In: IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems, Baden-Baden, Germany, August 20 - 22, 2001

Safe Human-Robot-Cooperation: Problem Analysis, System Concept and Fast Sensor Fusion

Dirk EBERT and Dominik HENRICH

Embedded Systems and Robotics Lab. (RESY)
Faculty of Informatics, Building 48
University of Kaiserslautern, D-67653 Kaiserslautern, Germany
E-Mail: [debert | henrich]@informatik.uni-kl.de, http: //resy.informatik.uni-kl.de/

Abstract

We present a system concept allowing humans to work safely in the same environment as a robot manipulator. Several cameras survey the common workspace. A look-up-table-based fusion algorithm is used to back-project directly from the image spaces of the cameras to the manipulator's configuration space. In the look-up-tables both, the camera calibration and the robot geometry are implicitly encoded. For experiments, a conventional 6 axis industrial manipulator is used. The work space is surveyed by four grayscale cameras. Due to the limits of present robot controllers, the computationally expensive parts of the system are executed on a server PC that communicates with the robot controller via Ethernet.

1. Introduction

At present, the workspaces of robots are clearly separated from human workspaces. This is a result of the safety requirements prescribed by guidelines such as [ISO10218]. For future applications, it is necessary that humans and robots can cooperate safely in the same work space.

Safe human-robot-cooperation can be arbitrarily complicated. Among the different types of robot motions as defined in [Latombe96], the gross motions appear to be most dangerous for a human in the work cell. They affect a large area of the work cell and are executed at a high speed. Therefore, the first goal is to provide a safe gross motion. In this paper, we present a system concept to achieve this goal, since there is only little previous work available.

[Meisel94] shows an approach for observation of the work space of a portal robot with three cameras. A back-projection from image to Cartesian space is used. Also, the problem of phantom obstacles appearing in the Cartesian space is analyzed. However, the concept is used only for detecting obstacles in the work space and does not cope with the problem that the robot is seen in the images.

[Noborio92] presents an approach for coping with the phantom obstacles using color information. The approach assumes that the different obstacles have different colors, but that each obstacle is uniform in color. This approach might work for the extraction of uniformly colored robots from images, but other obstacles - especially humans - are not likely to be colored uniformly.

The possible errors occurring during the back-projection of one complex obstacle from image to Cartesian space is analyzed by [Niem97]. However, only the obstacle enlargement is regarded.

[Adolphs90] shows an approach to transform multiple obstacles from Cartesian space to configuration space using look up tables (OCMEM). In addition, a compression technique for the look-up-tables is presented. However, this approach assumes that the Cartesian coordinates of all obstacles are known.

The rest of the paper is organized as follows: In Section 2, we present the investigated problem. In Section 3, we present the system concept. As an essential part of this system we present in Section 4 a fast approach based upon look-up-tables for fusing images and joint angles. This approach also includes the extraction of the robot image from the observed scene. The implementation and experiments are presented in Section 5.

2. Problem analysis

In this section we present an answer to the following question: If a robot manipulator and a human work in the same work space, how fast can the robot move without being a safety threat for the human?

To answer this question, we assume that the environment of the robot is represented in its configuration space by a grid and that the discretization is computed by the *max-move* approach presented in [Henrich98]. According to this method the discretization resolution $\Delta q = (q_1, ..., \Delta q_D)$ of a D-dimensional configuration space is set with respect to the maximum movement of the robot endeffect at each step the robot moves along this coordinate. The discretization for each joint can be computed by

$$\Delta q_i = 2 \arcsin\left(\frac{m_{\max}}{2\, l_i}\right)$$

where l_i is the distance between the center of joint i to the farthest point the endeffect can reach, and $m_{\max}$ is a pre-set distance the robot may move

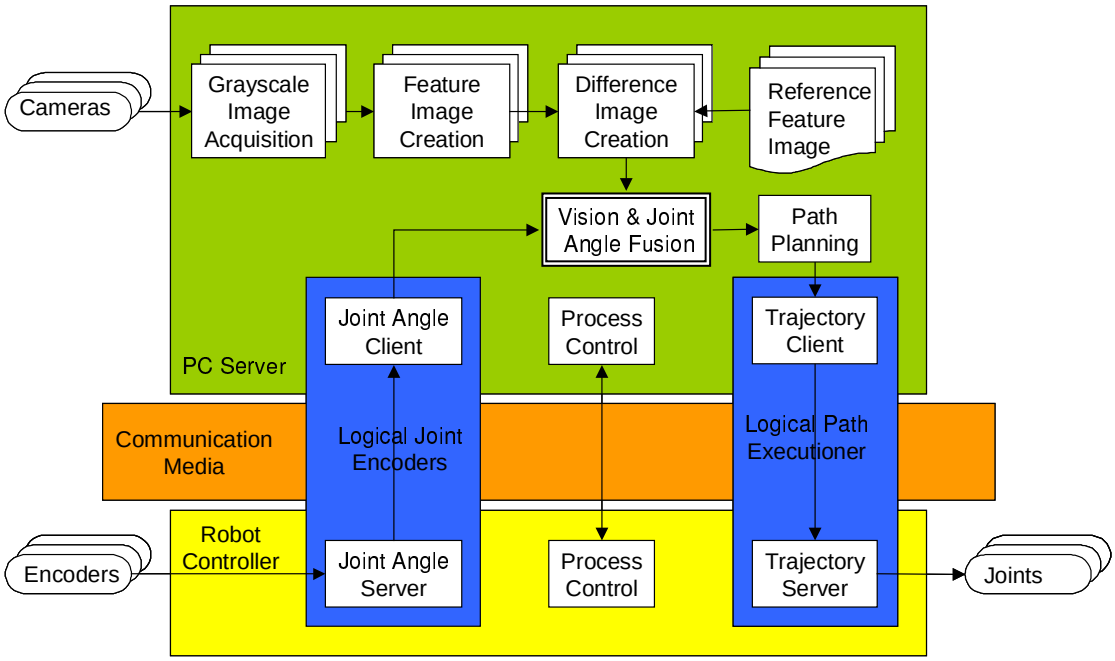

Figure 1: System structure. The system is divided into one part running on the robot controller and one running on the PC server.

in one step in Cartesian space.

Additionally, the robot is assumed to be equipped with sensors that periodically acquire data. By processing the sensor data, each configuration space cell is marked either blocked or free. It is further assumed that the human does not actively bump into the robot. This means that if the robot is not moving, no collision will occur. The distance to obstacles is measured in the number of configuration space grids.

The number of cells in the motion direction of a joint that are checked for obstacles is called the *look-ahead*, and can be different for each joint. As each access to a configuration grid consumes time, it is useful to restrict the number of cells that have to be checked for obstacles to the minimum. However, as the robot can not stop immediately, the look-ahead must be large enough for the given robot speed.

So, the initial question can be formulated as: What is the relation between the minimum look-ahead, configuration-space discretization, sensor update rate, robot speed and obstacle speed?

To find the searched relation, we used a time-based approach. For a continuous system, the deceleration time t_d of the robot has to be smaller than the time t_c remaining before a collision takes place. A time discrete system has to ensure that if the robot is not stopped in the current step, there is still enough time to do so in the next one. With t_s being the time between two time steps and t_p being the time it takes to process the sensor data of one time step the following holds true:

$$t_s + t_p \leq t_c - t_d$$

The deceleration time for the robot is, in the worst case, the time that the link with the largest speed/deceleration proportion needs to stop from maximum speed.

The time to collision t_c can be approximated in worst-case by dividing the distance of the robot to the closest obstacle by the sum of the maximum robot and obstacle speeds. The distance can be approximated by the look-ahead.

If there is a large difference in the deceleration times and the speeds of the different robot joints, it is useful to calculate t_c and t_d for each joint. The joint with the smallest difference is the *critical joint*.

In a system running on a single processor it is not efficient if the data processing takes longer than one time step. In these cases, an upper bound of t_s for the processing time t_p can be assumed.

The presented relation can be used both during system development and at run-time. During the system development phase, it can be used to specify the sensor data processing unit requirements. During run time it can be used to compute the maximum safe robot speed under specific conditions.

This approach is quasistatic and does not take the relative direction of the movements of human and robot into account.

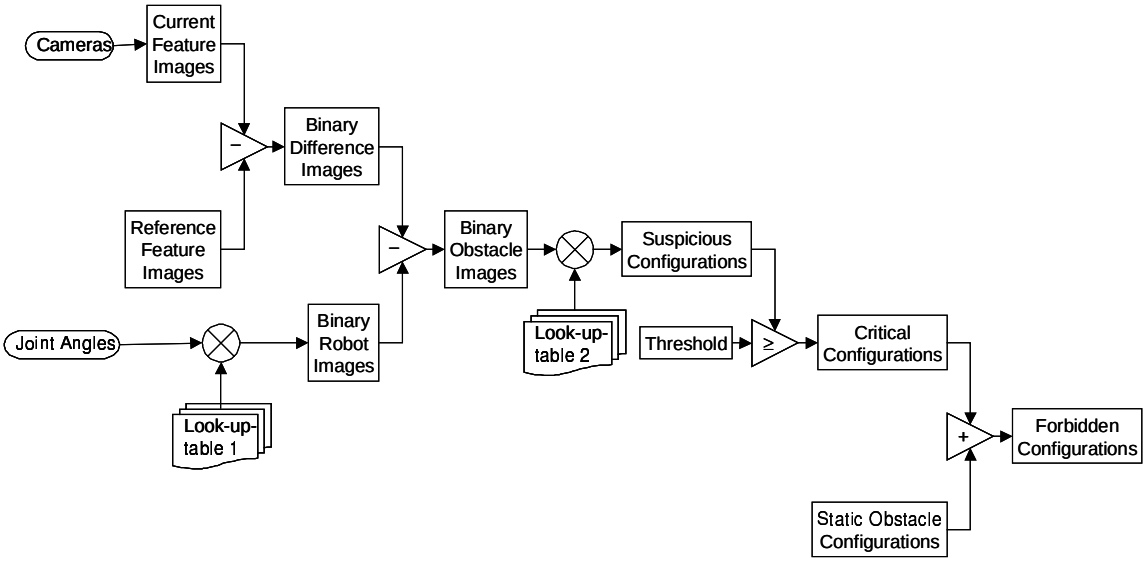

Figure 2: Vision and Joint Angle Fusion

3. System concept

The software provides functions to generate safe gross motions and is divided into two parts, one running on the robot controller and the other running on a server PC. This is done because multi-camera vision and path planning are resource intensive problems and standard robot controllers are not equipped with enough computational power and memory to solve these tasks. Additionally, it is difficult to equip the controller with the necessary hardware to acquire and process the sensor information. PC server and controller communicate over a network or via direct connection. Other robot programs can use the safe gross motion functions as a service. The software structure is presented in Figure 1 and explained in the following.

The images of all cameras are acquired simultaneously. After application of filters for noise reduction the feature images are created. The discretization of the feature images is coarse compared to the original images. This permits the use of several grayscale pixels to produce one feature image pixel. Features can be arbitrarily defined. The only requirement is that a distance metric is defined, which returns a scalar value representing the difference between two feature image pixels. If the difference between the current feature image and the reference image is larger than a set threshold, the corresponding pixel in the difference image is marked as *blocked*. Otherwise the pixel is marked *free*.

If the robot can not be removed from the cell, the reference images can be created artificially by combining several images with the robot in different positions.

The process of fusing several difference images and the joint angles into one world representation is described in detail in Section 4.

A path planner operating on the grid in configuration space is used. Both local and global planners can be integrated into the system, provided they are fast enough.

To meet the high security requirements needed to allow humans to be in the same work space as the robot, several security precautions have to be integrated. For example, if a hardware or software failure is detected, the robot has to be stopped immediately.

4. Vision and Joint Encoder Fusion

The core of the presented approach is a look-up-table-based sensor fusion algorithm (see Figure 2). This algorithm uses a back-projection from image space to configuration space. First, the robot is removed from the difference images using the information stored in look-up-table 1 (LUT-1). The resulting binary obstacle images are fused into a configuration space map of the *suspicious configurations* using look-up-table 2 (LUT-2). If more than a certain number of cameras consider a configuration to be suspicious, the configuration is called a *critical configuration*. In the third step, the configuration space map including all static obstacles, is added, since the static obstacles do not appear in the difference images. The result is the configuration space map of all *forbidden configurations*. In the following, the algorithm is described in more detail.

4.1. Implicit calibration and table generation

The implicit calibration and generation of look-ip-tables is done offline. One set of look-up-tables is required for each different robot geometry. For

example, if the robot has to carry a large beam, the robot with and without the beam in the gripper are considered as two different geometries.

In the brute force approach, the robot is successively positioned in all configurations. When the position is reached, difference images are created for all cameras. In a smarter and less time consuming approach, only key configurations are actually established with the robot, while the other images are created by interpolating either from the key images or from a robot model that is computed from the key images.

For each configuration q, the blocked difference image pixels p of one camera are stored in a table. The layout of LUT-1 for one camera is presented in Figure 3. There is one such look-up-table for each camera.

Configuration	Blocked Pixels
q_1	$p_{1,1}, \ldots, p_{1,N}$
…	…
q_Q	$p_{Q,1}, \ldots, p_{Q,M}$

Figure 3: LUT-1 layout. For each configuration, all blocked pixels in the difference image are stored.

After all configurations have been processed, the contents of LUT-1 is used to create a new table named LUT-2. For each pixel p in the difference image of a specific camera, all configurations q, in which the pixel is blocked, are stored. These configurations are called *suspicious configurations*. The layout of LUT-2 is presented in Figure 4. There is one such look-up-table for each camera.

Pixel	Configuration
p_1	$q_{1,1}, \ldots, q_{1,R}$
…	…
p_P	$q_{P,1}, \ldots, q_{P,S}$

Figure 4: LUT-2 layout. For each pixel all suspicious configurations are stored.

In the look-up-tables LUT-1 and LUT-2 for each camera, the geometry of the robot and the calibration data of the cameras are stored implicitly.

4.2. Balancing resource usage

The look-up process is very fast – however, this speed is achieved by an intense memory usage. Although it is possible today to equip computers with large amounts of memory, for some systems a less memory-intensive approach might be preferable. By storing the look-up-tables in compressed formats, it is possible to reduce the memory requirement, but his results in higher CPU usage to access the information stored in the tables.

As the data encoded in both look-up-tables represents regions, algorithms for region merging can be applied. If one difference image pixel in LUT-1 is blocked, adjacent pixels are likely to be blocked, too. So many pixels $p(u,v)$ can be represented by one region r that is encoded in the table by $r(u_{beg}, v_{beg}, u_{end}, v_{end})$. As this encoding is done off-line, processing intensive algorithms that provide a high compression can be used. To compress LUT-2, the same method can be used. If one configuration is forbidden, neighboring configurations are likely to also be forbidden – so they can be represented by a 6-dimensional box. This method has the advantage that the time critical access to the information does not require much more processing time than the uncompressed tables.

4.3. Image data fusion

To fuse image data, a data structure representing an empty configuration space is created, e.g. all cells contain zeros. Then for all cameras and all blocked pixels of the current difference image, the value of all configuration cells q are increased by 1 if the configuration q is found in the according line of LUT-2 of the current camera.

After the *binary obstacle images* of all cameras are processed, the contents of all configuration space cells are compared with a threshold. If the value is larger than the threshold, the configuration is called critical. All *critical configurations* are stored for further processing.

4.4. Threshold and phantom obstacles

The threshold can have a range from 1 to the total number of cameras in the system. The threshold determines how many cameras must deliver a blocked difference image pixel before a specific configuration is considered to be critical.

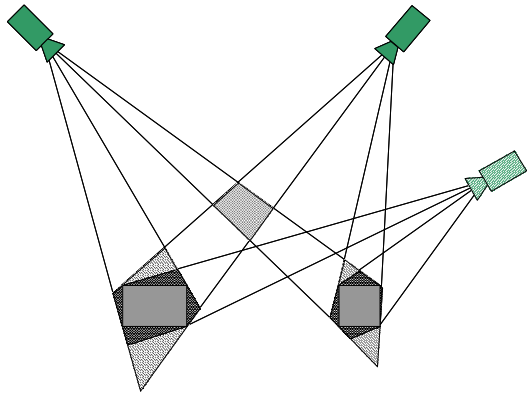

Figure 5: Phantom obstacles and obstacle enlargement. The real obstacles are represented by solid rectangles. If this scene is observed by two cameras, the hatched areas are the enlarged obstacles reconstructed from the camera images and the light cross-hatched area is a phantom obstacle. If the third camera (hatched) is used additionally, the phantom obstacle disappears and the obstacle enlargement is represented by the dark cross-hatched areas. (Image is adapted from [Meisel94])

Because the back-projection does not resolve the correspondence between the blocked pixels of images from different cameras, phantom obstacles can appear. For the back-projection to Cartesian space, this is shown in Figure 5. The phantom obstacles might prevent successful path planning by blocking all possible paths. However, if the threshold is set too high, real obstacles might not be detected. Phantom obstacles can appear if the number of convex objects is equal to or larger than the number of cameras. Additionally, the more cameras are used, the more accurate the obstacle reconstruction will be.

4.5. Sensor data synchronization

The data delivered by the cameras and the joint angle encoders is not necessarily sampled at the same time. Due to the different clocks in robot controller and server PC and to the communication latency, a synchronization is very difficult. However, to remove the robot from the image, the joint angles at the time the camera images were acquired must be is known.

This synchronization is performed in two steps. First, on the server side, a logical sensor is created. This logical sensor communicates with a task on the controller that delivers the joint angles. When the results arrive at the logical sensor, the data gets a timestamp that takes the average communication latency into account. The data is then stored in a smart buffer.

The smart buffer generates artificial joint values by inter- or extrapolating from the data available in the buffer, if joint data is requested with a time stamp not present in the buffer.

5. Experimental results

The main components of our prototype system are a Stäubli RX130 robot manipulator, an Adept CS7 robot controller, a standard PC as server and sensors. As sensors, we use four stationary grayscale CCD cameras positioned in the four top corners of the work cell. The robot controller and the server PC are connected via Ethernet. The frame grabbers for the CCD cameras reside in the server PC.

The images of all four cameras are acquired simultaneously. The noise is reduced by cutting of the least-significant bits of all pixels. The number of bits, that are cut off, depends on the average noise for a pixel value. For this process, the internal look-up-table of the frame grabber is used.

After applying the noise filter, the feature images are created. The discretization of the feature images is 64×64 pixel. One feature image pixel is computed using the data of 9×11 grayscale image pixels. The features used are average, variance, span, contrast and an edge count.

The span is the difference between the brightest and the darkest pixel, while the contrast is the span divided by the sum of the brightest and the darkest pixel. The edge count is the sum of all grayscale pixels after an Laplaceian of Gaussian (LoG) filter has been applied. All features are normalized to return integer values ranging from 0 to 255. In Figure 6 the different features are visualized as grayscale images.

Figure 6: Feature Image Creation. The smaller images visualize the different features of the feature image computed from the larger grayscale image. For visualization, the histogram of each feature image has been normalized to a median of 128. This causes the feature images to appear more similar than they are in reality. From left to right and from top to bottom, the features are average, variance, span, contrast and edge count.

As the robot can not be removed from the cell,

Figure 7: Reference Image Creation. The four smaller images are acquired by the camera. Together with five other pictures showing the robot in different configurations, the reference grayscale image is created. The result using the median of all nine images is shown in the larger image.

the reference images are created by combining several images with the robot in different configurations. The *reference grayscale images* are created by taking the median of the corresponding pixels of all images. These are then used to computed *reference feature images*. In our system we use 9 different robot configurations. The configurations are selected such, that in each camera image and for all configurations the robot covers as few common pixels as possible. In Figure 7, four camera images and the computed reference grayscale image of one camera is shown.

For our robot, a maximum robot movement m_{max} = 18cm results in a configuration space with 181,400 cells, assuming that the robot covers an average of 64 pixels in the difference image and that one pixel can be identified by two bytes. LUT-1 is about 22MB for each camera if the configuration space contains 181,400 cells. For LUT-2 the required memory is about the same, resulting in a total memory need of about 50MB per installed camera. This memory usage can be reduced by using compression techniques.

6. Conclusions

We presented an analysis and a system concept designed to allow a human to work safely in the same workspace as a 6 axis industrial manipulator. The system is equipped with several grayscale cameras and uses a look-up-table-based approach to back-project the difference images directly into configuration space. As the correspondence between pixels is not resolved, the reconstruction of objects from silhouettes has two major difficulties: Obstacle enlargement and phantom obstacles. The effect on work-spaces with many complex obstacles remains to be investigated.

The approach will be tested in various scenarios to determine its robustness and limits. Efficient algorithms for reducing the time to create the look-up-tables as well as for compressing them have to be investigated. If the approach is successful for gross motions, an extension to other motion types can be investigated.

References

[Adolphs90] Adolphs P; Nafziger D: „A Method for Fast Computation of Collision-Free Robot Movements in Configuration-Space“, IEEE Int. Workshop on Intelligent Robots and Systems, July 1990

[Henrich98] Henrich D; Wurll Ch; Wörn H; “Online path planning with optimal C-space discretisation”, In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS’98), Victoria, Canada, Oct. 12-16, 1998

[ISO10218] ISO 10218, EN 775:”Manipulating industrial robots – Safety”, (ISO 10218 modified),1992

[Latombe96] Latombe J-C :”Robot motion planning”; 4. print.; Boston : Kluwer Acad. Publ., 1996.

[Meisel94] Meisel A: “3D-Bildverarbeitung für feste und bewegte Kameras”, Vieweg Verlag, Reihe Forschritte der Robotik Nr. 21, 1994

[Niem97] Niem W: "Error Analysis for Silhouette-Based 3D Shape Estimation from Multiple Views", Proc. on Int. Workshop on Synthetic - Natural Hybrid Coding and Three Dimensional Imaging, Rhodos, September 1997