



Developing a Motion-Stereo Parking Assistant at BMW

By Dr. Eric Wahl, BMW Group, and Dr. Rolf-Dieter Therburg, THECON

For many drivers, finding and then navigating a vehicle into a suitable parking spot is a challenge—mainly because the driver cannot see around the vehicle or determine the size and shape of the parking space.

Several automotive manufacturers, including BMW, now equip vehicles with side-view cameras, which drivers can use to examine blind areas around the car. These cameras provide only a visual representation (a two-dimensional video image) of the area, and are not well-suited to evaluating the size of a potential parking spot or warning the driver of hazards in the car's path. These more complex tasks require a spatial representation. Some companies are working on ultrasonic- and LIDAR-based driver-assistance systems, but such systems impose the cost of additional sensors, and typically have relatively low range and resolution, not to mention low classification abilities.

At BMW, we have developed a motion-stereo system that constructs a 3-D model of the area around the car from 2-D images generated by a single side-view camera (Figure 1). Because it uses the generic 2-D camera already installed on BMW vehicles, the motion-stereo approach offers a cost-effective alternative to solutions that require specialized sensors.

Using MATLAB®, Image Processing Toolbox™, and THECON's MATLAB based MOBIL Image Acquisition and Processing toolbox, engineers from BMW and THECON developed a real-time prototype of the system, which combines images taken from different car positions with vehicle speed, wheel position, and angles data to determine the shape and dimensions of the parking space. In addition to parking assistance, the 3-D

Products Used

- MATLAB®
- Image Processing Toolbox™
- MATLAB Compiler™

representation produced by the system can be used for a variety of other applications, including collision avoidance and automated warnings of curbs and obstacles.

Processing a Series of Images

The first iteration of the system used images collected from the side-view camera while the car was moving past parking spaces. In this phase, the orientation and position of

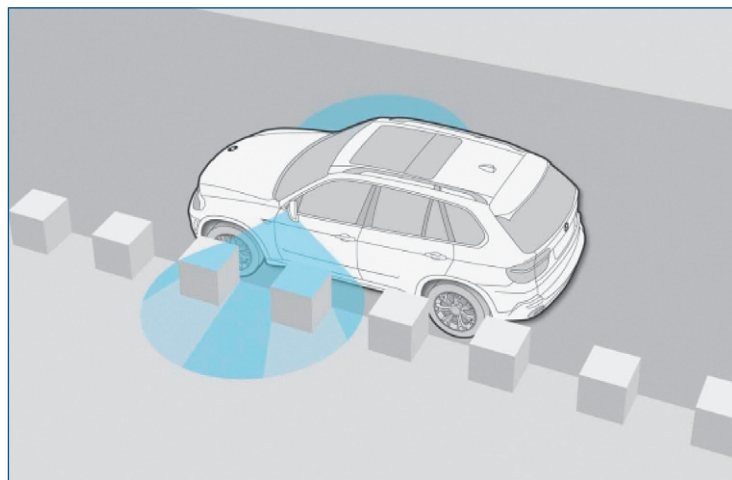


Figure 1. Vehicle equipped with side-view camera. As the vehicle moves, the side-view camera acquires images that are used to measure depth.

the camera were predetermined; development focused on filtering the images and constructing a 3-D representation. We used MATLAB eigenvalue and Hessian matrix operations to create stable edge-detection algorithms (Figure 2).

Image Processing Toolbox functions and the visualization capabilities of MATLAB were particularly helpful as we debugged the edge-detection algorithms. For example, after filtering an image or performing a matrix operation, we could visually compare the original with the new image to validate the results. This kind of debugging is very time-consuming in C or other low-level languages because it requires custom visualization code for each step. Developing image processing applications typically requires many matrix calculations and filtering steps. We have found that coding these applications in C or C++ takes three to four times longer than coding them in MATLAB.

After developing the object reconstruction algorithms in non-real-time, we verified that we could perform the necessary processing as the images were acquired at



Figure 2. Identifying significant edges in an image using edge-detection algorithms.

the camera's standard rate of 50 frames per second. To acquire and process images, we used the MOBIL toolbox. The toolbox combines MATLAB with the real-time image acquisition of a frame-grabber. Operation is carried out with a GUI into which camera images are displayed as stills or as live video. A MEX DLL places function calls for

initializing the frame-grabber, changing camera parameters, acquiring and displaying live images, and transferring images into MATLAB arrays.

With the MOBIL toolbox, real-time image acquisition operates autonomously in the background of the application program with a separate task in kernel mode. Acquired images are stored in a ring buffer structure and signaled to the MATLAB application for further processing. Basic filtering functions in the MOBIL toolbox operate directly in the acquisition buffer. These preprocessed arrays are transferred to MATLAB and are then handled with Image Processing Toolbox.

Integrating CAN Bus Data

To construct a 3-D model from two consecutive camera images, we need to determine how the position and orientation of the camera have changed from image to image. We calculate these changes using data on the vehicle's speed and direction, acquired

What is Motion Stereo?

To create 3-D images of the environment, robotics researchers often use multiple cameras that capture at least two images of the same scene taken at the same instant but from different positions or angles. This procedure is called stereo. Motion stereo varies this concept by exchanging a multiple camera with a single camera that is moved to determine different positions. The motion-stereo system developed at BMW uses the same approach, using two images taken by a single camera and relying on the motion of the automobile to change the point of view from image to image.

While the relative position of the twin cameras in a stereo camera is fixed and well-known, in a motion-stereo system the relative positions change with the speed and direction of the automobile. As a result, our system must gather velocity and wheel position and angles data from the car's CAN bus and then apply sophisticated object-reconstruction algorithms to transform the 2-D point clouds from sequential images into a 3-D model.

via the CAN bus. Our CAN bus interface module, developed in MATLAB, acquires vehicle speed and wheel angle data every 100 milliseconds. The CAN bus interface communicates autonomously with the car to send live signals and receive and decode messages for navigation purposes.

In practice, a car may be traveling at speeds of up to 30 kilometers per hour as the driver looks for a parking space. At these speeds, we must process every image from the camera because we need all the information available. Camera data, however, arrives at a faster rate than CAN bus data, and the two data sources are not synchronized. To determine camera position for each image, we used a Kalman filter in

MATLAB to denoise the CAN bus data and then calculated the vehicle's route by interpolating the available data points into a smooth curve.

Test-Driving the Prototype

To test the system in real-world conditions, we assembled a prototype system and conducted test drives in a BMW 5 Series Touring. We used MATLAB Compiler™ to build a standalone executable of the motion-stereo application, which we deployed to a ruggedized mobile PC with 12 VDC power supply equipped with a 2.4 GHz Intel Core2Duo 4 processor, 2 GB RAM, an asynchronous 4-channel image acquisition board, and a CAN bus hardware interface.

A screenshot of the application is shown in Figure 3.

Initial road tests revealed a weakness in the original algorithms. Periodic background structures such as fences and cobblestones caused the system to misinterpret depth values. We corrected the problem by using MATLAB to create a filter for these kinds of objects.

The system performed well during test drives, easily keeping up with the supply of real-time images and CAN data. When the system is used in production, the application will run on an embedded processor and use the vehicle's on-board LCD display screen.

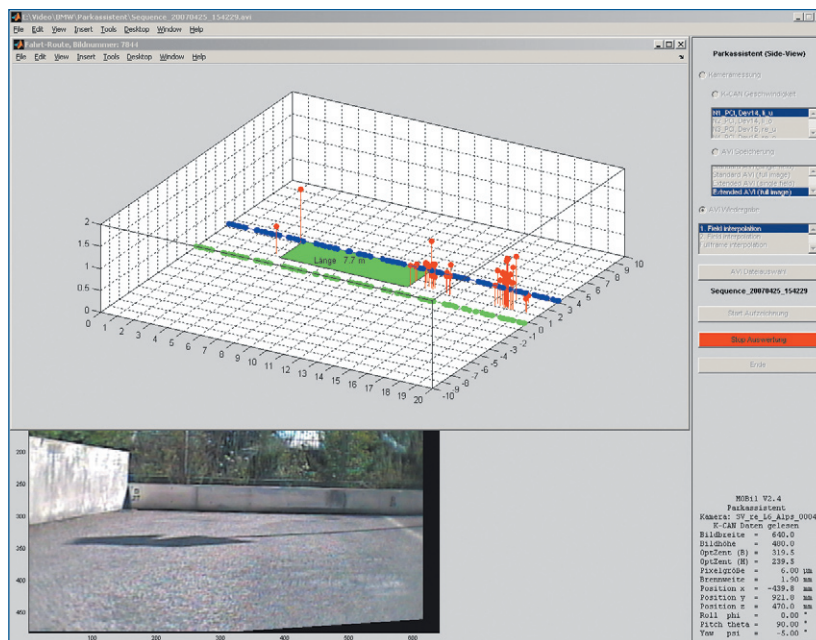


Figure 3. The prototype motion-stereo application, used to measure the parking environment. The dotted blue and green lines mark the region examined. The green rectangle represents the parking space detected by the system. Red dots show potential obstacles. (Image taken from a point to the left of the green rectangle, looking right).

Identifying Objects

In developing the prototype, we were careful to separate the object construction and object identification functions. This distinction enables engineers throughout BMW, as well as third-party vendors, to design applications that use the 3-D data for different purposes in different vehicles. Control engineers, for example, can use the data to automatically park a vehicle. Since many BMW control engineers also use MATLAB, there is an opportunity to simulate a complete system that combines their control models with our MATLAB application.

Work is currently under way to improve object construction by increasing the density of point clouds. We are also working on improved object identification, which will enable the system to classify street light poles, curbs, and other objects—a task that is all but impossible with ultrasonic systems. In further experiments we will be evaluating the use of a top-view camera, which provides a wider-viewing angle but lower resolution than the side-view camera.

Using the movement of the car to assemble 3-D data from a single camera is an example of BMW innovation in the automotive industry. Costly sensors are useful for research but impractical for real applications. By using equipment already on our vehicles, we can provide timely alerts and information to our customers, adding value without increasing manufacturing costs. ■

For More Information

- **BMW**
www.bmw.com
- **Webinar: Image Processing Using MATLAB**
www.mathworks.com/wbnr30823

Resources

VISIT

www.mathworks.com

TECHNICAL SUPPORT

www.mathworks.com/support

ONLINE USER COMMUNITY

www.mathworks.com/matlabcentral

DEMOS

www.mathworks.com/demos

TRAINING SERVICES

www.mathworks.com/training

THIRD-PARTY PRODUCTS AND SERVICES

www.mathworks.com/connections

Worldwide CONTACTS

www.mathworks.com/contact

E-MAIL

info@mathworks.com

© 2008 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

91594v00 11/08