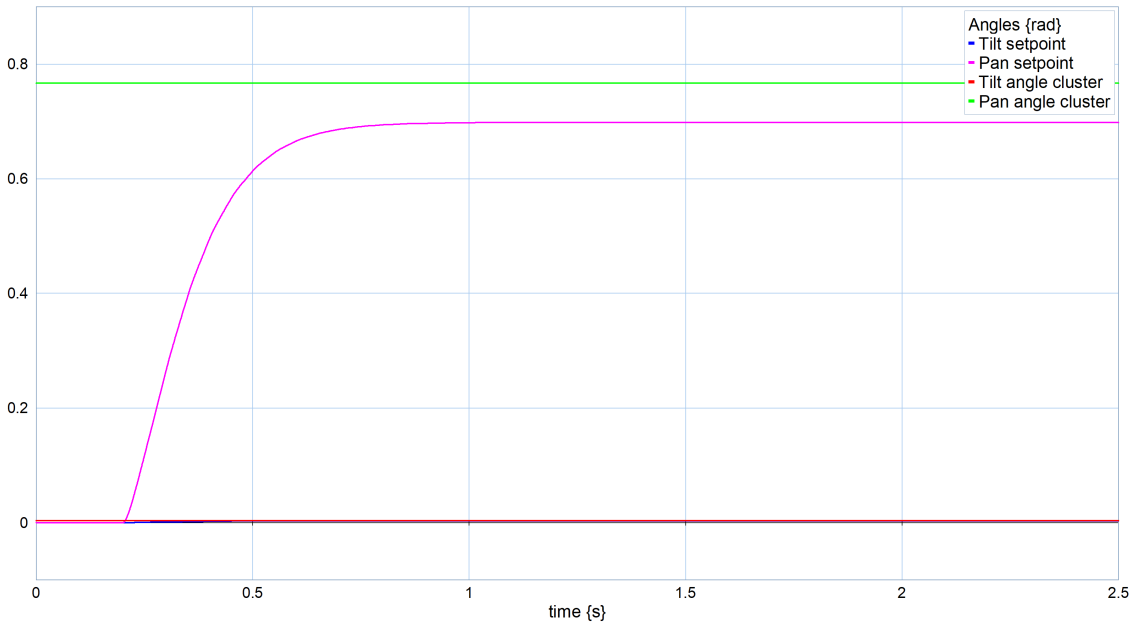


Figure 4.16: Simulated camera angles in response to an asymmetric distribution of features.

4.3 Virtual Field-of-View

In the real world, the VI-sensor has a limited field-of-view (FoV) that limits the amount of features of which information is available. To study the effect of this, a virtual field-of-view is implemented in the simulation. This implementation consists of applying a zero weight to the features that are outside of a certain angle with respect to the camera frame. This angle is selected to be equal to the physical field-of-view of the VI-sensor, namely 98° for the horizontal field-of-view and 73° for the vertical field-of-view.

In order to test this, the features of section 4.2.2 are copied and adapted by translating three features by $\Delta\vec{x} = \begin{pmatrix} 2 & 0 & 0 \end{pmatrix}^\top$, to increase their effect on the implemented FoV. The response to this situation is visible in figures 4.17, 4.18 and 4.19. These figures show that now that there are two clusters, the torque jumps up two times, but due to the inertia of the camera, this jump in torque does not introduce a discontinuity in the angular response. Therefore, the field-of-view does not disrupt the behaviour of the camera in an undesirable manner.

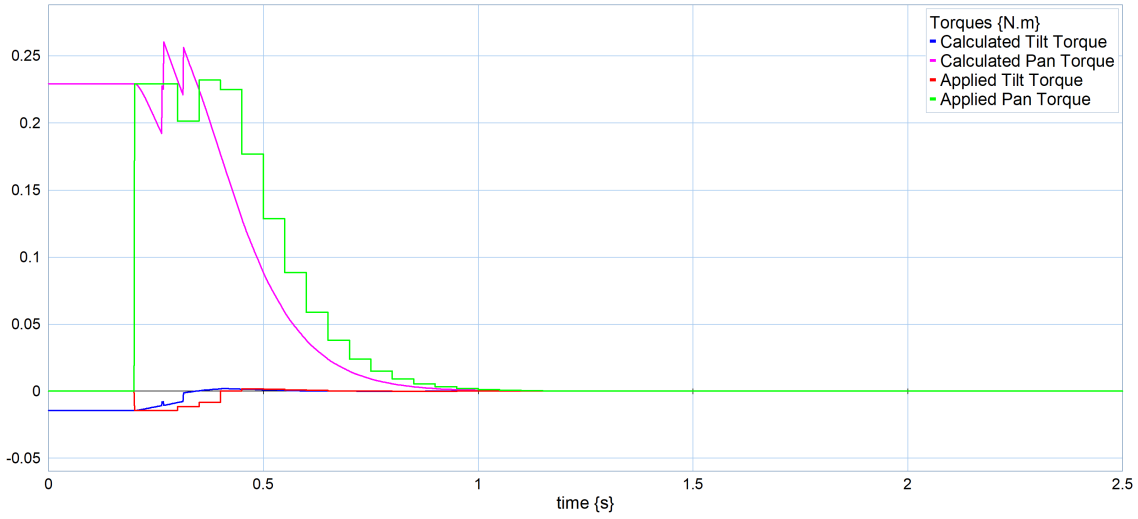


Figure 4.17: Simulated torque response after implementing a limited field-of-view.

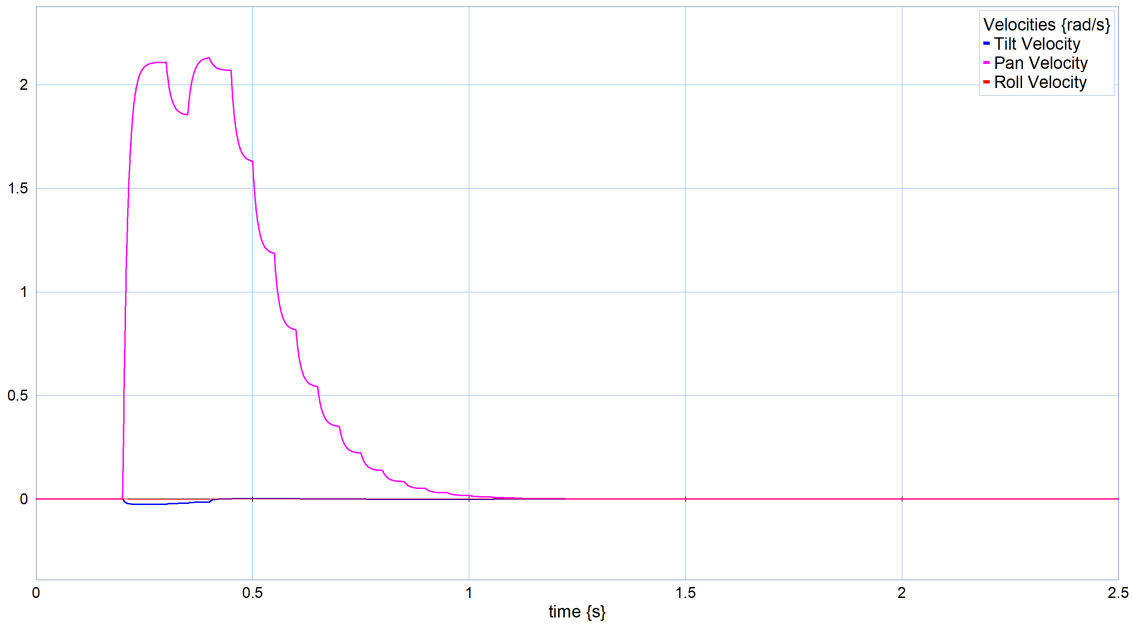


Figure 4.18: Simulated velocity response after implementing a limited field-of-view

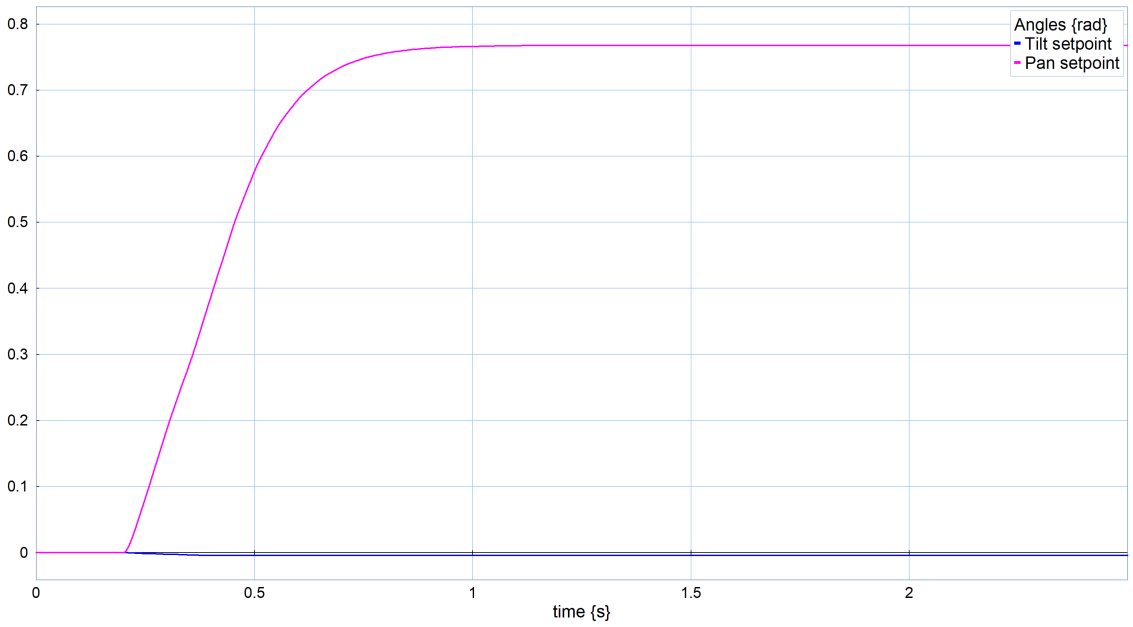


Figure 4.19: Simulated camera angles after implementing a limited field-of-view

4.3.1 Torus cloud

To test the response of the gaze controller to an (approximately) symmetric situation, a cloud of features is randomly generated within a torus shape with a central radius r_1 and a secondary maximum radius $r_{2,\max}$. The torus is generated around the origin with the central axis of symmetry along the z-axis and translated with a vector of $\Delta\vec{x}$. See figure 4.23. With the unrestricted model, the camera is expected to move to the center of the torus. Eccentricities of the generated torus shape will skew this result.

The radii and offset are chosen such that the features are just outside of the field-of-view in neutral configuration: for an offset of $\Delta\vec{x} = \begin{pmatrix}0 & 0 & 3.5\end{pmatrix}^\top$, the following relation should hold: $r_1 - r_{2,\max} > \sqrt{4.02^2 + 2.58^2}$. For a secondary radius of $r_{2,\max} = 0.4\,\text{m}$, the primary radius becomes $r_1 = 6.35\,\text{m}$.

The responses are shown in figures 4.20, 4.21 and 4.22. The response shows that the hypothesis is confirmed: the center of the torus is the z-axis, and thus the resulting angles should be minimal.

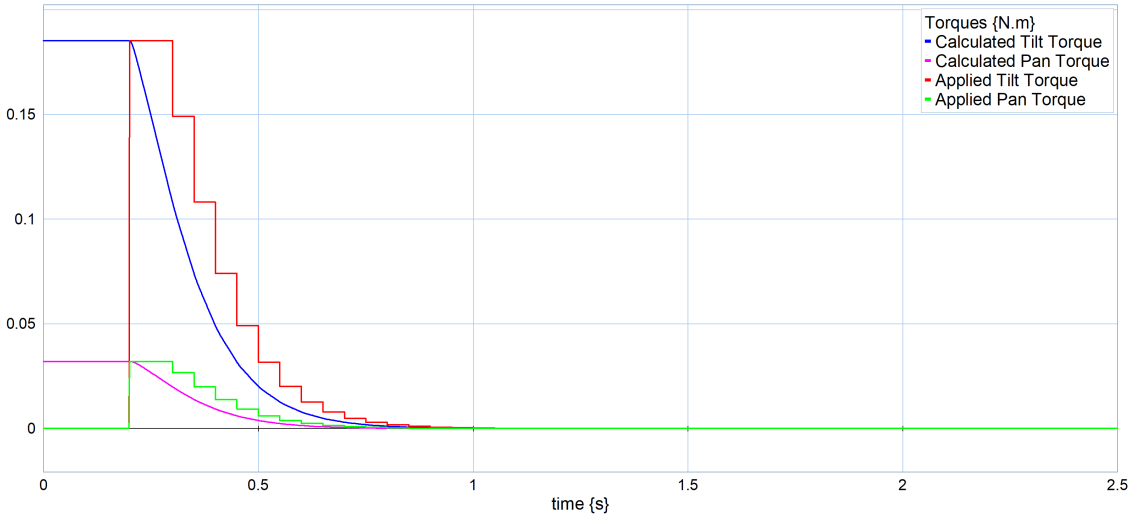


Figure 4.20: Simulated torque response to a torus-shaped cloud of features.

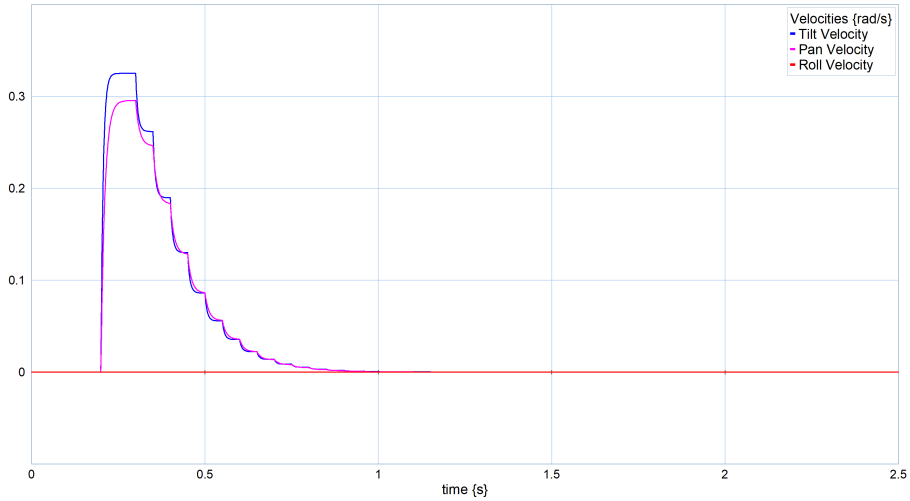


Figure 4.21: Simulated velocity response to a torus-shaped cloud of features.

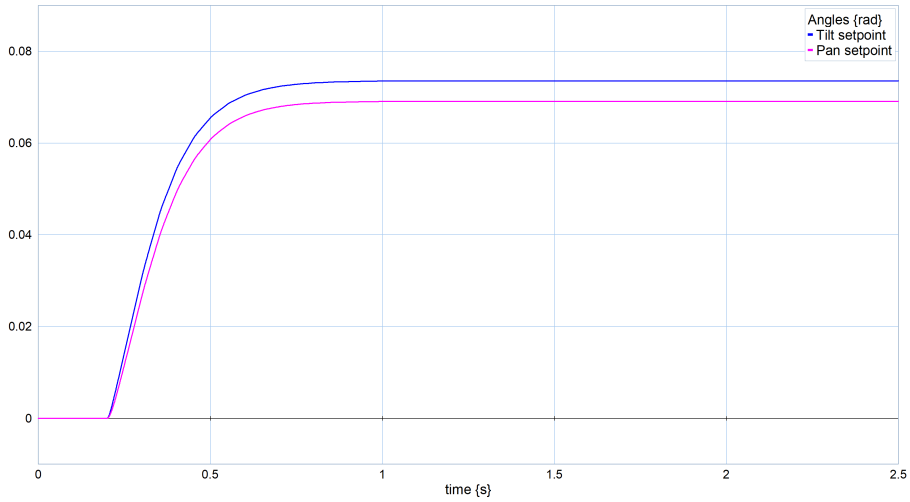


Figure 4.22: Simulated camera angles in response to a torus-shaped cloud of features.

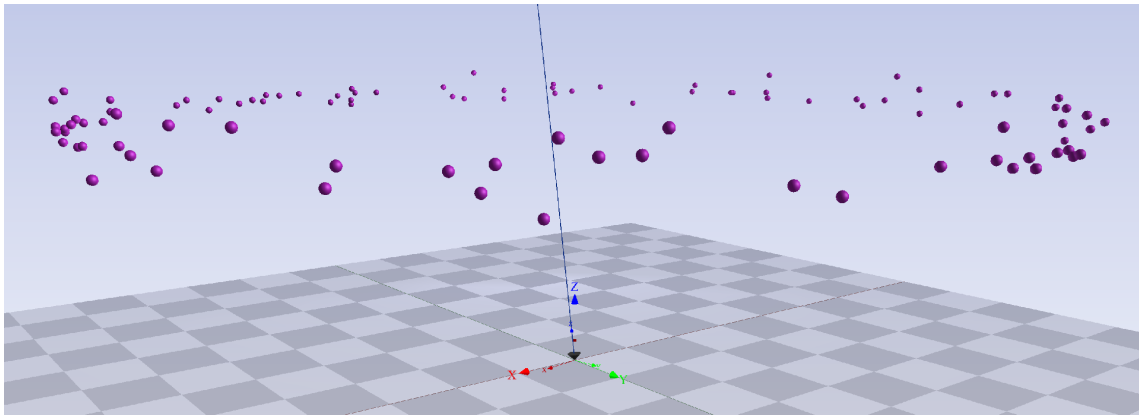


Figure 4.23: 3D simulation snapshot at $t = 2.5\,\mathrm{s}$ of the response to a torus-shaped cloud of features.

Next, the field-of-view is applied to the torus situation. Then the expected response becomes zero. This is indeed the case as showed in figure 4.24.

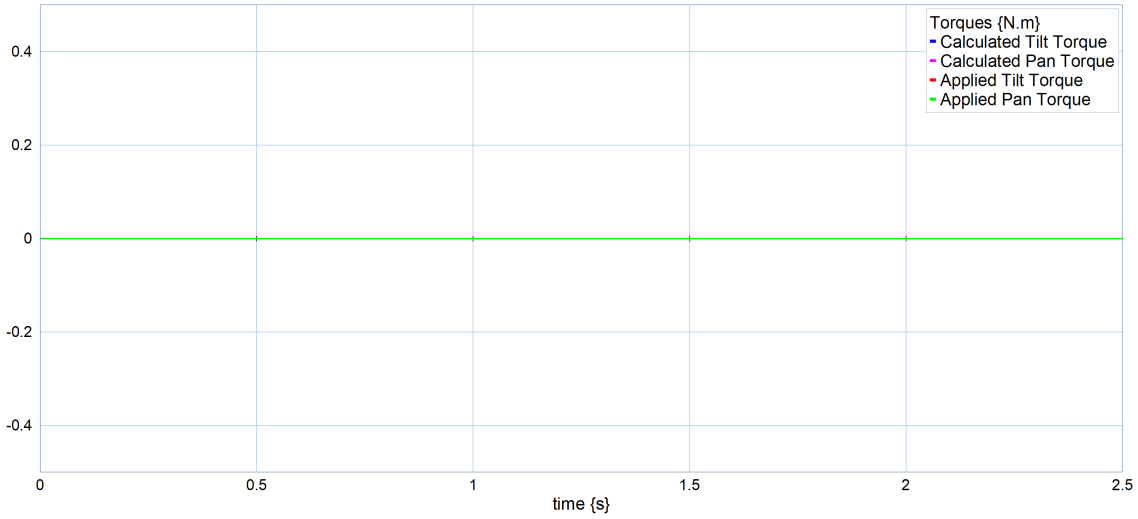


Figure 4.24: Simulated torque response to a torus-shaped cloud of features, while applying the limited field-of-view.

Next, a small offset in the camera angle, 0.2 rad about x and y is introduced, so that a number of features is just visible in the corner of the FoV. The expectation is that the features in this corner of the FoV will draw the camera towards that side of the torus and that the camera angle settles in that direction. The response is visible in figures 4.25, 4.26 and 4.27. These figures show that the camera angle does not directly settle on the initial direction, but changes direction during its motion.

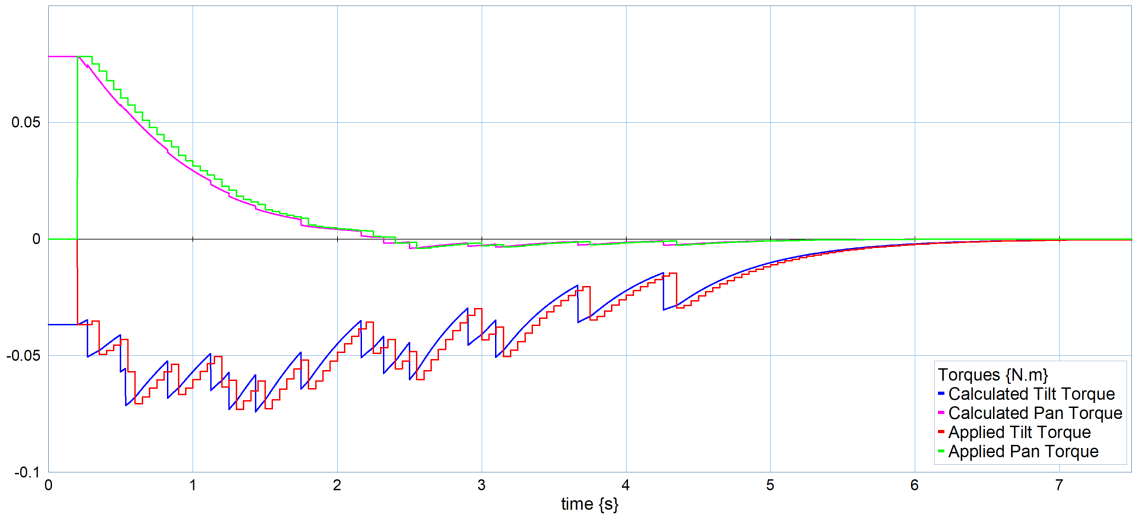


Figure 4.25: Simulated torque response to a torus-shaped cloud of features. The camera angle includes an initial offset.

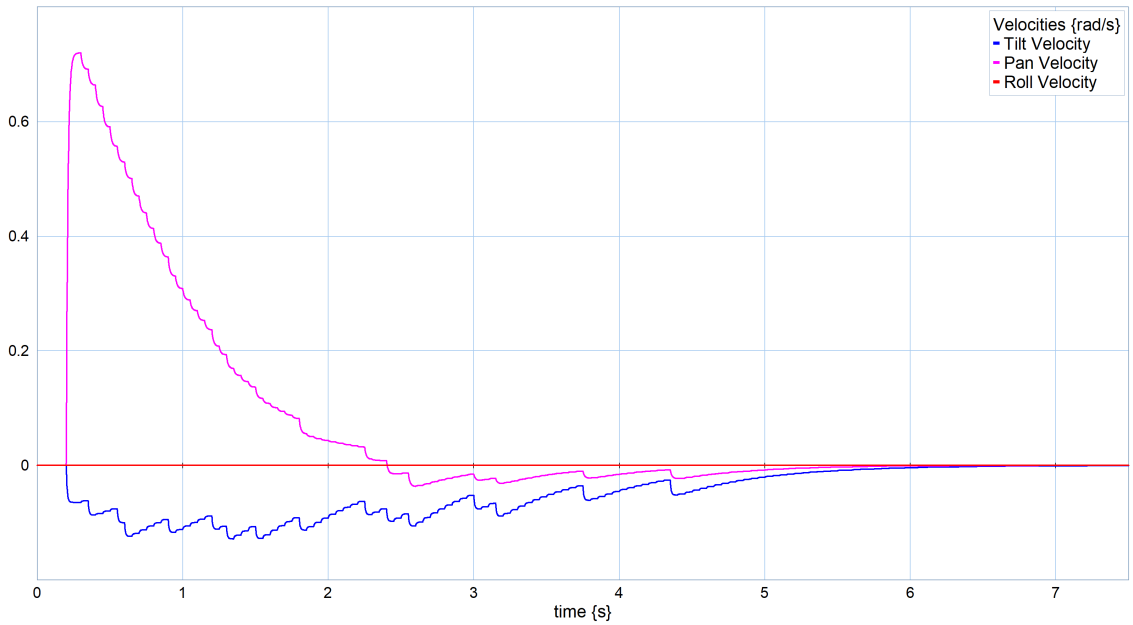


Figure 4.26: Simulated velocity response to a torus-shaped cloud of features. The camera angle includes an initial offset.

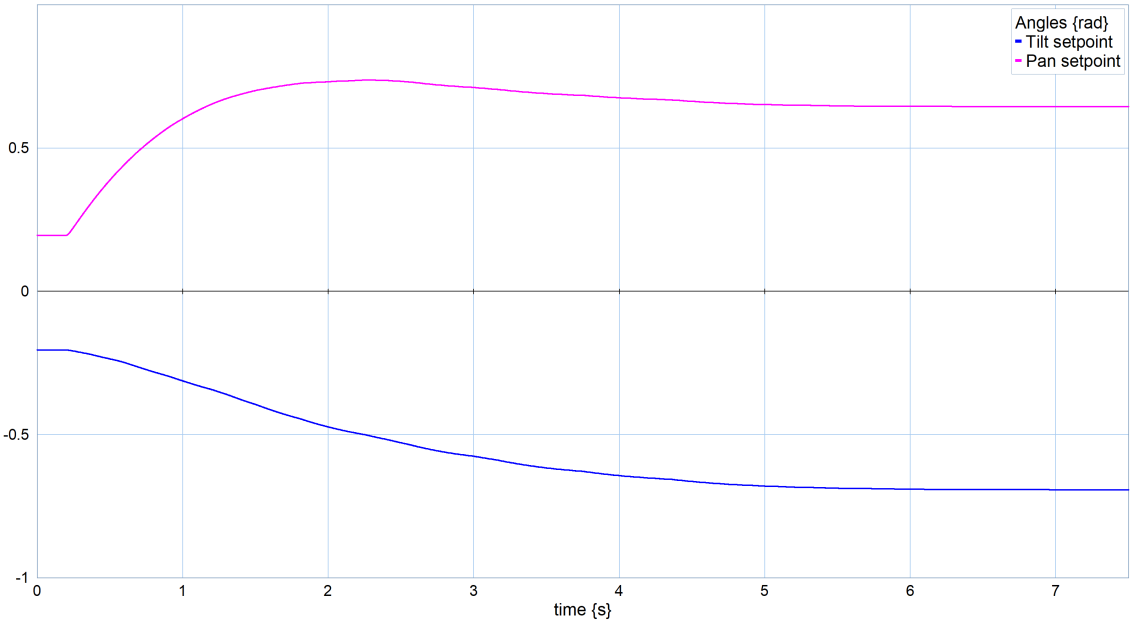


Figure 4.27: Simulated camera angles in response to a torus-shaped cloud of features. The camera angle includes an initial offset.

4.4 Spring Torque Penalties

In this section the penalties that are applied to the spring torque are subjected to testing in a simulated scenario. Through these tests, the efficacy of these penalties are investigated.

4.4.1 Distance penalty

To test the distance penalty, a specific situation is created: a torus-shaped cloud that is just outside the field-of-view at a close distance (i.e. in the unity region of α), and a spherical-shaped cloud at a far distance (i.e. in small constant region of α). Note that the field-of-view is included in this simulation. To ensure that the simulation is demonstrative, these distance limits are scaled down, so that all features fit in the simulation window. The spherical cloud has a radius of $r = 2\,\text{m}$ and is generated around $\Delta\vec{x} = \begin{pmatrix}0 & 0 & 8\end{pmatrix}^\top$. The torus cloud has a primary radius of $r_1 = 3.0356\,\text{m}$ and a secondary maximum radius of $r_{2,\max} = 0.3\,\text{m}$ and is generated around $\Delta\vec{x} = \begin{pmatrix}0 & 0 & 2\end{pmatrix}^\top$.

The scaled α becomes:

$$\alpha(d\{m\}) = \begin{cases} d & 0 \leq d < 1 \\ 1 & 1 \leq d < 4 \\ 1 - 0.9 \cdot (d-4) & 4 \leq d \geq 5 \\ 0.1 & d > 5 \end{cases} \tag{4.14}$$

The response without the effect of α is shown in figures 4.28, 4.29 and 4.30. This shows that the camera stays focused on the central spherical cloud, even though that cloud has a greater distance than the torus cloud. This effect is expected to be undesired as it is assumed that features that are far away are less valuable for the visual odometry process than features that are closer.

The same situation is tested with the effect of α included. The response of that experiment is shown in figures 4.31, 4.32 and 4.33. These figures show that the effect of α is as expected: a preference is shown for the features that are within the preferred range (i.e. the torus cloud).

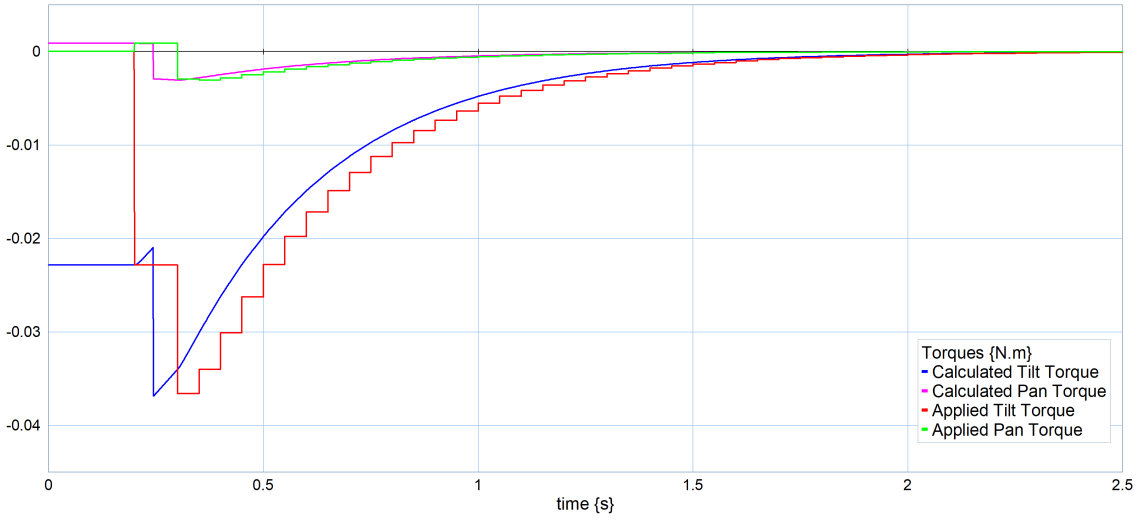


Figure 4.28: Simulated torque response to a torus-shaped cloud and a spherical shaped cloud of features at different distances. The effect of α is not included.

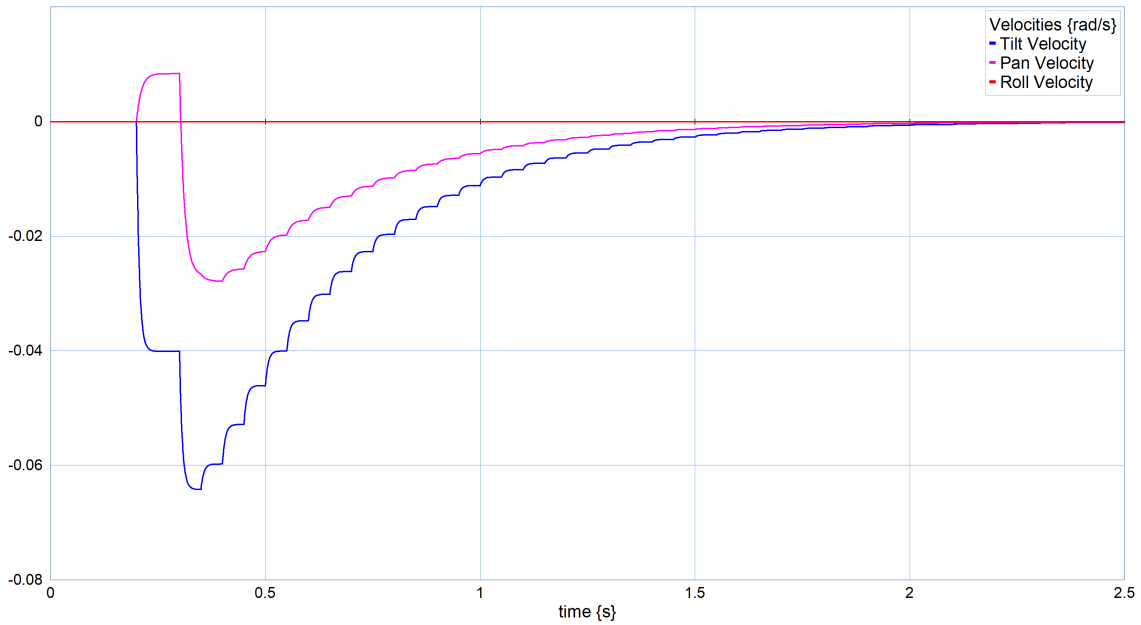


Figure 4.29: Simulated velocity response to a torus-shaped cloud and a spherical shaped cloud of features at different distances. The effect of α is not included.

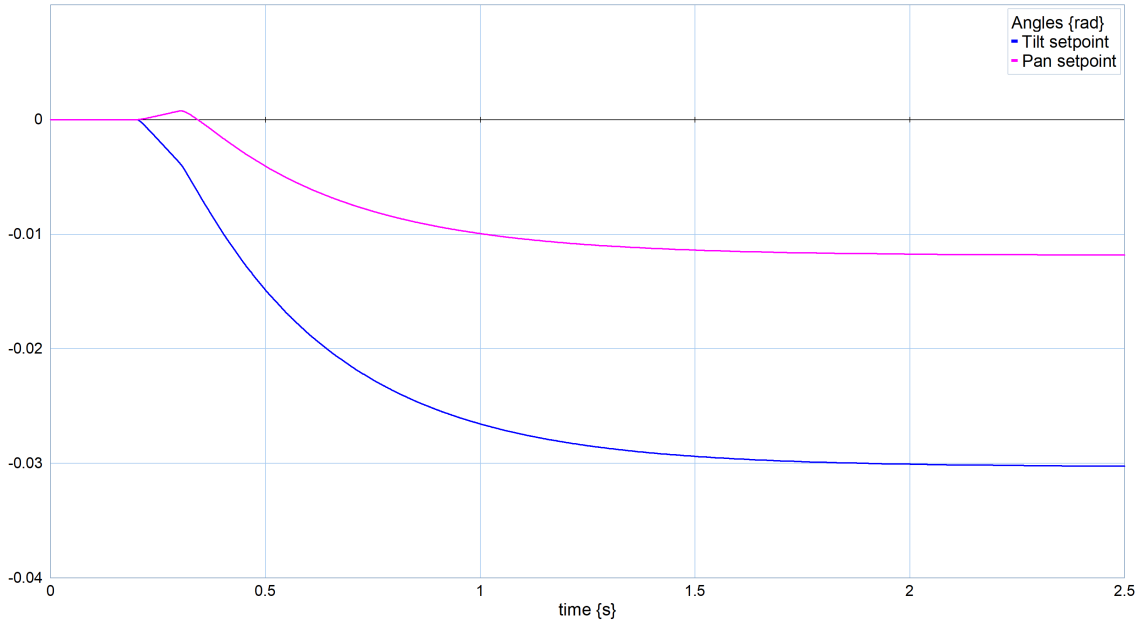


Figure 4.30: Simulated camera angles in response to a torus-shaped cloud and a spherical shaped cloud of features at different distances. The effect of α is not included.

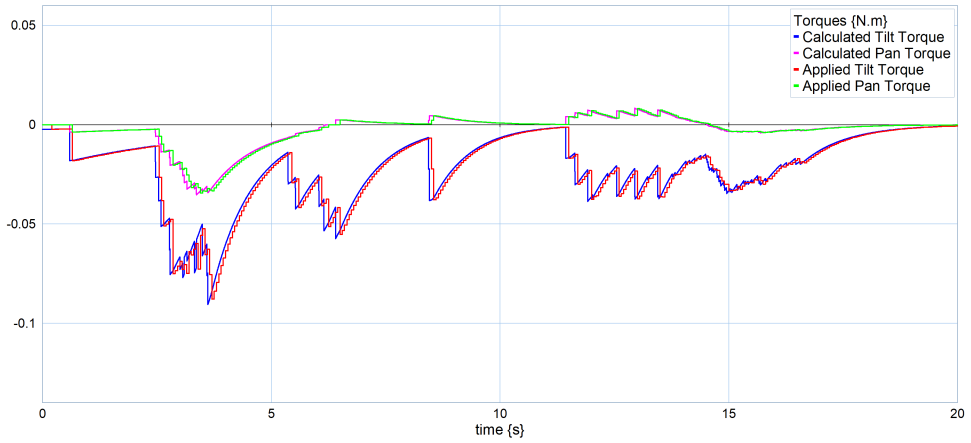


Figure 4.31: Simulated torque response to a torus-shaped cloud and a spherical shaped cloud of features at different distances. The effect of α is included.

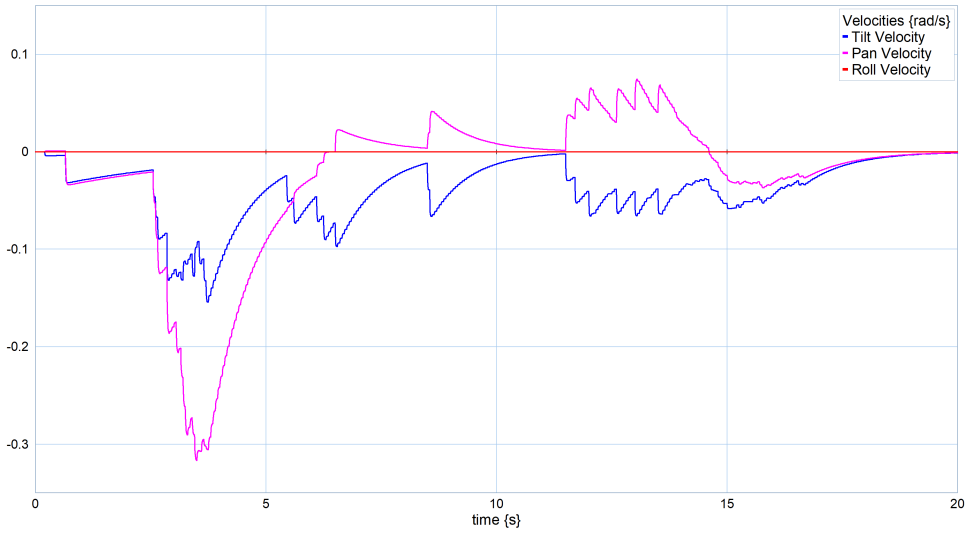


Figure 4.32: Simulated velocity response to a torus-shaped cloud and a spherical shaped cloud of features at different distances. The effect of α is included.

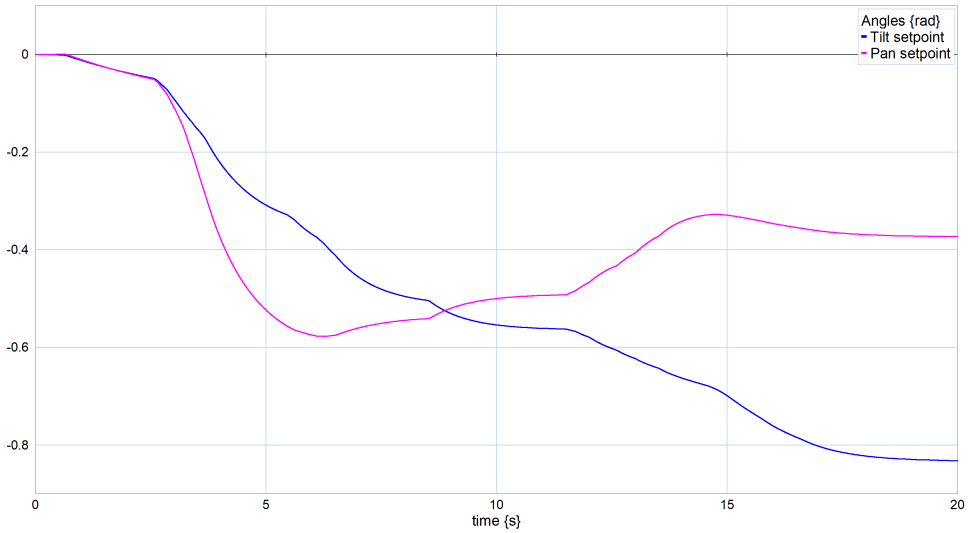


Figure 4.33: Simulated camera angles in response to a torus-shaped cloud and a spherical shaped cloud of features at different distances. The effect of α is included.

4.4.2 Physical limit penalty

The effect of β is tested by creating three features with coordinates:

$$P_{f_1}^w = \begin{pmatrix} 4 \\ 0 \\ 3.5 \end{pmatrix} \qquad P_{f_2}^w = \begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix} \qquad P_{f_3}^w = \begin{pmatrix} 4 \\ 0 \\ -2 \end{pmatrix} \tag{4.15}$$

The hypothesis for the created situation is that without β the final angle is aimed towards the average angle of all three features, which is $\theta_{\text{avg}} \approx 1.49\,\text{rad}$. However, with the effect of β the lower two features would not count any more, which means that the final angle will be focused on the top feature. Figures 4.34 and 4.35 show the response without β. Figures 4.36 and 4.37 show the response including β. Figures 4.35 and 4.37 confirm the hypothesis. Note that the response including β is slower than the response without β. This is an undesired side effect of the spring constant normalization discussed in section 4.2.

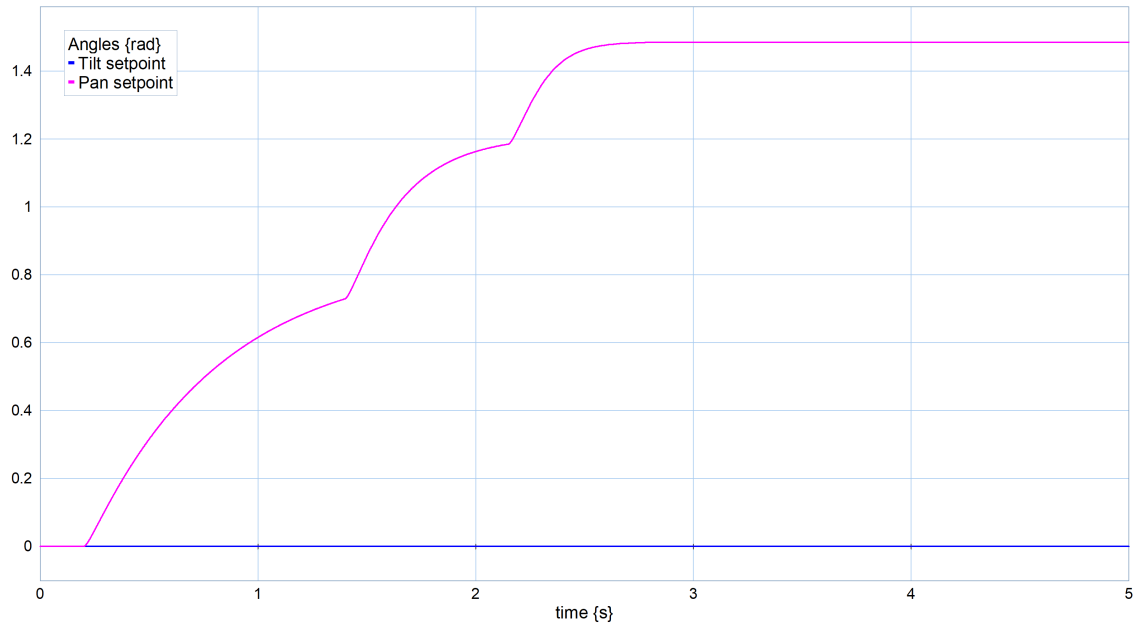


Figure 4.34: Simulated camera angles in response to three aligned features, without the effect of β.

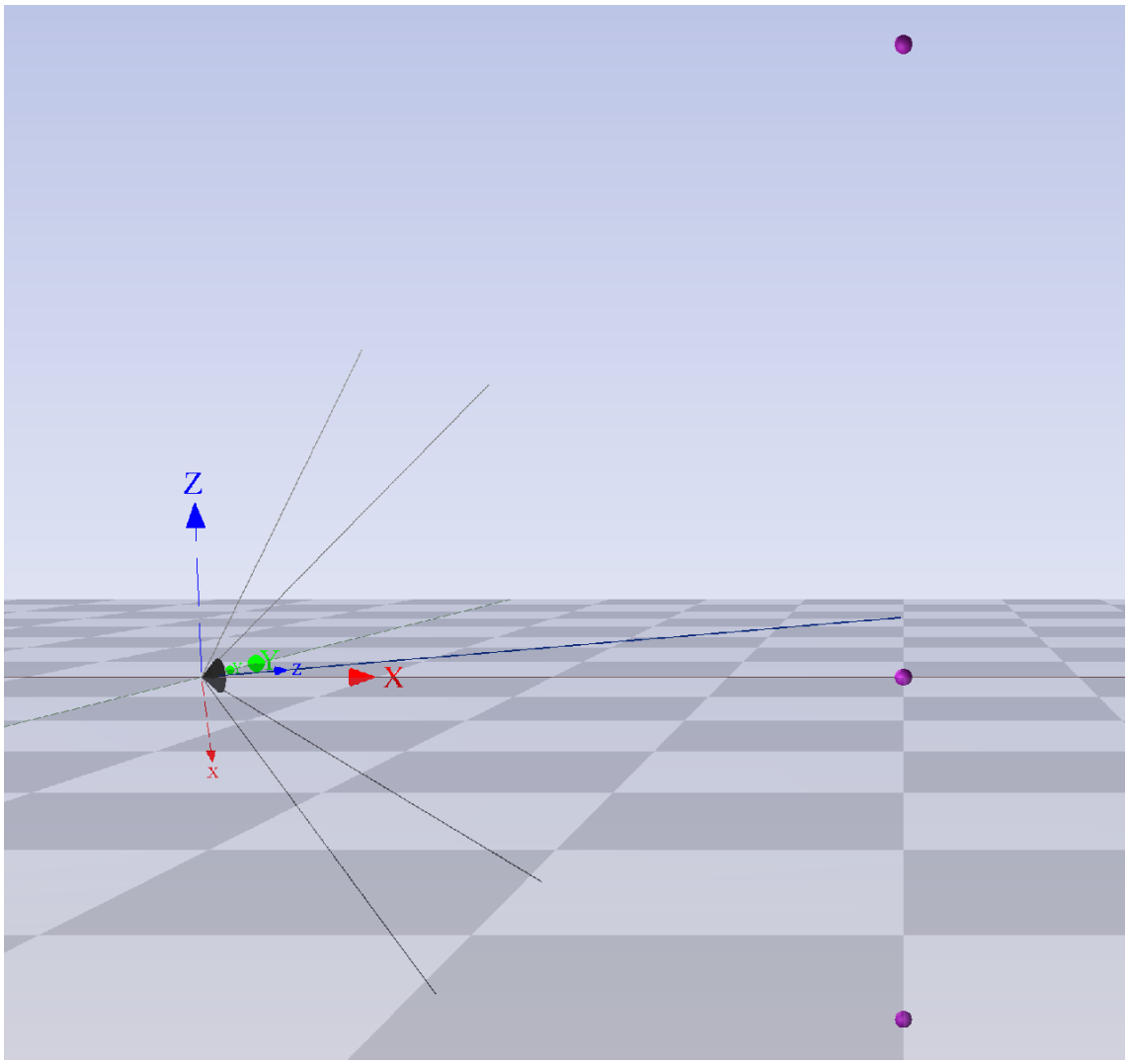


Figure 4.35: Final angle in response to three aligned features, without the effect of β.

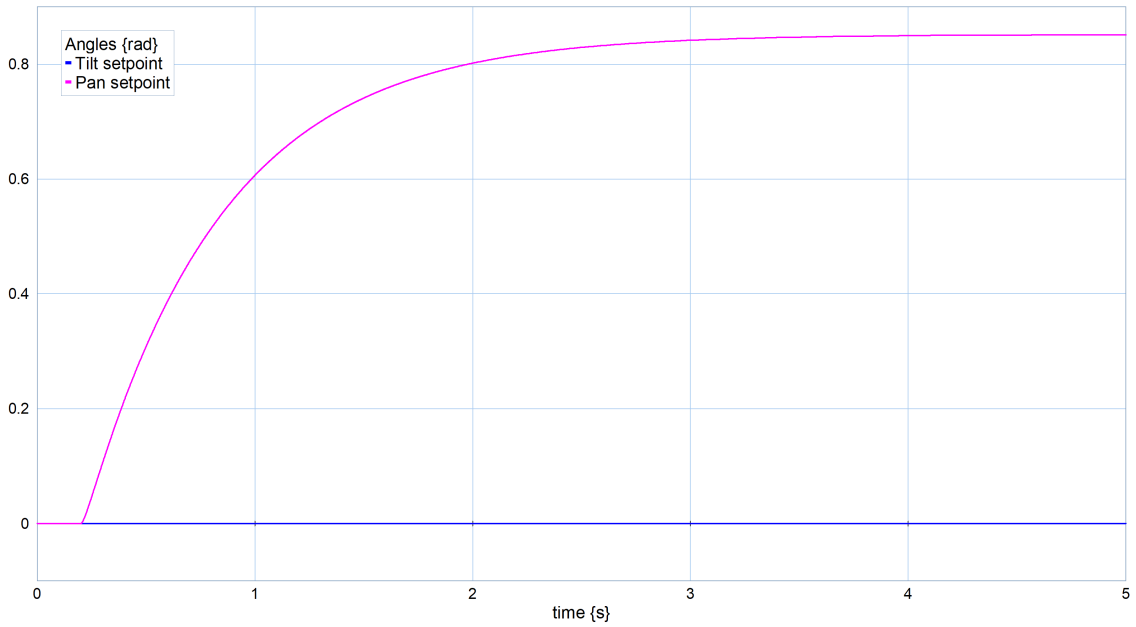


Figure 4.36: Simulated camera angles in response to three aligned features, including the effect of β.

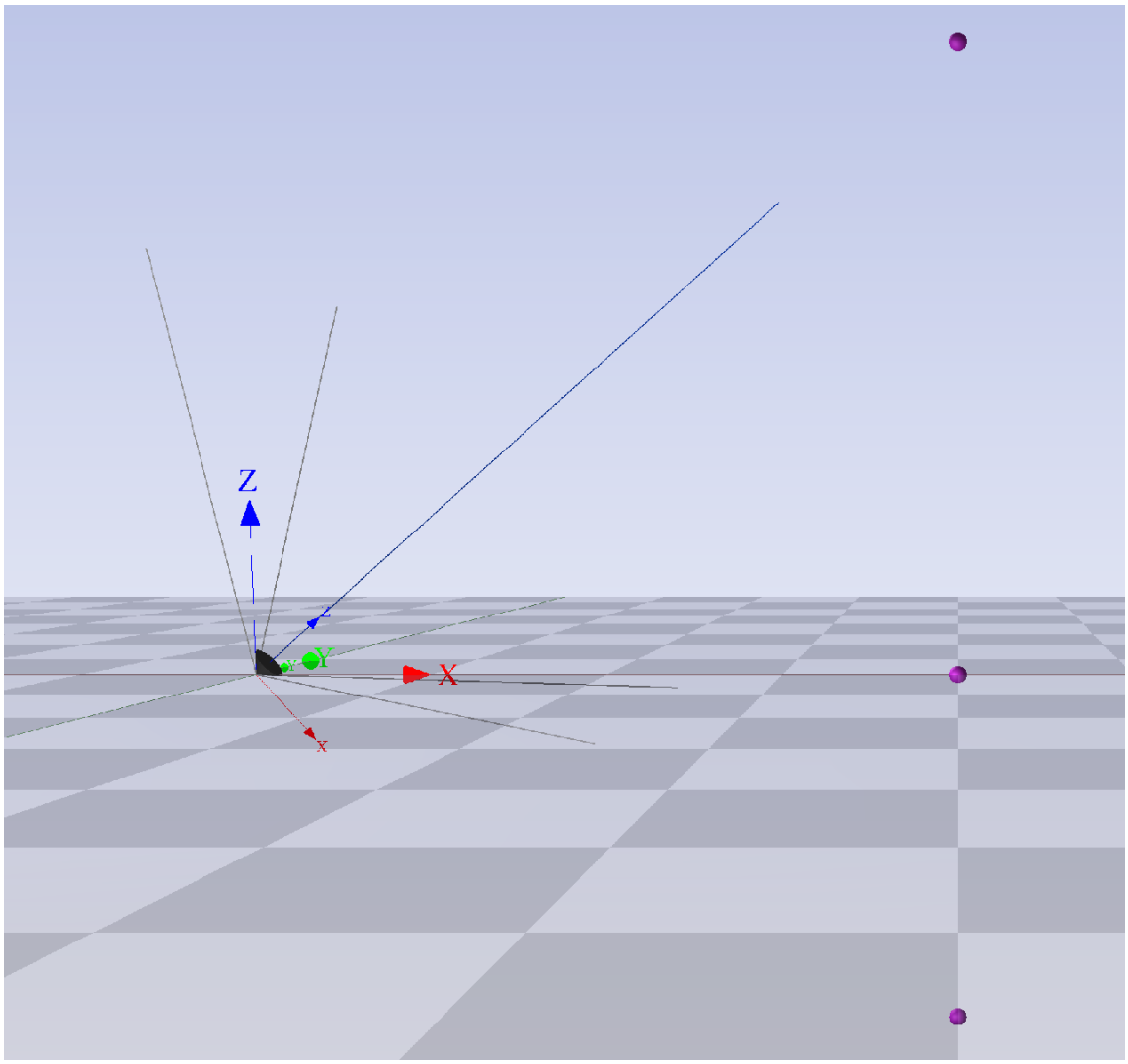


Figure 4.37: Final angle in response to three aligned features, including the effect of β.

4.4.3 Feature initialization penalty

To test the effect of the feature initialization penalty γ, a scenario is created with three equidistant (in angle) features, with an elevation angle ranging from 0 to $\frac{1}{2}\pi$ rad from the x-axis, plus one with an elevation angle of 0.5 rad. The goal of this experiment is to demonstrate that a feature like γ is desirable to counteract sudden increases in acceleration (jerky behaviour), because that is exactly what you would expect when a new feature enters the field-of-view.

Figures 4.38, 4.39 and 4.40 show that this happens. Next, the effect of γ is applied with a maturation time $t_m = 1$ s. The resulting response is shown in figures 4.41, 4.42 and 4.43. These figures show that the response is smoothed. The downside of this effect is that the response takes longer to converge. Another side-effect of γ is the creation of a sawtooth-wave like form for the torque response. However, this effect does not form a problem, as the discretization process and the inertia effect limit the consequences of this disturbance.

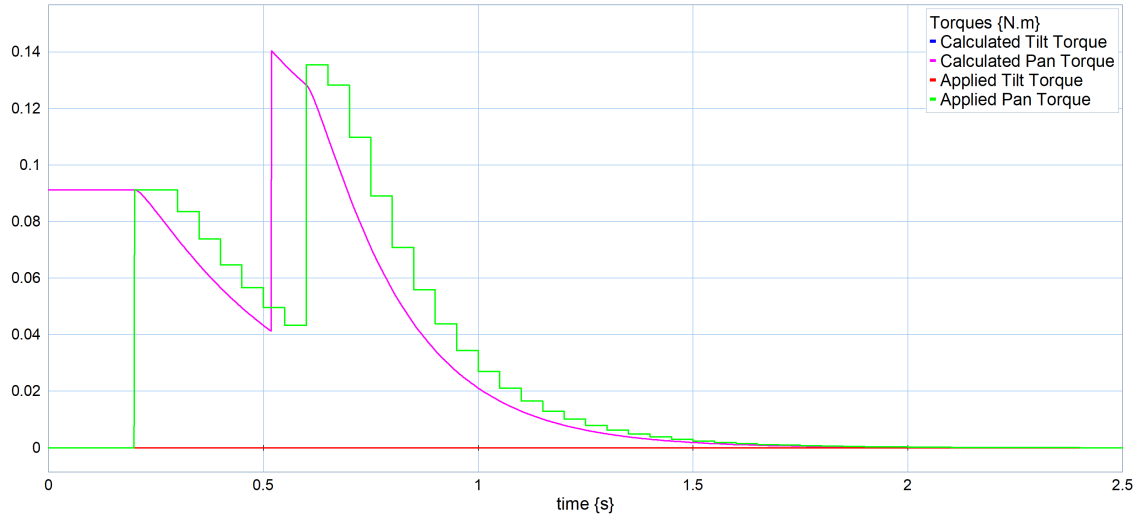


Figure 4.38: Simulated torque response to features with different elevation angles. The effect of γ is not included.

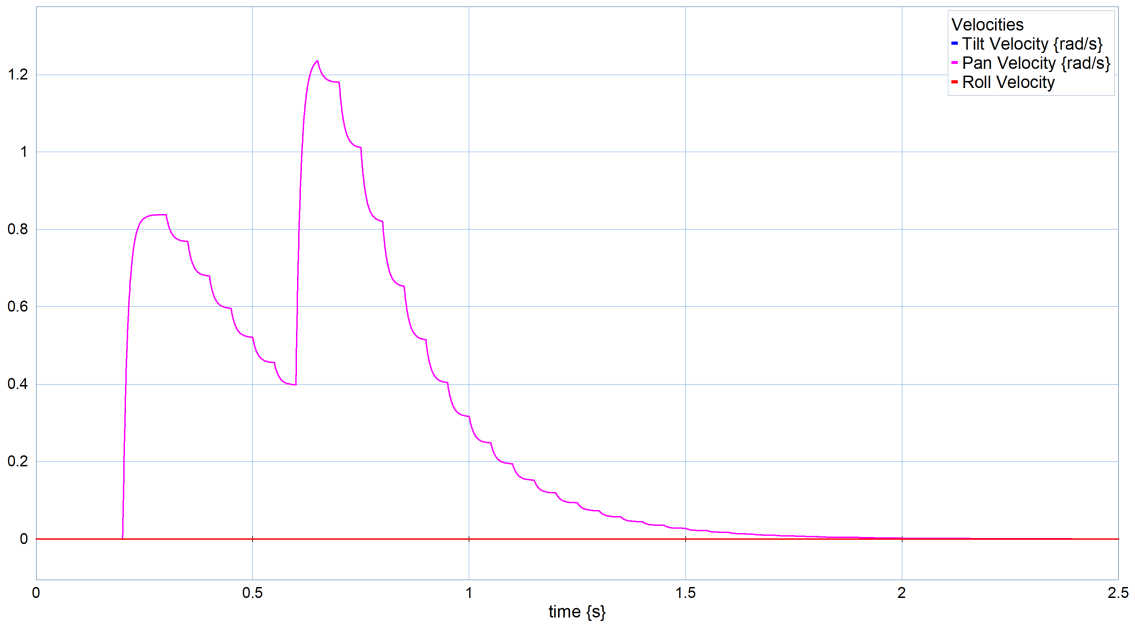


Figure 4.39: Simulated velocity response to features with different elevation angles. The effect of γ is not included.

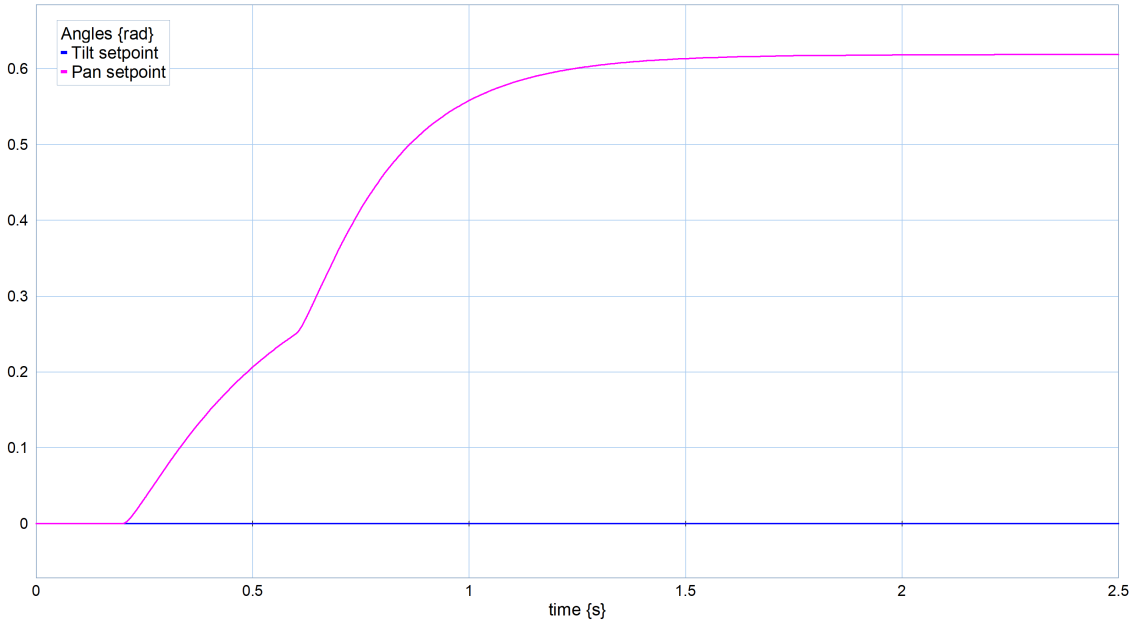


Figure 4.40: Simulated camera angles to features with different elevation angles. The effect of γ is not included.

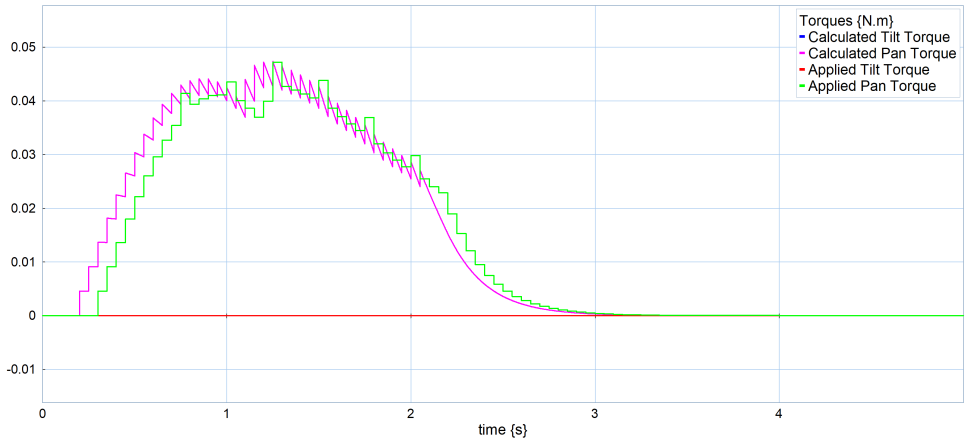


Figure 4.41: Simulated torque response to features with different elevation angles. The effect of γ is included. Note the larger time scale.

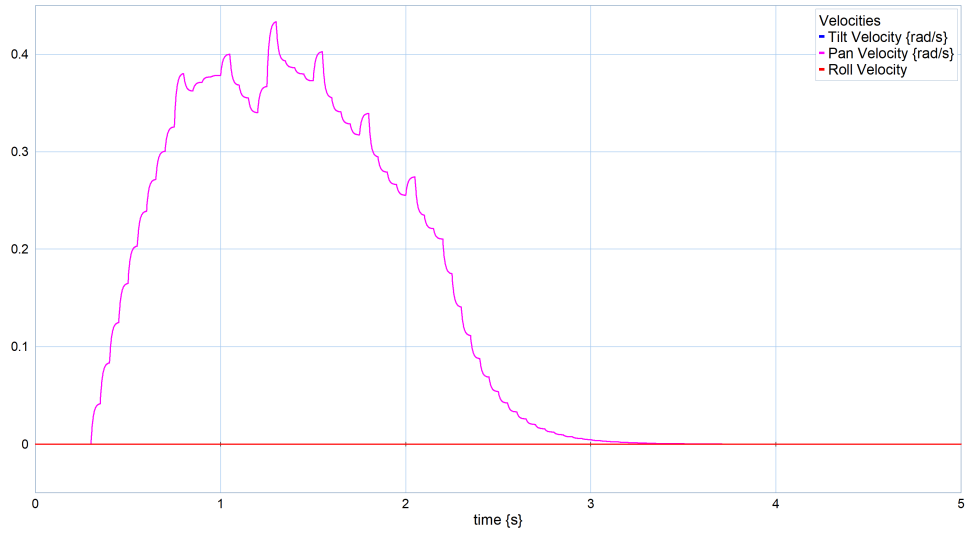


Figure 4.42: Simulated velocity response to features with different elevation angles. The effect of γ is included. Note the larger time scale.

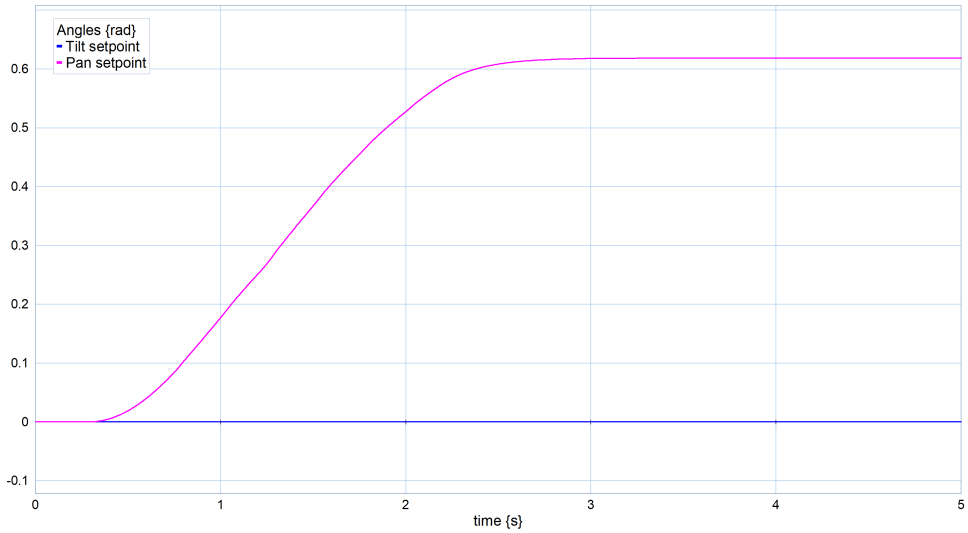


Figure 4.43: Simulated camera angles to features with different elevation angles. The effect of γ is included. Note the larger time scale.

4.5 Motion experiments

The intended goal of visual odometry is (relative) localization while the camera is in motion. Therefore, three simulation experiments are investigated where the camera is in motion to test the response of the gaze control algorithm to motion of the base. In both experiments the motion of the base consists of a linear trajectory. More specifically, a constant base velocity v_b of $0.5\,\mathrm{m\,s^{-1}}$ is applied. All three penalties are applied as well.

4.5.1 Spherical cloud

The first situation is a spherical cloud of features with a radius of $r = 5\,\mathrm{m}$ centered around $\Delta\vec{x} = \begin{pmatrix}5 & 0 & 8\end{pmatrix}^\top$. The base velocity is in the positive x-direction. The initial position of the camera is $p_c^w(0) = \begin{pmatrix}-15 & 0 & 0\end{pmatrix}^\top$. The response is shown in figures 4.44, 4.45 and 4.46. The figures show that initially the cloud is completely out of view. As soon as it enters the view at 9.3 s, the torque increases as more features will move into the field-of-view as the camera rotates. As soon as the entire cloud is in view and the relative angle between the camera center and the feature cloud center decreases, the torque decreases as well. As the base trajectory progresses, the relative angle changes slowly, and the camera has no problem tracking the cloud (see figure 4.47). At $t = 40\,\mathrm{s}$ the camera is directly under the cloud, and it is visible that in this region, the angular velocity reaches a local extremum. Overall, the response is exactly as expected.

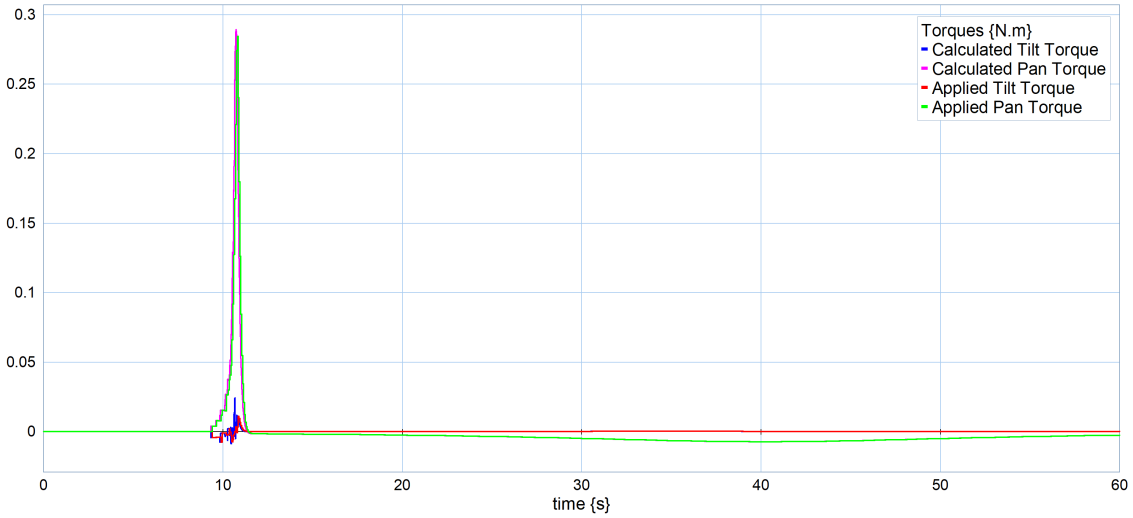


Figure 4.44: Simulated torques while moving past a spherical feature cloud.

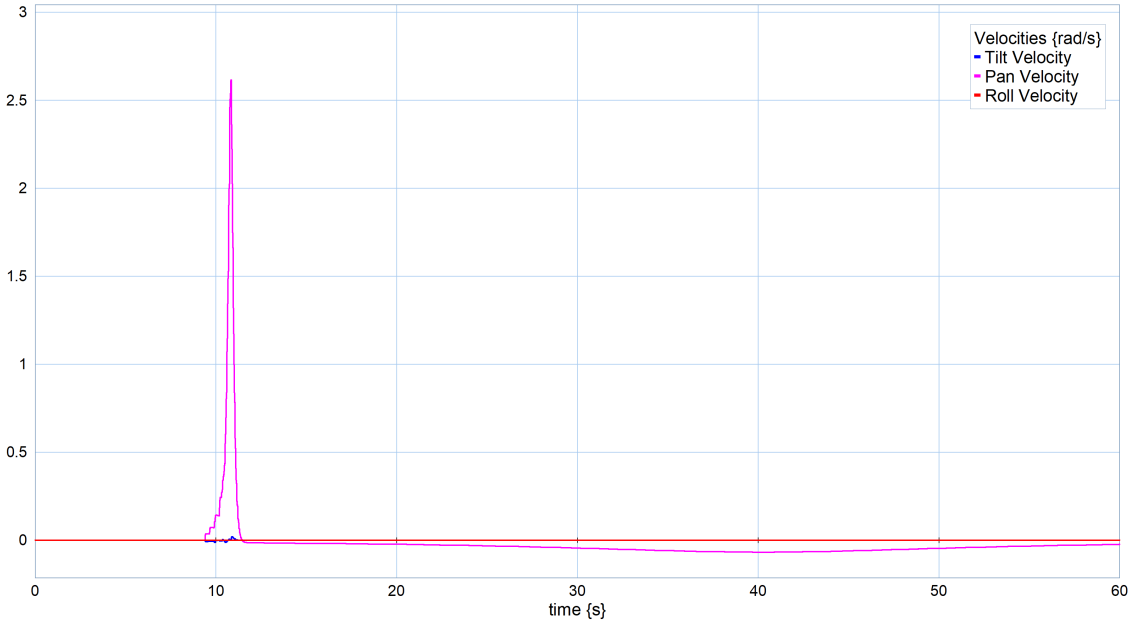


Figure 4.45: Simulated velocities while moving past a spherical feature cloud.

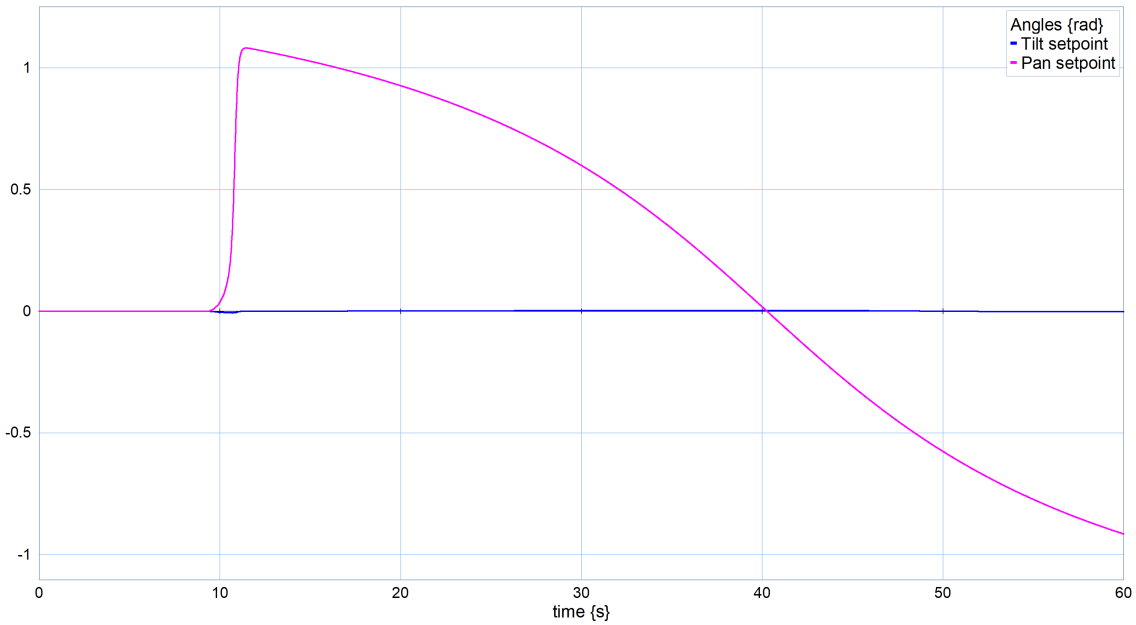


Figure 4.46: Simulated camera angles while moving past a spherical feature cloud.

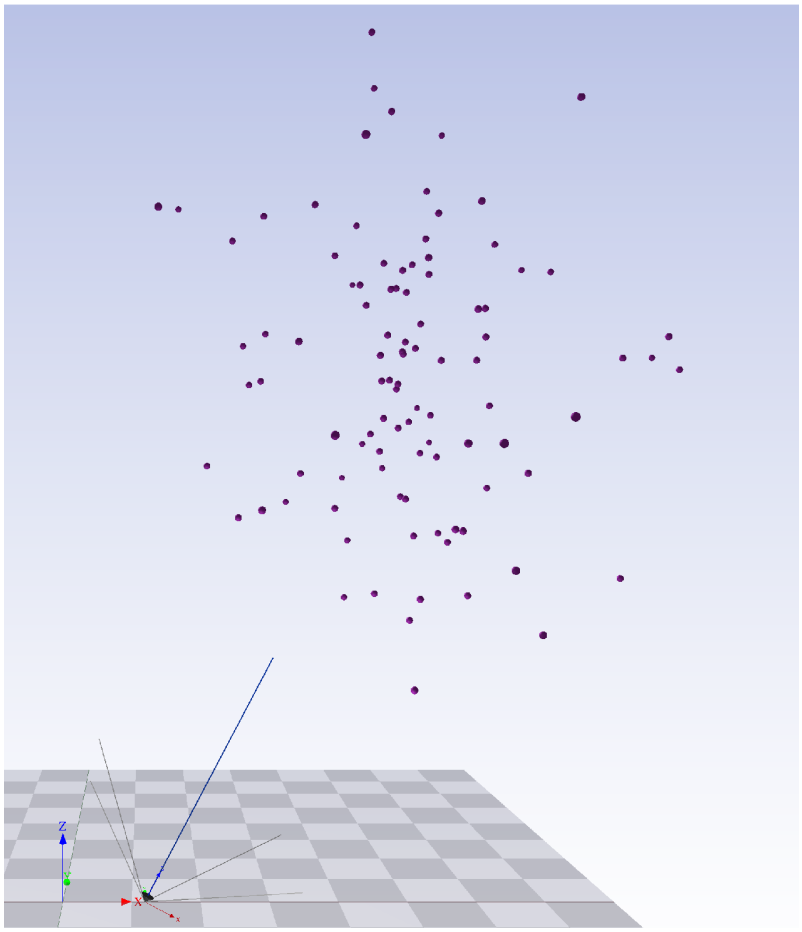


Figure 4.47: 3D visualization of the camera angle and the feature cloud at $t = 32.4\,\text{s}$. The camera moves past a spherical feature cloud.

4.5.2 Torus cloud

The second situation is a torus-shaped cloud. This cloud is generated about the z-axis with a primary radius of $r_1 = 2.5\,\text{m}$, a secondary radius of at maximum $r_{2,\max} = 0.5\,\text{m}$ at an offset of $\Delta\vec{x} = \begin{pmatrix}0 & 0 & 5\end{pmatrix}^\top$. The base velocity is directed in the positive z-direction, and thus moving towards the center of the torus. The hypothesis is that due to eccentricities of the torus, the camera aim will be pulled towards one side of the torus and will remain in that direction while moving upwards. However, due to the effect of β there will be a limit where the angle cannot rotate any further.

The response in this situation is shown in figures 4.48, 4.49 and 4.50. Figures 4.51 and 4.52 show 3D visualizations of the situation at $t = 7.4\,\text{s}$ and $t = 15.3\,\text{s}$ respectively. These figures show that indeed the camera aim is pulled towards one side of torus. Also, they show the limit introduced by β. However, the camera aim does not settle in the initial direction and keeps rotating continuously until the limit is reached.

One particular observation from figures 4.51 and 4.52 is that the camera is continuously gazing just above the torus. This can be counter-intuitive, but can be explained by taking the FoV and the effect of β into account. The effect of β decreases the spring torque as it moves closer to the physical limit. This motion is in the same direction as the torus moves while the base motion progresses. The effect of the FoV (see figure 4.51) is that features around the edges of the FoV lie in the middle of the camera vertical axis, which partially compensate for the pull of the features in the middle of the visible part of the torus which is directed downwards. Figure 4.51 also shows that the camera is tilted, which is also an effect of β, as it becomes increasingly difficult to rotate in the negative tilt direction. To compensate, the camera still tries to move closer to the features by panning to the left, which result in the tilted orientation of the camera.

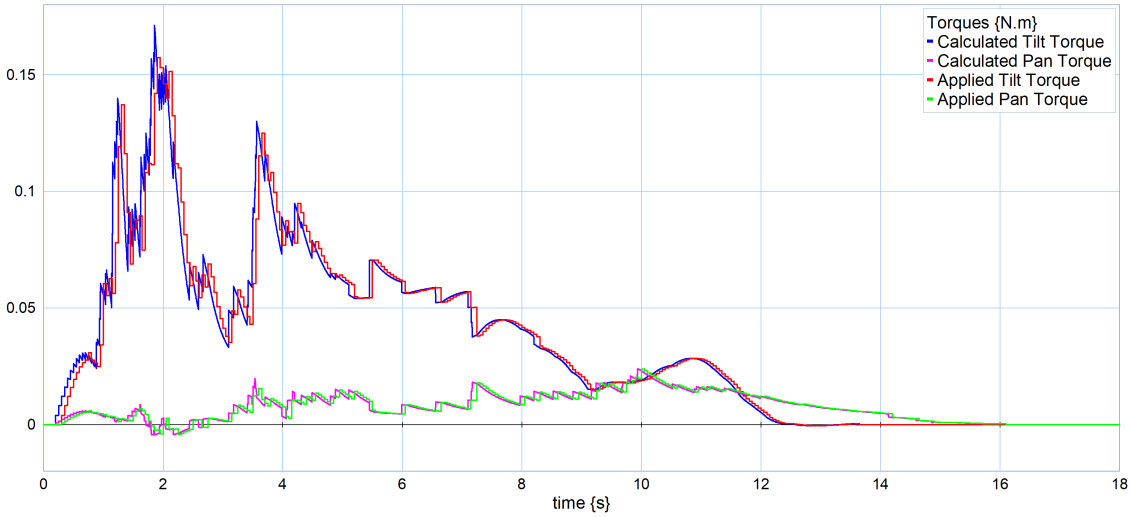


Figure 4.48: Simulated torques while moving through a torus-shaped feature cloud.

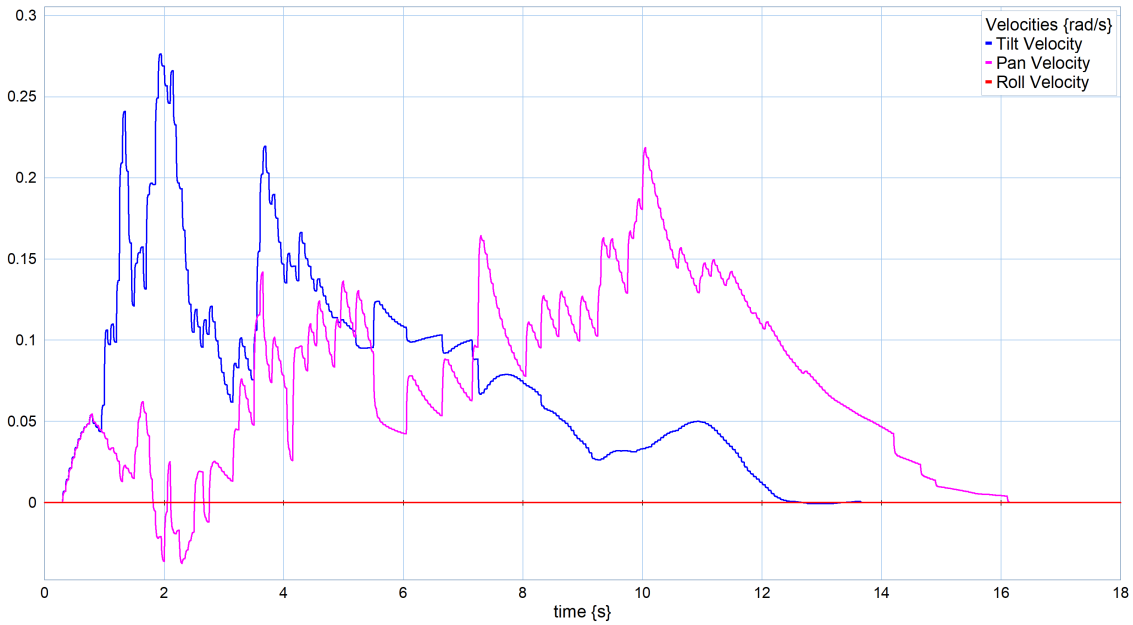


Figure 4.49: Simulated velocities while moving through a torus-shaped feature cloud.

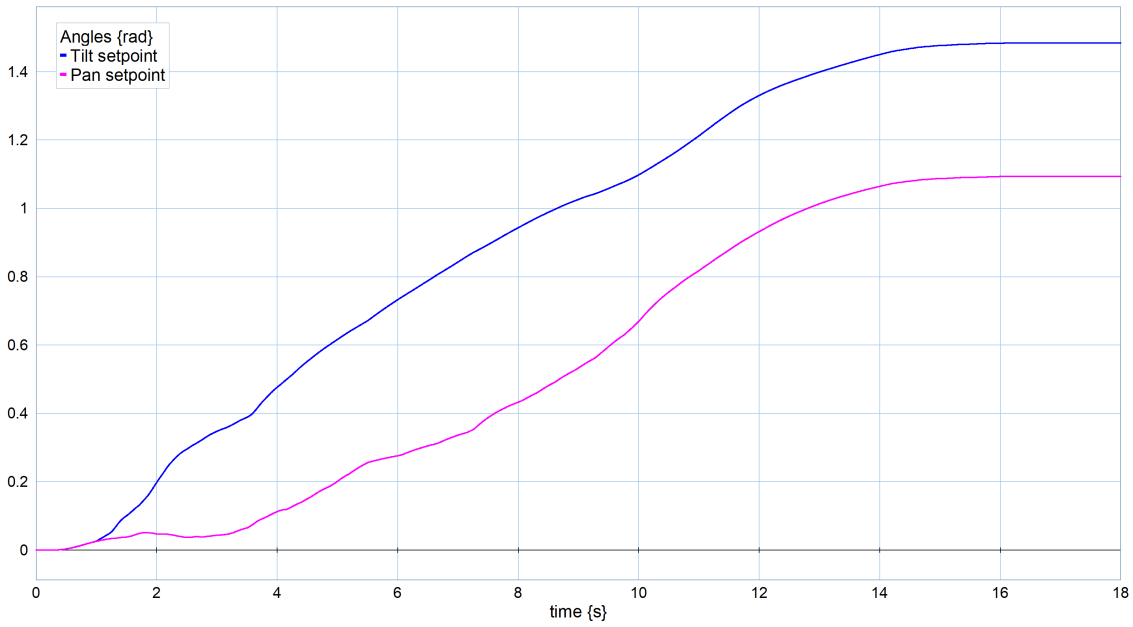


Figure 4.50: Simulated camera angles while moving through a torus-shaped feature cloud.

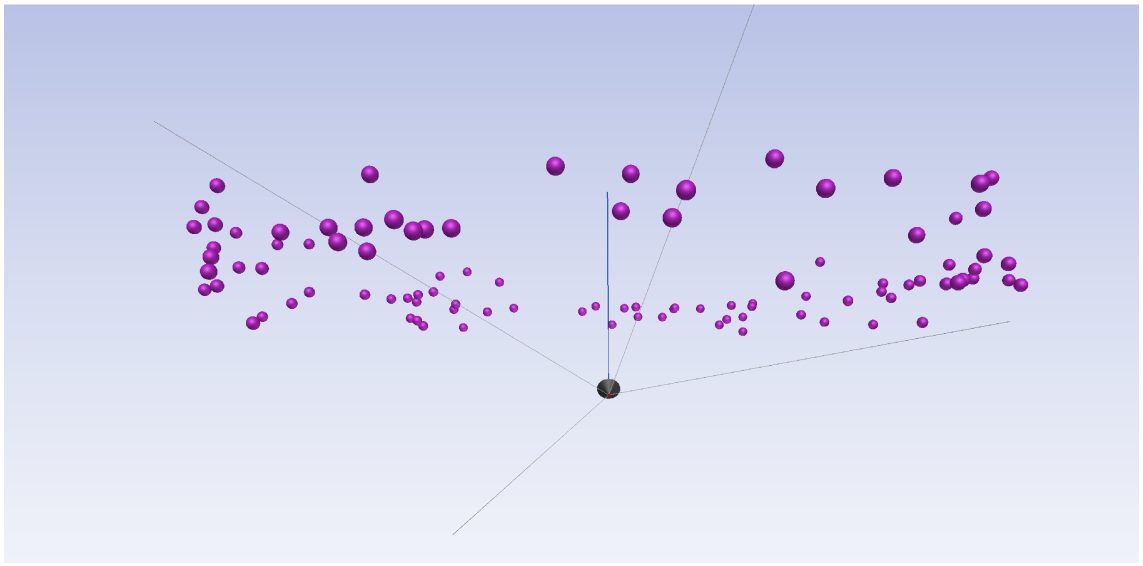


Figure 4.51: 3D visualization of the camera angle while moving through the torus-shaped feature cloud at $t = 8.1\,\text{s}$. The field-of-view is visualized with the grey lines.

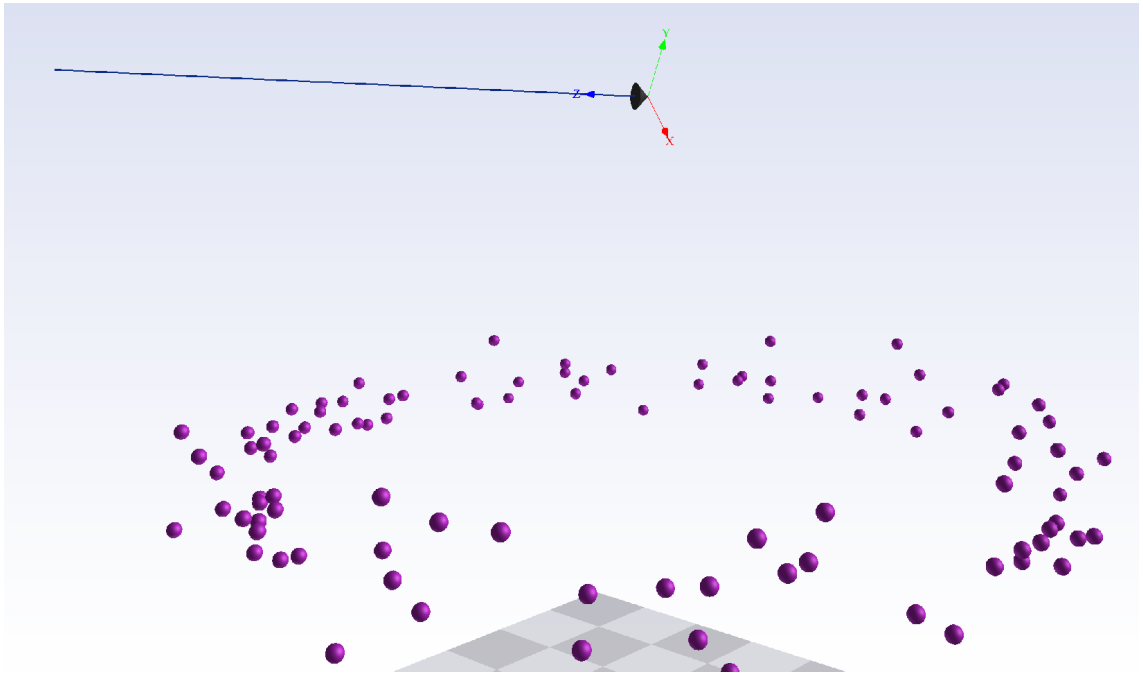


Figure 4.52: 3D visualization of the camera angle while moving through the torus-shaped feature cloud at $t = 15.3\,\text{s}$.

4.5.3 Wall with patches

The third situation under investigation is a wall with patches: a two-dimensional surface with three adjacent regions of 2 × 2 m. The outer two regions are feature rich, they contain 45 features, and the middle region contains only 10 features. This patched wall is placed as a horizontal ceiling at a height of $h = 1\,\text{m}$.

The trajectory of the base velocity is changed to a trapezoidal reference profile (see equation 4.16 and figure 4.53) to reduce the shock due to the acceleration and to see the initial and final behaviour when the velocity is zero.

$$v_b(t) = \begin{cases} 0.4 \cdot (t-2) & 2 \le t < 3 \\ 0.4 & 3 \le t < 12 \\ -0.4 \cdot (t-12) + 0.4 & 12 \le t < 13 \\ 0 & \text{elsewhere} \end{cases} \tag{4.16}$$

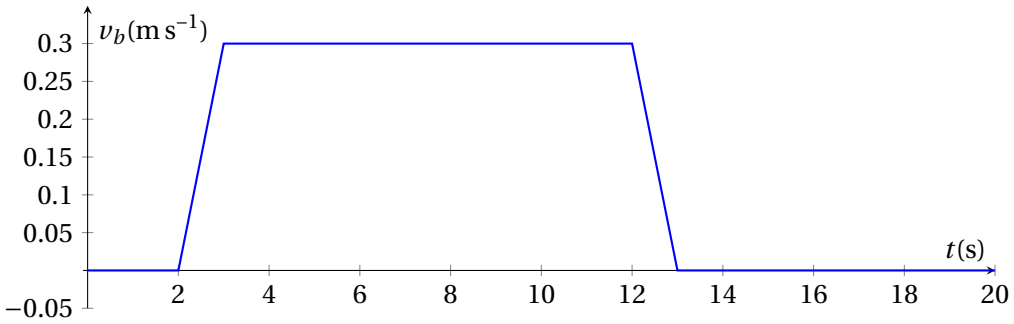


Figure 4.53: Trapezoidal velocity reference profile

The response to this situation is visualized in figures 4.54, 4.55 and 4.56. This response shows that initially, because the features on the wall are not symmetrically placed, the spring torque is unbalanced and the camera starts rotating towards a slight pan and tilt angle. As soon as the base starts moving the pan angle increases to a maximum value of $\theta(12.1) = 1.06\,\text{rad}$ as the first patch of wall moves away from the camera. The tilt angle does not increase much to a maximum value of $\phi(4.6) = 0.3\,\text{rad}$. As the motion is stopped, the pan angle decreases slightly to a final value of $\theta(20) = -1.1\,\text{rad}$, as the second patch comes partially into the FoV. The tilt angle settles on a value of $\phi(20) = 0.12\,\text{rad}$. A snapshot of the end situation is visualized in figure 4.57.

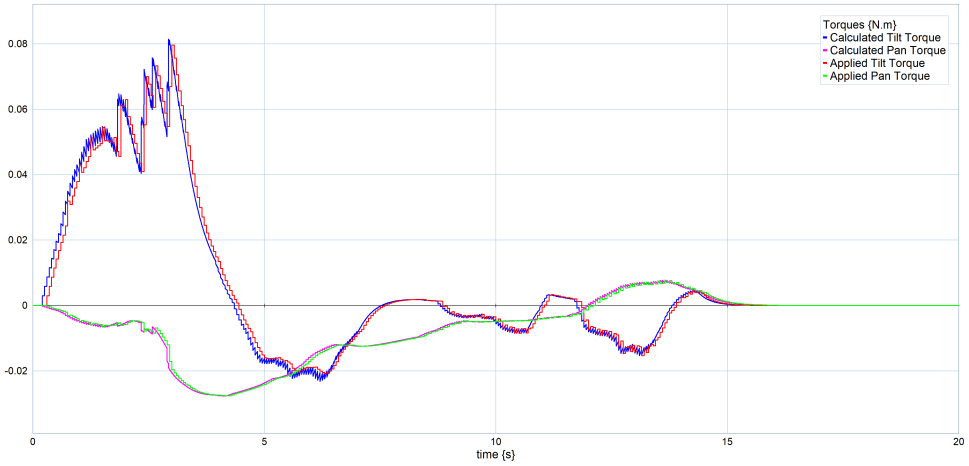


Figure 4.54: Simulated torques while moving under a wall with patches.

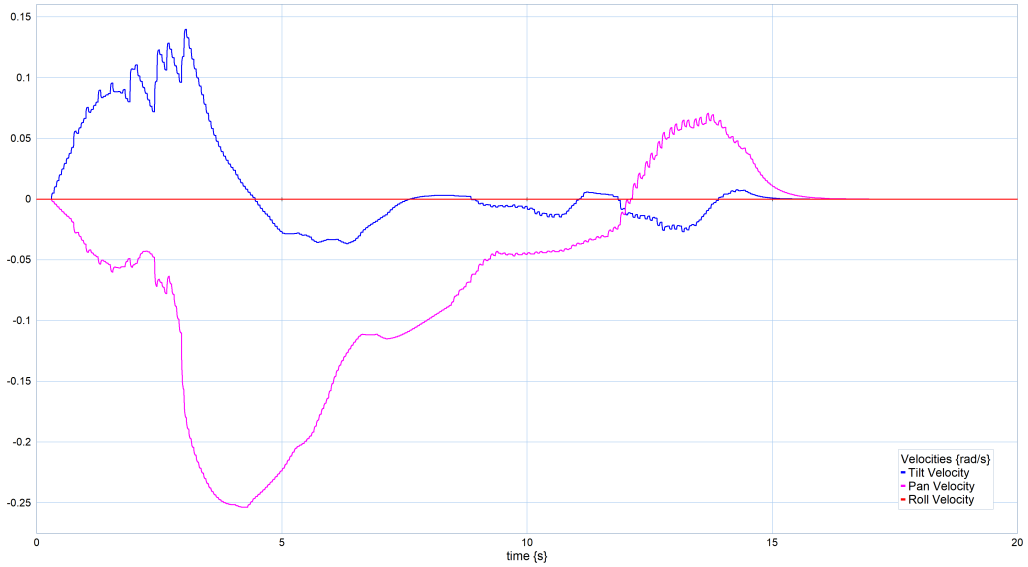


Figure 4.55: Simulated velocities while moving under a wall with patches.

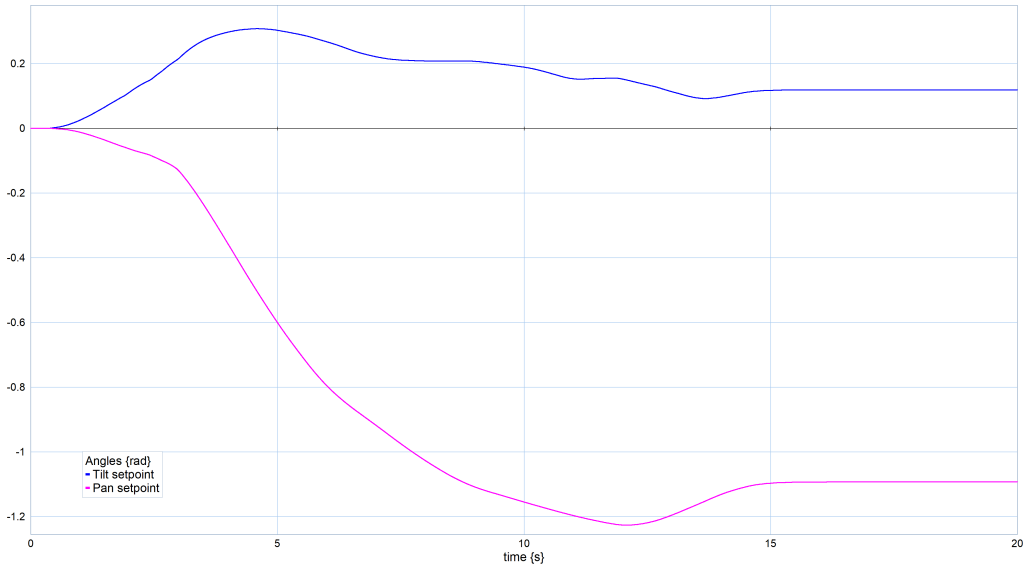


Figure 4.56: Simulated camera angles while moving under a wall with patches.

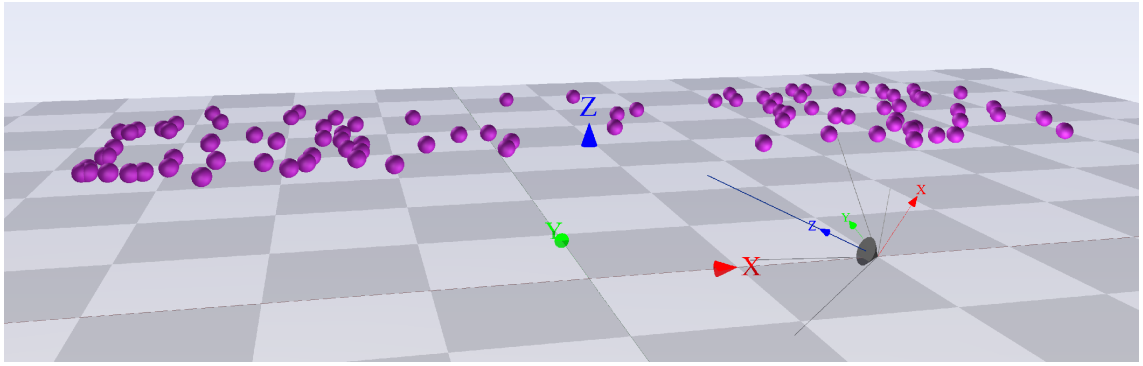


Figure 4.57: 3D visualization of the camera angle while moving under a wall with patches at $t = 20\,\text{s}$. The grey lines indicate the field-of-view.

Let us define the concept of a viewing angle ξ_p as the angle between the normal at a point p on a plane and the ray going from the camera to that point p (see figure 4.58).

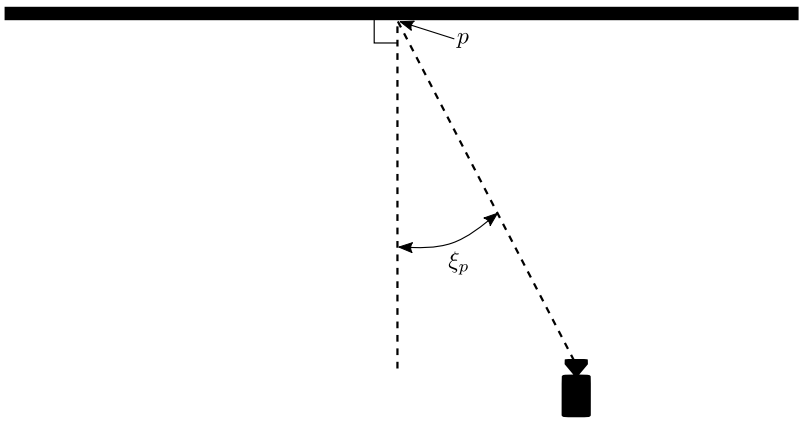


Figure 4.58: Viewing angle ξ_p, defined as the angle between the normal at a point p on a plane and the ray going from the camera to that point p.

The planar cloud of features, extracted from the wall, inherently causes the camera to diverge from the neutral position in the current simulations, as a larger viewing angle with respect to the intersection between the optical axis of the camera and the wall ξ_o allows for more features to be within the field-of-view. This is an unrealistic effect, because in physical experiments this larger angle will partially conceal the features, and ROVIO will not be able to extract all of these features. In this planar scenario the viewing angle with respect to each feature ξ_{f_i} is a desired effect, because the concealment reduces the effect of the unbalanced spring torque described above. Due to concealment, the effect of features will reduce with an increasing viewing angle ξ_{f_i}. That will most likely result in a different behaviour, where the camera will prefer areas with a smaller viewing angle ξ_{f_i}, and as such are located closer.

In this specific simulation scenario, the shortest distance to the features is consistent along the complete wall of features, which is the height h. Therefore, the viewing angle ξ_{f_i} is in this case directly related to the Euclidean distance d_i between the camera and the features through trigonometry. A maximum distance value $d_{i,\max}$ is then related to a maximum viewing angle $\xi_{f_i,\max}$.

In order to simulate $\xi_{f_i,\max}$, a fourth penalty to the spring torque is introduced, dubbed δ. δ is a binary penalty of the following form:

$$\delta(d_i) = \begin{cases} 1 & 0 < d_i \leq d_{i,\max} \\ 0 & \text{elsewhere} \end{cases} \tag{4.17}$$

The maximum distance is chosen to be $d_{i,\max} = 2\,\text{m}$. This results in a maximum viewing angle of $\xi_{f_i,\max} = \arccos\left(\frac{1}{2}\right) = \frac{\pi}{3}\,\text{rad} = 60°$. The resulting response is visible in figures 4.59, 4.60 and 4.61. The intended effect of the viewing angle limit can be observed by comparing the angular response of figures 4.56 and 4.61. The maximum pan angle is $\theta(8.3) = -0.78\,\text{rad}$ and the final pan angle is $\theta(30) = 0.02\,\text{rad}$. The maximum value for the tilt angle is $\phi(4.46) = 0.31\,\text{rad}$ and the final tilt angle $\phi(30) = -0.28\,\text{rad}$. These final angles are much closer to the center of the second high-density region, which is a beneficial result, as the viewing angles ξ_{f_i} are relatively small, which facilitates feature extraction. A snapshot of the end situation is displayed in figure 4.62.

Figure 4.59: Simulated torques while moving under a wall with patches with a limited viewing angle.

Figure 4.60: Simulated velocities while moving under a wall with patches with a limited viewing angle.

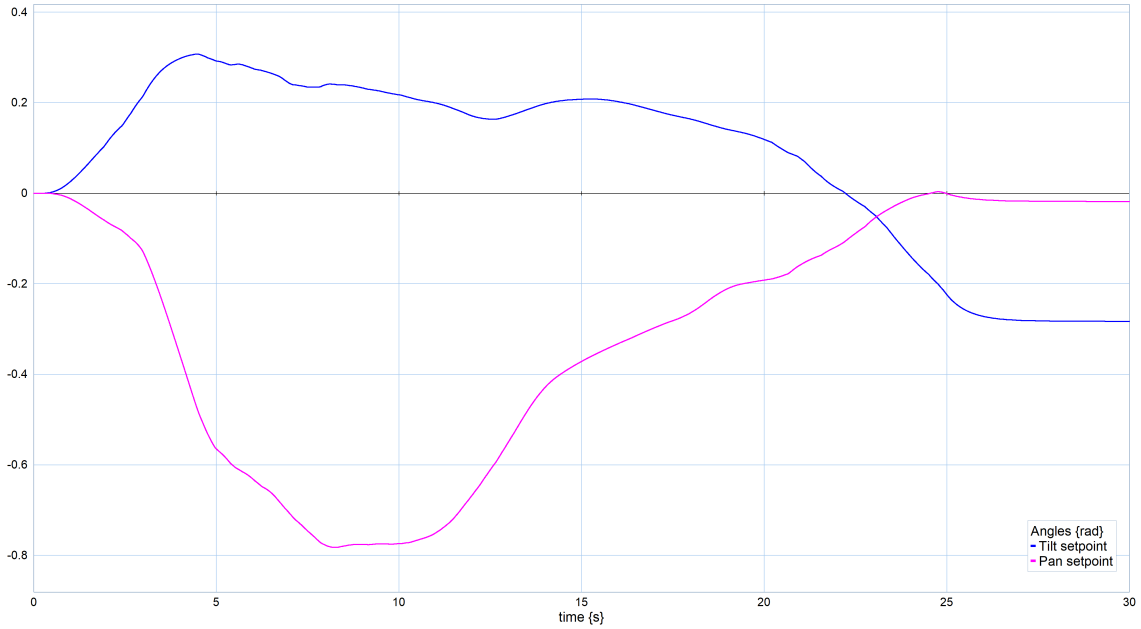


Figure 4.61: Simulated camera angles while moving under a wall with patches with a limited viewing angle.

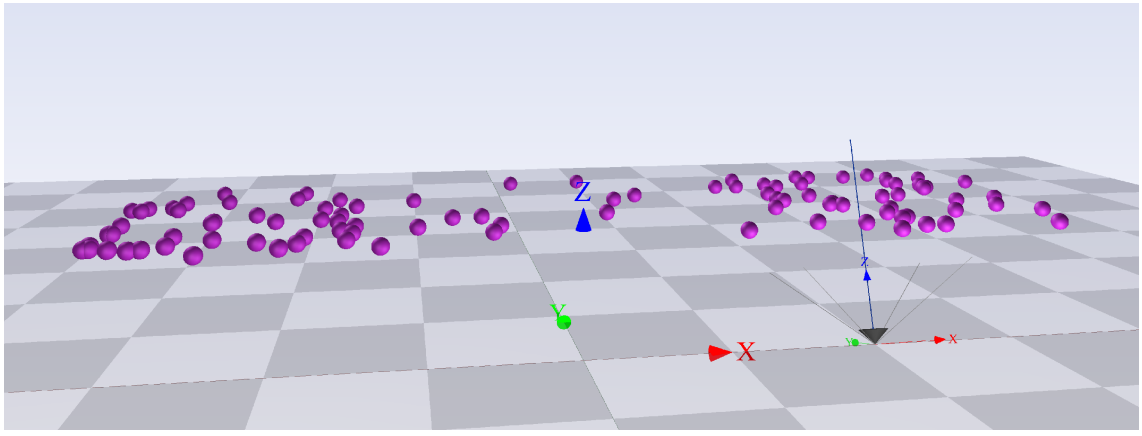


Figure 4.62: 3D visualization of the camera angle while moving under a wall with patches with a limited viewing angle at $t = 30\,\text{s}$. The grey lines indicate the field-of-view.

5 Proposed experiments

Now that the model is created, a series of experiments can be performed to evaluate the efficacy of the gaze control algorithm.

5.1 Aspects

The aspects that have to be investigated in this series of experiments are explained in this section, and a suggestion is made for how to investigate these aspects.

The first aspect of the algorithm that has to be checked is the tuning of the virtual parameters of the physical model, such as the neutral stiffness of the virtual springs, the virtual inertia and the virtual friction. Tuning of these parameters is important to ensure the model remains stable under operating conditions. Tuning of these parameters is best in a simplistic scene with only a few distinct features.

The second aspect of the algorithm that is tested is how it handles the absence of features in a confined area, in the form of a plain patch of wall. The rail is positioned in a horizontal orientation in order to have the neutral position of the robot arm orient the VI-sensor towards the wall in question.

Another aspect of the gaze control algorithm that has to be checked is the effect of the three spring torque penalties on the behaviour of the robot arm. The first of these penalties is the distance penalty α. In order to test α, a scene with a large variety of distances from the camera to objects is used to see how the algorithm reacts to the difference in distance penalty associated to each feature.

The second spring torque penalty is β: the physical limit penalty. The effect of β that is checked is the influence of β and if this influence does not impede the operation of the gaze control process. This can be checked by moving a single feature towards the limits and examining the response of the algorithm.

The last spring torque penalty is γ. The purpose of γ is to smooth the introduction of new features in the field-of-view of the camera. The effect of γ can be tested by going from a featureless region to a region filled with features, and comparing the response of the gaze controller with and without γ applied.

If this all works as desired, a final experiment is to see if the gaze control algorithm has a beneficial effect on the visual odometry process. This beneficial effect can be tested by repeating a trajectory (with and without the algorithm applied) and comparing the accuracy of the Visual Odometry estimation. In order to do this, some form of ground truth has to be available.

5.2 Final experiment

In order to check the efficacy of the complete algorithm, a repeatable trajectory is needed. Along this trajectory, scenes with differing feature quantity and/or quality should be visible. For the sake of simplicity, a linear trajectory is chosen. This trajectory is established using a cart mounted with ball bearings on a rail, which is actuated using a timing belt system (see figure 5.1). The robot arm with the VI-sensor is mounted on top of the cart. The rail is mounted on two tripods to enable easy placement at a certain elevation from the ground. The timing belt is driven by a pulley, that is driven by a servomotor. The maximum obtainable linear velocity is:

$$v_{\max} = f_{\max} \cdot \pi d_p \tag{5.1}$$

where $f_{\max}$ is the no-load rotational velocity of the servomotor and d_p is the diameter of the driving pulley.

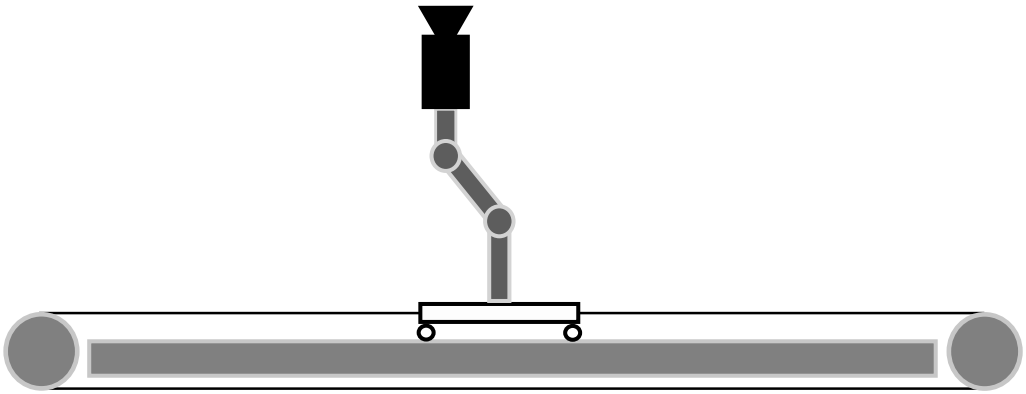


Figure 5.1: Schematic drawing of rail set-up

A constant linear velocity of the cart is desired. To include start-up acceleration and end deceleration, a trapezoidal reference profile for the angular velocity is created that is repeated in the opposite direction to end up approximately back at the starting point (see figure 5.2).

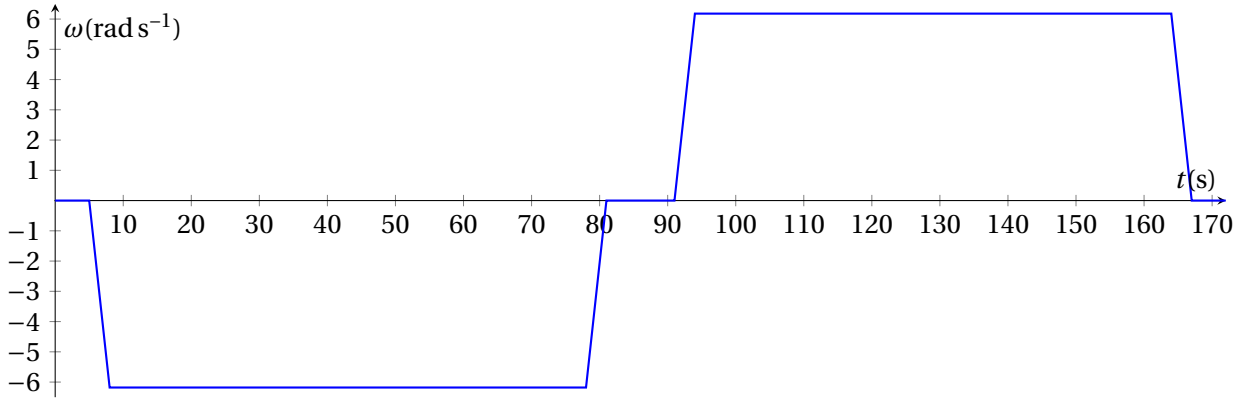


Figure 5.2: Trapezoidal velocity reference profile

The data that can be recorded consists of:

- The pan servo angular position setpoint and status sampled at 20 Hz, which contains[21]:
  - Joint name
  - Motor ID
  - Motor temperature
  - Goal position (in radians)
  - Current joint position (in radians)
  - Error between current and goal position (in radians)
  - Current joint speed (in radians per second)
  - Current load
  - If the joint is currently moving (boolean)
- The tilt servo angular position setpoint and status sampled at 20 Hz
- The belt servo angular velocity setpoint and status sampled at 20 Hz
- The stereo camera image streams sampled at 20 Hz
- The IMU data stream sampled at 200 Hz

The VI-sensor that is available, contains Lensagon BM2820 lenses, which have a horizontal field-of-view of 98° and a vertical field-of-view of 73°.[11]

6 Conclusions

This research aimed to investigate if the reliability of the visual odometry process could be improved by continuously adapting the camera angle by means of a virtual dynamical model. ROVIO was used as the VO algorithm for this investigation. By means of a series of preliminary experiments, potential pitfalls of ROVIO were analysed. ROVIO depends on both visual and IMU data. It was shown that a significant disturbance to either of these data streams resulted in a significant drift in the motion estimation. A dependency of ROVIO on feature-rich environments in an intermediate distance range was demonstrated.

A solution to the feature-dependent reliability issues was proposed. This solution consisted of manipulating the camera used for VO by means of a pan-tilt actuator, in order to continuously aim the camera towards a sufficiently feature-rich region, if possible. This manipulation is controlled by a gaze control algorithm that has been designed in this report. The setpoint of the pan-tilt actuator is determined by a virtual, dynamical model, which consists of virtual inertia, damping and springs. The virtual springs are torsional springs attached to the camera frame and the feature positions.

To reduce undesired effects, the spring constants of these virtual springs are adjusted by three penalties. These penalties take into account the Euclidean distance of a feature with respect to the camera frame, the physical limits of the pan-tilt actuator and the sudden introduction of new features within the field-of-view of the camera.

The motion induced by the virtual model was demonstrated through simulations. These simulations demonstrated that a smooth motion is obtained if tuned correctly, that is able to track features while the camera is in motion. The effect of the introduced spring torque penalties have been demonstrated to work as intended, to a limited extent. These simulations also showed limitations of the designed algorithm.

A physical experiment was proposed to demonstrate the beneficial effect of the gaze control algorithm. However, meaningful data to support the hypothesis that the designed gaze control algorithm has a beneficial effect on the VO process has not yet been obtained.

The simulation results combined with the preliminary experiments make it plausible that the designed gaze control algorithm can improve the reliability of the visual odometry process. However, due to the lack of physical, experimental data, the intended improvement by the gaze control algorithm has not been demonstrated. Further experimentation is required to determine the beneficial effect of the gaze control algorithm.

6.1 Discussions

A number of important assumptions and design choices were made that affect the results. Their effects are discussed in this section.

6.1.1 Situation dependency

Most of the aspects of the gaze control algorithm are designed with a specific situation in mind, which results in that these aspects work well for certain types of scenarios and less for other scenarios, with each aspect working well for a different scenario. Through this, a certain situation dependency of the gaze control algorithm is created. Further evaluation is required to test the boundaries of this dependency and to potentially redesign the algorithm to allow for broader boundaries.

6.1.2 Camera model

In this research, the model of the camera is assumed to be a single camera. However, the VI-sensor contains a stereo camera. The values for the field-of-view that were used, are specified for each individual lens. Therefore, this has limited the field-of-view in simulation more than is the case in the real situation. To what extent is unknown, because it depends on how ROVIO uses these datastreams exactly, which requires more intimate knowledge of ROVIO.

6.1.3 β design

The current design of the β penalty limits the available range of the pan-tilt actuator. Because β is calculated using the compound angle η, the actuator will not be able to reach an angle of $\frac{1}{2}\pi$. This means that an improvement could be reached by designing β differently by taking into account that the compound angle is used.

6.1.4 Normalization of k

The normalization of k by defining a maximum which is divided over all possible features will limit the speed of the response. However, if this process is not introduced, the possibility exists that the response will become underdamped to such a degree that it becomes unstable. In simulation this effect has more impact, since in this case $n = 100$ (i.e all simulated features). In the case of a physical test, $n = 25$, which is the maximum number of features ROVIO can take into account. Because of this, in some cases, the spring constant of the simulation will be significantly lower.

6.1.5 Concealment

One of the physical effects that the current simulation does not take into account is concealment of features. This is an effect that will happen in a physical situation, but is not incorporated in the evaluated simulations. This skews the results to an unknown extent.

6.1.6 Feature distribution

Another discrepancy that exists between the simulations and physical experiments is the distribution of features. In the simulations, the features are placed using a pseudo-random generator within a certain area. Even though a pseudo-random generator generates with a uniform chance distribution, it can create areas with varying feature-densities. In physical experiments ROVIO takes only a limited amount of features into account. One of the devaluing criteria of feature selection used by ROVIO is the proximity with respect to other features. In other words, ROVIO tries to select features from possible candidates that are as spaced out as possible. This effect has not been taken into account by the simulation.

Bibliography

[1] Mohammad OA Aqel et al. 'Review of visual odometry: types, approaches, challenges, and applications'. In: *SpringerPlus* 5.1 (2016), p. 1897. URL: https://springerplus.springeropen.com/articles/10.1186/s40064-016-3573-7.

[2] Michael Bloesch et al. 'Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback'. In: *The International Journal of Robotics Research* 36.10 (2017), pp. 1053–1072. URL: https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/187364/ROVIO.pdf?sequence=1.

[3] Michael Bloesch et al. 'Robust visual inertial odometry using a direct EKF-based approach'. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 298–304. URL: https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/155340/eth-48374-01.pdf?sequence=1.

[4] Peter Breedveld and H Unbehauen. 'Modeling and Simulation of Dynamic Systems using Bond Graphs'. In: *Encyclopedia of Life support systems* (Jan. 2008).

[5] Jan F. Broenink. *Introduction to Physical Systems Modelling with Bond Graphs*. Tech. rep. University of Twente, 1999. URL: https://www.ram.ewi.utwente.nl/bnk/papers/BondGraphsV2.pdf.

[6] Roland Bunschoten and Ben Krose. 'Visual odometry from an omnidirectional vision system'. In: *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. Vol. 1. IEEE. 2003, pp. 577–583. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1241656.

[7] Peter Corke, Dennis Strelow and Sanjiv Singh. 'Omnidirectional visual odometry for a planetary rover'. In: *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 4. IEEE. 2004, pp. 4007–4012. URL: http://eprints.qut.edu.au/32778/1/32778_corke.pdf.

[8] Javier Correa and Alvaro Soto. 'Active visual perception for mobile robot localization'. In: *Journal of Intelligent and Robotic Systems* 58.3-4 (2010), pp. 339–354.

[9] Gabriele Costante et al. 'Perception-aware path planning'. In: *arXiv preprint arXiv:1605.04151* (2017). URL: https://arxiv.org/pdf/1605.04151v2.

[10] Stefan Cruceanu. *Free Response of a Spring Mass System*. Tech. rep. M340. Colorado State University, 2004. URL: http://www.ima.ro/people/scruceanu/Teaching/Spring2004/M340/Notes/FreeResponse.pdf.

[11] Lensation GmbH. *Lensagon BM2820 Datasheet*. Sept. 2017. URL: https://www.lensation.de/pdf/BM2820.pdf.

[12] Jindřich Havlík and Ondřej Straka. 'Performance evaluation of iterated extended Kalman filter with variable step-length'. In: *Journal of Physics: Conference Series*. Vol. 659. 1. IOP Publishing. 2015, p. 012022. URL: http://iopscience.iop.org/article/10.1088/1742-6596/659/1/012022.

[13] Albert S Huang et al. 'Visual odometry and mapping for autonomous flight using an RGB-D camera'. In: *Robotics Research*. Springer, 2017, pp. 235–252. URL: http://www.cs.washington.edu/research/projects/aiweb/media/papers/Huang-ISRR-2011.pdf.

[14] Jonathan Kelly and Gaurav S Sukhatme. 'An experimental study of aerial stereo visual odometry'. In: *IFAC Proceedings Volumes* 40.15 (2007), pp. 197–202. URL: http://www.academia.edu/download/45398299/543.pdf.

[15] Travis Manderson, Florian Shkurti and Gregory Dudek. 'Texture-aware SLAM using stereo imagery and inertial information'. In: *Computer and Robot Vision (CRV), 2016 13th Conference on*. IEEE. 2016, pp. 456–463. URL: http://www.cim.mcgill.ca/~mrl/pubs/travism/crv_2016_texture_aware_slam.pdf.

[16] Bill Messner et al. *Control Tutorials for Matlab & Simulink*. URL: http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction%5C&section=SystemModeling.

[17] Raul Mur-Artal, Jose Maria Martinez Montiel and Juan D Tardos. 'ORB-SLAM: a versatile and accurate monocular SLAM system'. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[18] Janosch Nikolic et al. 'A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM'. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 431–437. URL: http://e-collection.library.ethz.ch/eserv/eth:8003/eth-8003-01.pdf.

[19] Valentin Peretroukhin, Jonathan Kelly and Timothy D Barfoot. 'Optimizing camera perspective for stereo visual odometry'. In: *Computer and Robot Vision (CRV), 2014 Canadian Conference on*. IEEE. 2014, pp. 1–7. URL: http://starslab.ca/wp-content/papercite-data/pdf/2014_peretroukhin_optimizing.pdf.

[20] Brent Perreault. *Introduction to the Kalman Filter and its Derivation*. 2012. URL: https://www.academia.edu/1512888/Introduction_to_the_Kalman_Filter_and_its_Derivation.

[21] Antons Rebguns. *dynamixel_msgs/JointState Documentation*. ROS.org. 2011. URL: http://docs.ros.org/kinetic/api/dynamixel_msgs/html/msg/JointState.html.

[22] Davide Scaramuzza and Friedrich Fraundorfer. 'Visual odometry [tutorial]'. In: *IEEE robotics & automation magazine* 18.4 (2011), pp. 80–92. URL: http://www.ifi.uzh.ch/dam/jcr:5759a719-55db-4930-8051-4cc534f812b1/VO_Part_I_Scaramuzza.pdf.

[23] Davide Scaramuzza and Roland Siegwart. 'Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles'. In: *IEEE transactions on robotics* 24.5 (2008), pp. 1015–1026. URL: https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/14362/eth-30946-01.pdf?sequence=1.

[24] Stefano Stramigioli and Herman Bruyninckx. 'Geometry and Screw Theory for Robotics'. In: IEEE ICRA. Mar. 2001. URL: https://archive.org/details/Stefano_Stramigioli_and_Herman_Bruyninckx___Geometry_and_Screw_Theory_for_Robotics.

[25] J Straub et al. 'Fast relocalization for visual odometry using binary features'. In: *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE. 2013, pp. 2548–2552. URL: https://www.researchgate.net/profile/Julian_Straub/publication/237048601_Fast_Relocalization_For_Visual_Odometry_Using_Binary_Features/links/572b8f2608ae2efbfdbdd978.pdf.

[26] Jean-Philippe Tardif, Yanis Pavlidis and Kostas Daniilidis. 'Monocular visual odometry in urban environments using an omnidirectional camera'. In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE.

2008, pp. 2531–2538. URL: https://pdfs.semanticscholar.org/10a5/f5bd0f35a80f0c49b4ed812aec5af86bf955.pdf.

[27] Wikipedia contributors. *List of moments of inertia*. [Online; accessed 20-December-2018]. Wikipedia, The Free Encyclopedia. 2018. URL: https://en.wikipedia.org/w/index.php?title=List_of_moments_of_inertia.

[28] Zichao Zhang et al. 'Benefit of large field-of-view cameras for visual odometry'. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 801–808. URL: https://www.ifi.unizh.ch/dam/jcr:e104c901-5d1b-4264-a03d-e65e3b3ec7c6/ICRA16_Zhang.pdf.