ER-Force
Team Description Paper for RoboCup 2012

Jan Kallwies, Simon Dirauf, Philipp Nordhus,
Simon Kerschbaum, Stefan Friedrich, and Michael Bleier

Robotic Activities Erlangen e.V.
Pattern Recognition Lab, Department of Computer Science
University of Erlangen-Nuremberg
Martensstr. 3, 91058 Erlangen, Germany
`info@robotics-erlangen.de`
`http://www.robotics-erlangen.de/`

Abstract. This paper presents an overview description of ER-Force, the RoboCup Small Size League team from Erlangen, Germany. The current tracking algorithms and processing stack is outlined. The main part of the paper gives a detailed description of the model-based motion control implemented on the robots and trajectory planning and position control algorithms used on the external computer. Furthermore, upcoming changes and improvements are outlined.

1 Introduction

In this paper the motion control and trajectory planning components of the RoboCup team *ER-Force* is presented. The team, located at Friedrich-Alexander-University of Erlangen-Nuremberg in Germany, participates in RoboCup competitions since 2007. This year focus lies on improving the motion control algorithms to provide faster reaction times and on adjusting both the position and the orientation of the robots. In section 2 the tracking algorithm and the organization of data flow in the system is presented. The design and implementation of the control loops for velocity and position as well as the basic planning algorithms for valid trajectories is illustrated in section 3.

2 State Estimation and Tracking

The task of object detection has been greatly simplified by the introduction of SSL-Vision [1]. This software is able to identify objects reliably in most situations. However, since it does not include any form of tracking, detection will fail eventually. For instance, the ball will not be detected if it is occluded by a robot due to perspective projection. Furthermore, among other situations, fluctuations in lighting conditions may render an object temporarily invisible.

It is necessary for the control algorithms presented in section 3 to have valid data on each iteration. If no position data is available at any iteration, the software will fail to generate movement commands. Therefore, a tracking system has been implemented. The well-known Kalman Filter [2] is applied to filter noise in the detection data from SSL-Vision. This algorithm also tracks objects, i.e. it provides an estimate of the current position and velocity even if there are no new measurements available.

For optimal control the delay between receiving new vision data and the transmission of the control action to the robots has to be kept as small as possible. Moreover, for easier analysis and design of the control algorithms it is preferred that all low-level control loops run with a fixed frequency. SSL Vision does not provide any guarantees about the timing of the vision data packets since both cameras (or even all four with a large field) are triggered independently and the processing threads are not synchronized. Hence, the information from each individual processing stack is received with a small, non-constant delay and motion control cannot simply be executed whenever new vision data is available. In this implementation, a dedicated thread first runs the tracking algorithm and directly afterwards motion control calculations. As a compromise, this thread is triggered with a fixed frequency of 120Hz which relates to approximately the double camera frame rate. Upon activation the thread first iterates over all vision packets received since the last activation, in chronological order. For each packet, all currently tracked objects are predicted to the time when the packet was received. This *receive timestamp* is calculated by subtracting the SSL-Vision runtime from the time when the packet was received on the control computer. Any additional latency by Firewire and UDP communication is neglected. After prediction, each object is updated with the measurements from this vision packet. When all vision packets have been processed, all filters are predicted to the current time. Finally, motion control is run with this estimated data.

3 Motion Control

Motion control is one of the most important areas in the Small Size League due to the very fast and agile robots. This topic is already addressed in several works, e.g. [3] or [4].

In this paper, however, a new approach for motion control, in particular the velocity control on the robot, is presented.

The entire motion control system is divided into two parts, the *velocity control processed on the robot* and the *position control on the supervising computer*. The position control's main purpose is to calculate the way and the corresponding velocities of the robot through the waypoints generated by the path-finding in an adequate way. The desired velocities are subsequently sent to the robot by radio. Finally, the velocity control on the robot ensures that the robot motion complies to the desired velocity.

3.1 Velocity Control

The velocity control system takes care of the velocity in x- and y-direction as well as the rotational speed of the robot. The corresponding coordinate system of a four-wheeled robot can be seen in Fig. 1.

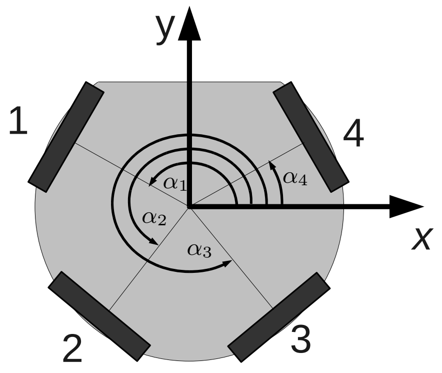


Fig. 1. Model of the robot including the axes and the angles of the motors

Model-based feed-forward control The first step of designing a control system is to think about a reasonable feed-forward system that controls the system correctly under optimal conditions. The actual feedback controller's only purpose is then to tackle disturbances caused by uncertainties such as model inaccuracy and extraneous impacts. This approach, as presented e.g. in [4] and [5], achieves much better results compared to a classical feedback-only control architecture by implementing a two degrees of freedom control system consisting of a feed-forward and a feedback system.

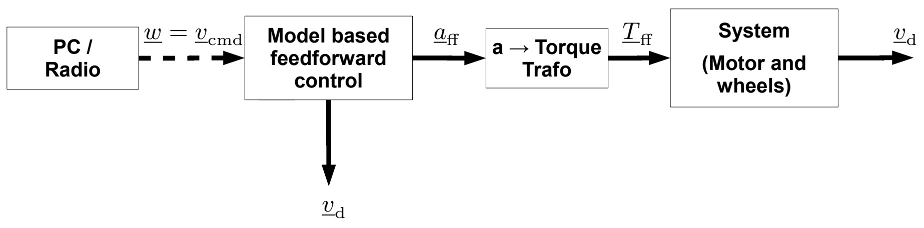

PC / Radio
$\underline{w} = \underline{v}_{\mathrm{cmd}}$
Model based feedforward control
$\underline{a}_{\mathrm{ff}}$
a → Torque Trafo
$\underline{T}_{\mathrm{ff}}$
System (Motor and wheels)
$\underline{v}_{\mathrm{d}}$
$\underline{v}_{\mathrm{d}}$

Fig. 2. Structure of the feed-forward system for the velocity control

Following such an approach we implemented a control mainly consisting of a model-based feed-forward control as it can be seen in Fig. 2.

The first block in Fig. 2 represents the model-based feed-forward control which generates a trajectory of the desired velocities

$$\underline{v}_{\mathrm{d}} = \begin{bmatrix} v_{\mathrm{d},x} \\ v_{\mathrm{d},y} \\ \omega_{\mathrm{d}} \end{bmatrix} \tag{1}$$

and appropriate values for the acceleration

$$\underline{a}_{\mathrm{ff}} = \begin{bmatrix} a_{\mathrm{ff},x} \\ a_{\mathrm{ff},y} \\ a_{\mathrm{ff},\omega} \end{bmatrix} \tag{2}$$

which are suitable for achieving the desired behavior of the velocities $\underline{v}_{\mathrm{d}}$.

How these trajectories and control values are generated in detailed is depicted in Fig. 3. Basically, we shape the real system consisting of the motors and the wheels in a model and build a controller around it. As a setpoint we use the desired velocities $\underline{v}_{\mathrm{cmd}}$ received by radio. The controller is a simple state controller:

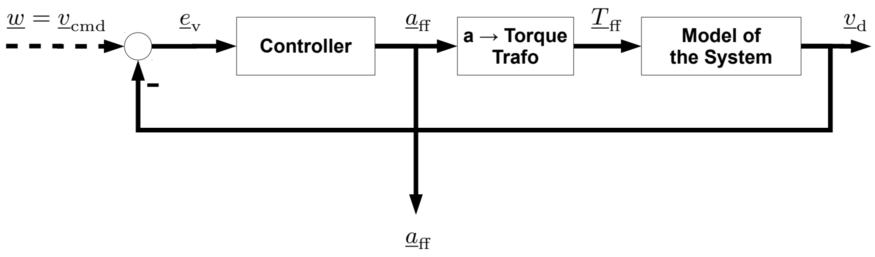
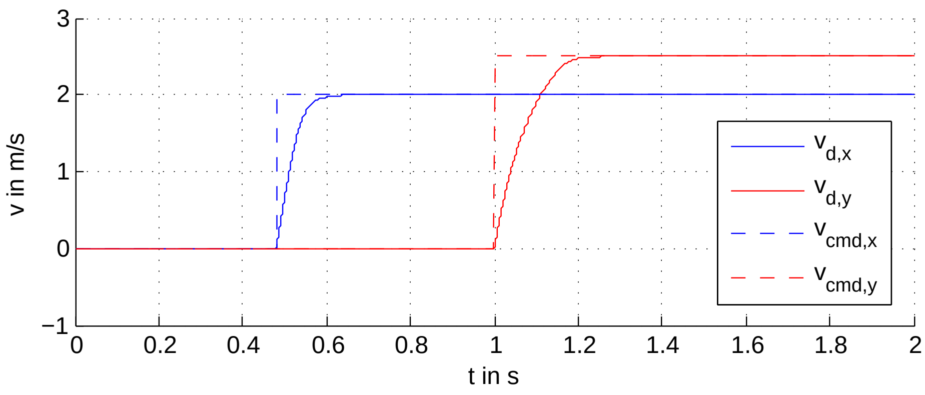

$\underline{w} = \underline{v}_{\mathrm{cmd}}$
$\underline{e}_{\mathrm{v}}$
Controller
$\underline{a}_{\mathrm{ff}}$
a → Torque Trafo
$\underline{T}_{\mathrm{ff}}$
Model of the System
$\underline{v}_{\mathrm{d}}$
–
$\underline{a}_{\mathrm{ff}}$

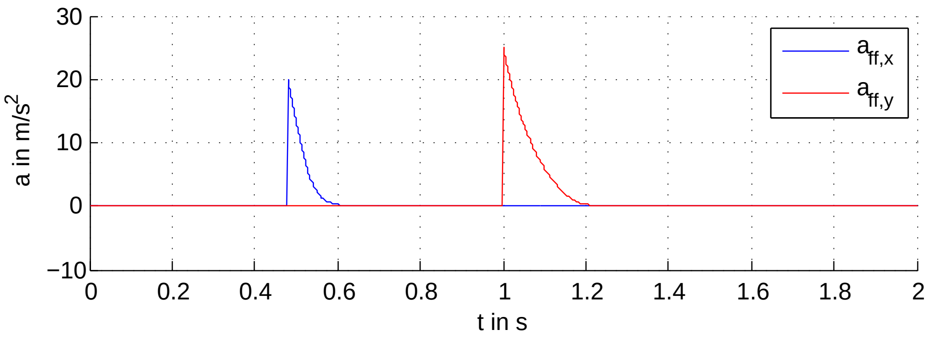
Fig. 3. The model-based feed-forward control

$$\underline{a}_{\mathrm{ff}} = \underline{K}_{\mathrm{ff}} \cdot \underline{e}_{v} = \begin{bmatrix} k_{\mathrm{ff},1} & 0 & 0 \\ 0 & k_{\mathrm{ff},2} & 0 \\ 0 & 0 & k_{\mathrm{ff},3} \end{bmatrix} \cdot \begin{bmatrix} v_{\mathrm{cmd},x} - v_{\mathrm{d},x} \\ v_{\mathrm{cmd},y} - v_{\mathrm{d},y} \\ v_{\mathrm{cmd},\omega} - v_{\mathrm{d},\omega} \end{bmatrix} \tag{3}$$

The controller parameters in $\underline{K}_{\mathrm{ff}}$ can be used to adjust the command response individually for all three dimensions. With higher values for the parameters faster responses to a change of $\underline{v}_{\mathrm{cmd}}$ can be achieved. However, the command response must not be made too fast since it may no longer be possible to bring up the required torque because of the limited voltage and current that can be applied to the motors. This problem can be avoided by taking actuator signal limitations in the model-based feed-forward control into account.



(a) Velocities



(b) Accelerations

Fig. 4. Example trajectories generated by the model-based feed-forward control

In Fig. 4 one can see an example of the output of the model-based feed-forward control.

Please note that the robot will perform exactly the desired trajectories $\underline{v}_{\mathrm{d}}$ as plotted in Fig. 4(a) if the values for the acceleration $\underline{a}_{\mathrm{ff}}$ as plotted in Fig. 4(b) are used as input values, assuming that the model used for the model-based feed-forward control is exact and there is no disturbances at all.

Therefore, the feed-forward system outlined in Fig. 2 will control the robot in the desired way without any feedback path.

Acceleration to Torque transformation The output of the primary control system as described above is a desired acceleration for the robot:

$$\underline{a} = \begin{bmatrix} a_x \\ a_y \\ \dot{\omega} \end{bmatrix} \tag{4}$$

Since the acceleration cannot be used directly as input for the motors it has to be transformed into a corresponding setpoint torque in the form of:

$$\underline{T}_{\text{set}} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \tag{5}$$

This transformation is not unique because there are three dimensions of acceleration transformed to four motor torques. Basically three motors would be enough for driving to the x- and y-direction and the rotation. Thus we have one redundant element in the torque vector $\underline{T}_{\text{set}}$. This can easily be seen in Fig. 5.

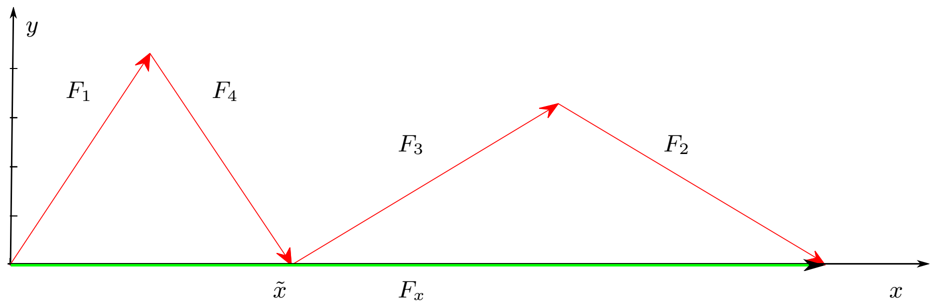


Fig. 5. The desired force in x-direction F_x divided into the four motor forces

A straight forward appraoch to use this redundancy is to equalize all four motor torques for moving in x- and y-direction.

$$T_1 = T_2 = T_3 = T_4. \tag{6}$$

In this case the point $\tilde{x}$ describes the distribution of the force in x-direction between the wheels with different angles and can be evaluated by

$$\tilde{x} = \frac{\sin(\alpha_1)}{|\sin(\alpha_2)| + \sin(\alpha_1)}. \tag{7}$$

Analogously the point $\tilde{y}$ can be evaluated for moving in y-direction:

$$\tilde{y} = \frac{\cos(\alpha_4)}{\cos(\alpha_4) + \cos(\alpha_3)} \tag{8}$$

To rotate the robot, i.e. driving in φ-direction, the y-components of each corresponding wheels 1–4 and 2–3 neutralize themselves. The x-components of the corresponding wheels are added and their sum has to be zero.

With these conditions the force feed-forward matrix $\underline{F}_{\mathrm{ff}}$ can be written as

$$\underline{F}_{\mathrm{ff}} = \begin{bmatrix} -\frac{\tilde{x}}{2\sin(\alpha_1)} & \frac{\tilde{y}}{2\cos(\alpha_1)} & \frac{1}{2r\frac{1-\sin(\alpha_1)}{\sin(\alpha_2)}} \\ -\frac{1-\tilde{x}}{2\sin(\alpha_2)} & \frac{1-\tilde{y}}{2\cos(\alpha_2)} & \frac{1}{2r\frac{1-\sin(\alpha_2)}{\sin(\alpha_1)}} \\ -\frac{1-\tilde{x}}{2\sin(\alpha_3)} & \frac{1-\tilde{y}}{2\cos(\alpha_3)} & \frac{1}{2r\frac{1-\sin(\alpha_2)}{\sin(\alpha_1)}} \\ -\frac{\tilde{x}}{2\sin(\alpha_4)} & \frac{\tilde{y}}{2\cos(\alpha_4)} & \frac{1}{2r\frac{1-\sin(\alpha_1)}{\sin(\alpha_2)}} \end{bmatrix} \tag{9}$$

where α_i are the angles from the x-axis to the single wheel axes (see Fig. 1) and r is the radius of the robot.

With this matrix we can compute the setpoint torque vector:

$$\underline{T}_{\mathrm{set}} = \underline{F}_{\mathrm{ff}} \begin{bmatrix} a_x \\ a_y \\ \dot{\omega} \end{bmatrix} \tag{10}$$

The controlled system – the robot with motors and wheels The model the feed-forward control is based on can be seen in Fig. 6.

The complete dynamics of the motors including the torque control is approximated by a first order lag element (PT1 element). By dividing the torques T_i by the wheel radius r the forces F_i can be calculated.

With the help of the force coupling matrix $\underline{F}_{\mathrm{c}}$, the forces of the single motors are added vectorially to the resulting forces in the three moving directions x, y, and φ

$$\begin{bmatrix} F_{\mathrm{x}} \\ F_{\mathrm{y}} \\ T_{\varphi} \end{bmatrix} = \underline{F}_{\mathrm{c}} \cdot \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} . \tag{11}$$

Where F_1, F_2, F_3 and F_4 are the forces of the single motors, $\underline{F}_{\mathrm{c}}$ is the force coupling matrix and F_x, F_y and T_φ are the resulting forces for the robot.

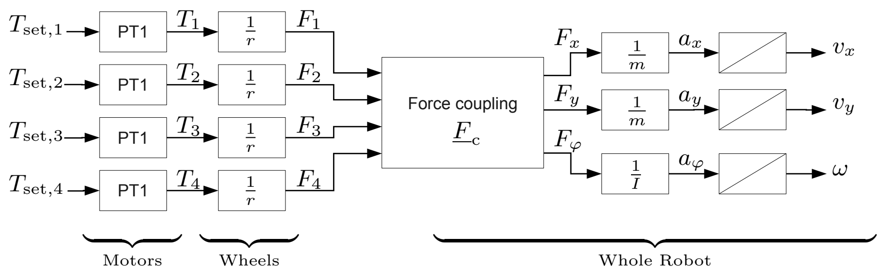


Fig. 6. Model of the controlled system

The force coupling matrix is given by

$$\underline{F}_{\mathrm{c}} = R_{\mathrm{G}} \cdot \begin{bmatrix} -\sin(\alpha_1) & -\sin(\alpha_2) & -\sin(\alpha_3) & -\sin(\alpha_4) \\ \cos(\alpha_1) & \cos(\alpha_2) & \cos(\alpha_3) & \cos(\alpha_4) \\ r & r & r & r \end{bmatrix} \tag{12}$$

where R_{G} is the gear ratio between motors and wheels, α_i are the angles of the four wheel axes to the x-axis as in Fig. 1 and r is the radius of the robot.

With equation (11) the forces driving the robot can be calculated and the final accelerations can be derived by dividing them by the mass of the robot m and the moment of inertia I, respectively.

Torque Control The interface between the velocity control for the whole robot and the torque control per motor is simply a setpoint for the torque. Note that the single wheel speeds are not controlled at all. We think that it is much more reasonable to control torque instead of rotational wheel speeds.

Our approach for the torque control is shown in Fig. 7. As a model we used a simple model of a DC motor as described in [6] whereas n_i is the current rotational speed, U_{M} the applied voltage, R_{A} the resistance of the anchor, I_{A} the current through the anchor and T_i the currently available torque. The motor constant k_{M} measured in Nm/A is used to calculate the torque $T = I_A \cdot k_{\mathrm{M}}$ and k_{EMF} measured in V/rpm is used to estimate the back EMF voltage U_{EMF}.

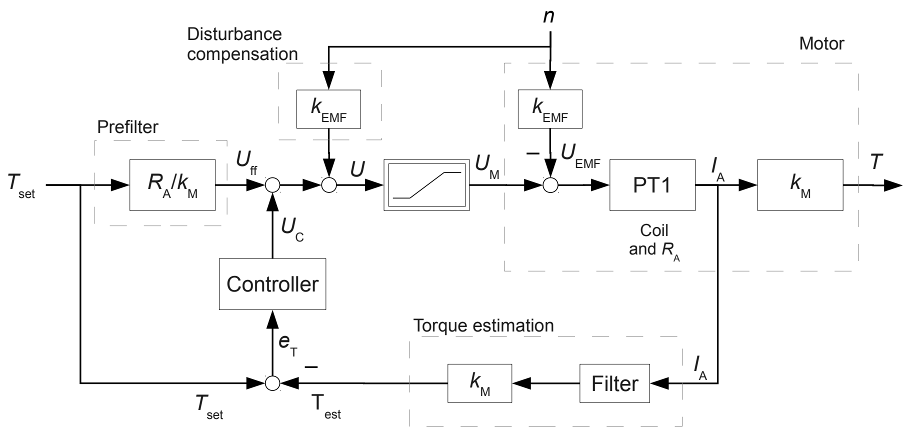


Fig. 7. Structure of the torque control per motor

The torque control consists of three parts to calculate the control input which is the applied motor voltage.

We do not control the rotational wheel speeds, but we take them into account in terms of the torque control. This way we can treat the back EMF voltage which reduces the voltage available for acceleration (see Fig. 7) as a measurable disturbance. Thus the first step is to compensate this effect by adding $U_\mathrm{n} = n \cdot k_\mathrm{EMF}$.

The next step is once again a feed-forward control in form of a static prefilter $U_\mathrm{ff} = T_{i,\mathrm{set}} \cdot \frac{R_\mathrm{A}}{k_\mathrm{M}}$ as it can be seen in Fig. 7. Under the assumption that the model is exact this control value will lead to a torque that is equivalent to the desired torque after a short time.

The last step is to compensate unknown disturbances or such that are not measurable, as well as model uncertainties by using a controller $U_\mathrm{C} = (T_{i,\mathrm{set}} - T_i) \cdot k_\mathrm{T}$ whereas the current torque is estimated using the measured value of the current according to Fig. 7.

Thus we can write the overall control formula as:

$$U_\mathrm{M} = \underbrace{n \cdot k_\mathrm{EMF}}_{\text{Compensate back EMF}} + \underbrace{T_{i,\mathrm{set}} \cdot \frac{R_\mathrm{A}}{k_\mathrm{M}}}_{\text{Feed-forward control}} + \underbrace{(T_{i,\mathrm{set}} - T_i) \cdot k_\mathrm{T}}_{\text{Feedback control}} . \qquad (13)$$

State controller A pure feed-forward system as described above would control the robot already quite well but will never be stationary exact because we can neither create an exact model nor neglect disturbances. Moreover, we do not know the initial conditions exactly, what would also be necessary for an exact feed-forward control. Thus we need a controller in order to compensate deviations from the desired trajectories.

The resulting complete structure of the velocity control system is shown in Fig. 8.

Since the system is completely controllable we can place the eigenvalues arbitrarily and individually for all three dimensions.

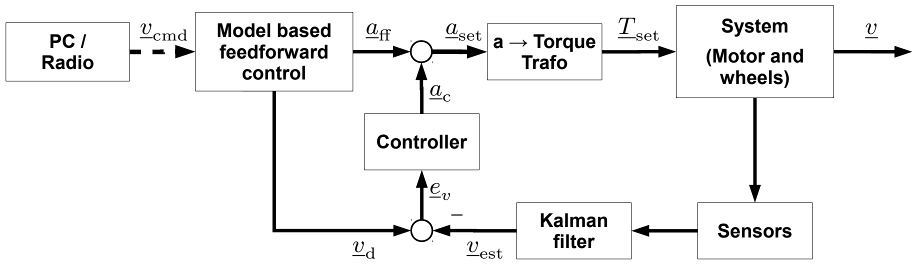

PC / Radio
$\underline{v}_\mathrm{cmd}$
Model based feedforward control
$\underline{a}_\mathrm{ff}$
$\underline{a}_\mathrm{set}$
a → Torque Trafo
$\underline{T}_\mathrm{set}$
System (Motor and wheels)
$\underline{v}$
$\underline{a}_\mathrm{c}$
Controller
$\underline{e}_v$
−
Kalman filter
Sensors
$\underline{v}_\mathrm{d}$
$\underline{v}_\mathrm{est}$

Fig. 8. The complete structure of the velocity control for the robot

Measurement and state estimation In order to be able to compensate the uncertainties using a state controller as described above we need to estimate the current state, in particular the velocity vector $\underline{v}$. Therefore, we have different measured values:

- Rotational speed n_i of the 4 wheels measured by a *magnetic encoder*
- Rotational speed ω of the robot measured by a *gyroscope*
- The acceleration in x- and y-direction measured by an *acceleration sensor*

In order to perform sensor data fusion and state estimation a Kalman Filter is used [2].

3.2 Position Control

Trajectory Planning The algorithm is implemented in the scripting language Lua [7] to avoid the need for recompilation of the source after implementing changes. This helps to quickly test modifications of the code.

The trajectory starts at the current position of the robot. The pathfinding generates waypoints which shall be passed by the robot in successive order. In addition it determines the velocity and rotational orientation which the robot should have, when it reaches the final waypoint. To connect all waypoints with a feasible trajectory, the velocity and its direction at every waypoint must be known.

First, all waypoints are connected via linear intercepts. The time needed to get from one waypoint to another is estimated on the basis of the length of these segments. The direction of the velocity at one waypoint is orthogonal to the angle bisector of the two lines which meet at one waypoint. The absolute value of the velocity is limited by three factors. If the waypoint is close to the previous one, the velocity should be similar to the velocity of the previous waypoint, as the acceleration is limited. If the waypoint is close to the last waypoint, the velocity is limited for the same reasons. Furthermore, we take the angle between the two lines, which meet at the waypoint, into account. If the angle is big, the velocity is close to the maximal velocity. With smaller angles a smaller velocity is chosen.

The robot shall rotate permanently and reach its final rotational orientation and its final position at the same time. This way a rotational orientation can be assigned to each waypoint.

The linear connection between two neighbouring waypoints is divided into two parts with the same length. The acceleration vector is constant inside of each part. The acceleration in one dimension (x, y and rotational) can be calculated independently from the other two accelerations. The values of the accelerations are completely determined by the boundary conditions.
The whole method is visualized in Fig. 9.

Exact Linearization The dynamics of the robot can be described approximately by a linear state space model

$$\underline{\dot{v}}_R(t) = \underline{A}\underline{v}_R(t) + \underline{B}\underline{u}(t) \tag{14}$$

$\underline{v}_R(t)$: velocity vector in the coordinate system of the robot
$\underline{u}(t)$: input vector

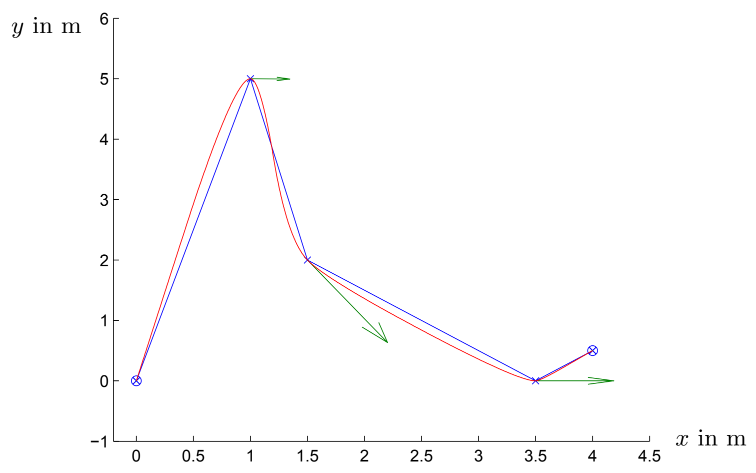


Fig. 9. The waypoints are connected via linear intercepts, a velocity vector is calculated for every waypoint, and all waypoints are connected in a final step using splines.

$\underline{A}$: dynamic matrix
$\underline{B}$: input matrix

in the coordinate system which is attached to the robot.

For the position controller the position of the robot in the global coordinate system is needed. To transform the velocity vector from the coordinate system of the robot into the global coordinate system

$$\underline{v}_R(t) = \underline{T}(\varphi)\underline{v}_G(t) \tag{15}$$

$\underline{v}_G(t)$: velocity vector in the global coordinate system
$\underline{T}(\varphi)$: transformation matrix
φ : angle between robot and global coordinate system

a transformation matrix is needed, which contains sinusoidal terms and is therefore nonlinear

$$\underline{T}(\varphi) = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{16}$$

The non-linearity of the resulting system model in global coordinates

$$\dot{\underline{v}}_G = \underline{T}^{-1}(\varphi)\left[\underline{A}\underline{T}(\varphi) - \dot{\underline{T}}(\varphi)\right]\underline{v}_G + \underline{T}^{-1}(\varphi)\underline{B}\underline{u} \tag{17}$$

can be compensated by a modified input signal

$$\underline{u} = \underline{B}^{-1}\underline{T}(\varphi)\Big[\overline{\underline{u}} - \underline{T}^{-1}(\varphi)\left[\underline{A}\underline{T}(\varphi) - \dot{\underline{T}}(\varphi)\right]\underline{v}_G\Big] . \tag{18}$$

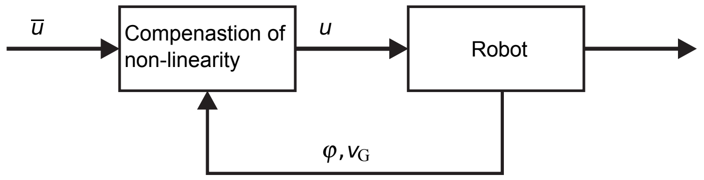


Fig. 10. The linearized system consists of the compensating filter and the dynamics of the robot

The elements of the new input signal $\underline{\overline{u}}$ are the accelerations of the robot in the global coordinate system. These accelerations can be chosen arbitrarily. The input signal that has to be sent to the robot (17) to accelerate in the chosen way can be calculated by (18).

The whole system, consisting of the dynamics of the robot and the compensator of the nonlinearity, can ultimately be treated as a linear system, as can be seen in Fig. 10.

4 Conclusion

Tests have shown that a well-performing tracking system increases the accuracy and reliability of the SSL-Vision detection. The presented algorithm has already proven to be very stable, and therefore reduces the dependency on a perfect vision calibration.

The motion control algorithms achieve a better command response by applying additional feed-foward control rather than common feedback-only. To reduce development time the algorithms have been verified in MATLAB/Simulink.

References

1. Zickler, S., Laue, T., Birbach, O., Wongphati, M., Veloso, M.: Ssl-vision: The shared vision system for the robocup small size league. RoboCup 2009: Robot Soccer World Cup XIII (2009) 425–436
2. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME – Journal of Basic Engineering **82 (Series D)** (1960) 35–45
3. Chaiso1, K., Ariyachartphadungkit, T., Sukvichai, K.: Extended team description 2010 paper team skuba. (2010)
4. Umeno, T., Hori, Y.: Robust speed control of dc servomotors using modern two degrees-of-freedom controller design. IEEE Transactions on Industiral Electronics **38**(5) (1991) 363–368
5. Grotjahn, M., Heimann, B.: Model-based feedforward control in industrial robotics. The International Journal of Robotics Research **21**(45) (2002)
6. Oguntoyinbo, O.: PID Control of Brushless DC Motor and Robot Trajectory Planning and Simulation with MATLAB/Simulink. PhD thesis, Vaasan Ammattikorkeakouluuniversity of Applied Sciences (2009)
7. Ierusalimschy, R.: Programming in Lua. Lua.org (2006)