Development of an Open Source Software Package for Autonomous Robotic Docking Using Stereoscopic Imaging

Kyle Liddell
Dr. Chih-Hao Wu

Electrical Engineering/Computer Science
Arkansas Tech University
Russellville, Arkansas

Abstract

The goal of our project, Development of Machine Vision Algorithms for Cooperation in a Multi-Agent Robotic System, is to create and implement algorithms to enable a robot to find and dock itself with another vehicle system. The intent of this system is to create a framework to allow for robotic missions to be decentralized by separating the various sensors from the rover itself. Those sensors can be placed on an inexpensive chassis without control or movement systems, leaving the actual rover system to be a much simpler device only needing equipment to move itself and a sensor package. Currently a matching pair of IEEE1394 digital video cameras are used on a PC running a combination of open source image processing software and our target tracking and control C code. Our code can currently detect a target in a cluttered scene, and then find the direction to the target. The code to plot the course to the target will be ready shortly, and then will be combined with our motor control code. Once the framework has been implemented and tested on our current hardware, work will then shift to making the software easier to adapt to various robot and camera systems, and, once it is available, allowing our code to work with NASA's CLARAty software. This will allow us to offer a package of free software that can be integrated into any robotic system to provide a ready to use docking and control mechanism.

Robotic systems have become more popular in various applications during recent years . Much of the work involved in implementing a robotic system involves writing software to perform basic tasks that are common to any robotic system. In particular, some mechanism is often needed to allow the robot to dock with a target device. If the positions of the robot and the target are known, a simple dead reckoning method can be used. However, if one of the positions is not known, then the more complex problem of finding the target must be solved. One solution is to use various computer vision techniques. The target must be found in a potentially cluttered image, and then position information must be extracted from the scene. Also, the distance to the target must be determined, which requires a secondary sensor device, such as a second camera to allow the use of stereoscopic imaging techniques.

Autonomous docking by means of computer vision systems has been implemented successfully in many research projects. However, the projects have shared two negative traits: their implementation only works with their equipment, and (perhaps for that reason) the software is not released to the public. This requires anyone wanting their robot to dock with a target to "reinvent the wheel" and write their own implemetation of an autonomous docking algorithm. The goal of our project is to produce a software package without those two problems. Instead of having the target information hard-wired into the software, the package will have a mechanism allowing the user to provide the needed characteristics of their target. Also, a well defined modular interface will be used to allow new hardware "drivers" to be written by the user. This will allow the user to make use of whatever robotic hardware is the best fit for their application. Finally, the code will be freely available, which will allow anyone to skip all the housekeeping work needed for their robotic system. The software will be released as free software licensed under the GNU General Public License, which allows anyone to use, modify, or copy the software for the own use.

Background

Our project is an extension of an earlier project to develop a software package for robotic autonomous docking using stereoscopic vision. In our implementation, two cameras were mounted on the robot platform to provide both position and range information. The position information is obtained by filtering the scene provided by a camera. Based on the visual appearance of the target, the scenes are searched to find the target. They are then analyzed further to determine the location of the target relative to the platform. The range to the target is found by performing the same filtering process on an image simultaneously taken from the other camera. Given some information about the position of the cameras, the distance to the target can be calculated from the differences in the apparent target position in each image.

Once the distance and direction of the target is known, the information is passed on to the navigation code. This code determines the best way to move the robot platform to the target. The navigation code then calls the motor control code to translate the movements into the appropriate commands for each motor on the robot.

Hardware

Our current robot platform consists of a custom built metal frame with the wheels in a tricycle layout. The platform is driven by two Dunkermotoren 12V motors with high resolution encoders. A pair of JR-Kerr single board servo controllers with RS-485 interfaces are used to drive the motors. The front center wheel is a freely rotating caster wheel. Our development computer system is mounted on the platform in a generic PC case. The cameras are mounted in a simple aluminum bracket. The cameras are a pair of Point Grey Research Dragonfly digital video cameras. Each camera provides 640x480 resolution, 16 bit color images at a rate of 30 frames per second. The IEEE1394 interface is used to transmit the images to the computer. The computer system is an inexpensive x86 architecture PC. An AMD Athlon64 3500+ CPU is mounted in a basic micro ATX motherboard. Currently, the system has 512MB DDR SDRAM. Since the motherboard only includes one IEEE1394 port, a separate controller card has been added so that both cameras can have their own IEEE1394 bus. A PCI RS-422/485 serial port card was also added to allow communication with the servo controllers. A small hard drive is attached to the system for development purposes, but this will be removed once the software has been written in favor of a smaller storage system.

Software

The software package is a set of programs and libraries written in C. They are currently written for the Gentoo Linux distribution, but the most popular distributions will be supported in the release version of the software. On top of the Linux OS, the libdc1394 software package is used to provide a simple interface to the IEEE1394 devices. Since the cameras transmit images as Bayer encoded grayscale images, the library also performs the Bayer conversion needed to interpret the data as a color image. The Gandalf image processing library is also used to perform some of basic image processing tasks.

The package consists of three components: the image processing programs, the navigation programs, and the motor control programs. The most complex of the three is the image processing component. This program takes an image and attempts to detect the target inside the scene. Once the target is located, that image will be compared with an image from the other camera and the distance will be calculated. Currently, the target is found by searching the image for a given color and shape. The color is chosen to be a unique color in the environment, and the shape can be selected to give the most information about positioning. In the current testing implementation, the target is a green square. The software processes a frame from the camera to find the average "greenness" of each pixel in the scene by iterating over all the pixels in the image and storing the green value. The average is calculated as a simple arithmetic mean by dividing the accumulated value by the number of pixels in the image. Then, the image is iterated over again to find those pixels which have a higher green value than the average. A matching tolerance can be specified if the color is not completely unique in the environment, so that the pixel must be more green than the average by a given amount. The addresses of those green pixels are set in a simple binary image. This target detection algorithm is very simple and

inefficient and will be optimized once all functionality is present in the package. The resulting image is filtered to remove any noise from the scene, to end up with an image containing only the target. The angle to the target can then be computed easily by calculating the distance from the center of the camera image to the center of the target shape. The aspect angle to the target is calculated by measuring the type and amount of distortion of the target. With the square target, if there is a large aspect angle, the near edge of the square will appear longer than the far edge. This information can be used to determine if the robot will be able to successfully dock with the target, or if the robot use some path besides a direct one to the target. Then the distance is calculated by performing the same color filtering on an image from the other camera that has been taken at the same time. The stereovision calculations are used to determine the range to the target. This information is combined to give the location of the target. This information can then be passed on to the next layer.

The navigation layer will take the position of the target to determine what path the robot should take to reach the target. Currently, a very simple algorithm has been developed and will be tested first. This algorithm is presented below in a pseudo-code:

```
while (target is too far away) {
        if  (aspect angle within limits) {
                if (target is not centered) {
                        turn robot towards target;
                } else {
                        move robot forward;
                }
        } else {                                //aspect angle too high
                move robot backward;
        }
}

dock with target;                               //target is not too far away
```

This algorithm is not optimal in situations where the target's aspect angle is too high. A possible modification to this algorithm would be to turn at a right angle towards the target and then travel until the robot should be directly in front of the target, based on the range and aspect information provided by the vision system, and then turn back in the direction of the target, which now should have an aspect angle close to zero.

This algorithm requires a prerequisite of having a general location of the target. While this would be the situation for some applications, a variety of search algorithms will be implemented to allow the robot to find the target completely autonomously. Some possible algorithms would be searching by making expanding circles or squares. Also, a secondary "beacon" type sensor system could be implemented by users of the package, and could be made to interface with the navigation system when a reliable image of the target is not available.

Another improvement to the navigation system would be a form of pathfinding around obstacles. If the vision system were expanded to also detect obstacles, then this information could also be provided to the navigation system which could then find the optimum path around the obstacles. Again, extra sensors could also be added to the system which could provide obstacle information to the navigation system.

The motor control subsystem has the straightforward task of translating its input of the desired direction, velocity, and distance into commands to the servo controller. Given the information about the movement characteristics of the robot, the motors can be effectively controlled. A possible improvement would be to have different operating modes, such as "fullspeed" or "powersave". The most time consuming task in creating this component will be translating simple commands such as right motor full power into the proper command protocol for the motors, as well as for the RS-485 communications channel. A software library called libnmc has been found which claims to perform these tasks. Testing will be performed on this package, and if it proves sufficient, a significant amount of time should be saved. Otherwise, the motor control code will be written and released along with the rest of the project.

All these components will be somewhat modular in that each component can be reused along with other projects. For testing purposes, all the components will be compiled into a single executable for the sake of simplicity. However, the release version of the software will have several options for communication. Some possibilities to be considered are UNIX pipes, simple text files stored in the file system, or a client-server approach. The last approach would possibly be the most extensible method. However, a major concern will be the performance of the code, which may make a client-server approach less attractive.

Indeed, performance is a major requirement for any code written for the project. It should be possible for the system to process images far more rapidly than is needed for any reasonable movement speed. However, the implementation should be made as fast as possible in order for the system to still produce reasonable results on very low performance systems. This is partly to allow the system to be used on old hardware, but also to allow the system to be executed on very low power consumption hardware platforms. While the current setup will require a power cord to be attached to the system, testing will be performed with the finished software to find the minimum requrements of the system to hopefully allow the entire hardware for the robot to be powered by batteries.

The distinguishing characteristics of this system are that it will be modular and free. While the project will be released as a ready to use package of software to be installed on the control computer, each component can be used in any other software project. For example, if one has a more sophisticated navigation subsystem and only requires target position information, the vision subsystem can be used separately from the other components. Also, it will be easy to add other subsystems to the package (for example, obstacle detection), or to extend any package (for example, adding a rear facing camera to make searching easier). Well documented, simple interfaces will be used to make the subsystems easily usable with other software systems. The

code will also be heavily documented to allow developers to understand the code more quickly so that any changes or improvements can be written quickly.

Also, this package will be released under the GPL free software license. The GPL license allows for anyone to modify the code, or copy the code to their own projects. The only requirement of the GPL is that the source code for the project remain freely available, and all derived software based on the GPL software also be licensed under the GPL. If users do not make public releases of their code, they are not required to release their source code. If their code is already under the GPL, they can freely use any code or ideas from this project with their project. However, this does not mean that this code can only be used on free software projects: if the software components are used as standalone binaries and communicated with using the file or client-server communication systems, the code that uses those applications can be written under any terms that the authors desire. This is what makes this package different from other robot control software packages, because most other projects are merely proof of concept type code. This project will be usable by anyone in the robotics community, allowing researchers to concentrate on their topics of interest, instead of spending programmer time on robot housekeeping.

This code will be released on the Internet once the basic features are implemented. It will be available to run on Linux systems, and may even include complete OS install packages. Also, a list of tested and recommended hardware will be compiled and posted. Once all of the basic functions have been written, the remainder of the time will be spent on testing many different types of imaging hardware, computing hardware, and motor hardware. Most robotics projects are intended to work with special robot control kits, but if instead our project works with the most common commodity robotics hardware, it should reduce the cost needed to get a robotic system in operation, and also work with a wider variety of applications. It is hoped that a few embedded computer systems will be acquired in order to test the project software on some very minimal, very low power systems.

It is expected that by the end of 2006, a robotic system using our current hardware should be able to successfully dock with a target that is in a known starting location. The code will then be revised and fully documented, and general documentation will be written. The project files will then be posted to the Internet, and work will then shift to providing support for multiple hardware types.

The website for the package has not yet been constructed. However, a placeholder and information page will be available at http://foobox.homelinux.net/robot.html and will be modified to give the address of the package website.