Ceilbot mapping and vision, final report

Fall 2010

Juhana Leiwo

I participated in the AS Department's Ceilbot student project in the fall of 2010. The aim of the Ceilbot project is to design and prototype a domestic helper robot that moves along the ceiling to circumvent the difficulties of navigation and locomotion along the floor in a household environment. The project has been going on for a year and a half. The general construction and purpose of the robot had been defined earlier in the project, and the general task for this term was to start building a working prototype. My task was to design the computer vision and environment mapping system of the robot. Earlier in the project I had already worked on mapping and vision algorithms and software. This time I concentrated on the mechanical and electronic systems required for the mapping and vision system.

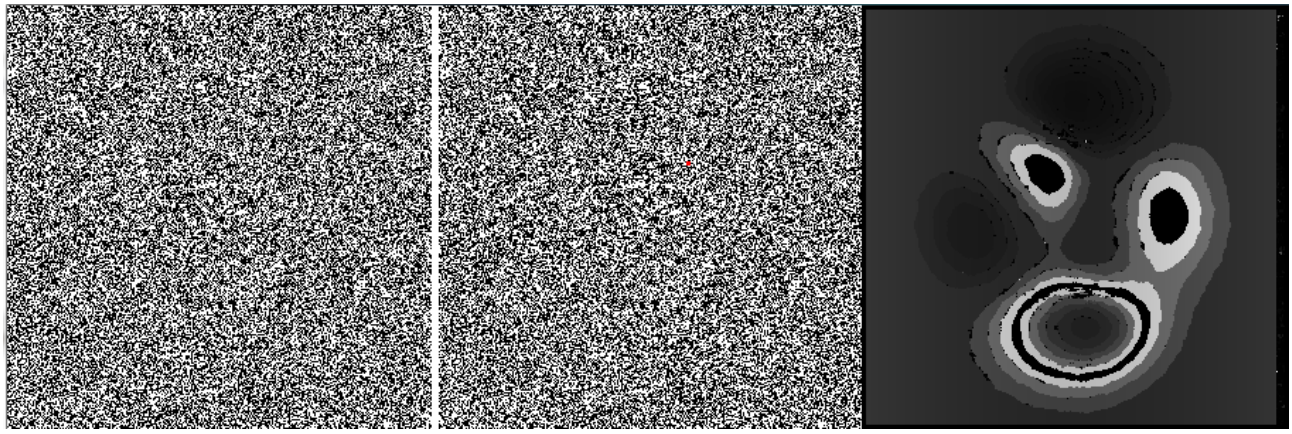
Sensors

The mapping and vision system is responsible for producing a computer-processable numeric representation of the robot's surroundings. The system needs to be able to measure a three-dimensional spatial map of the environment for navigation and obtain images for visual processing. These requirements define the type of sensors used in the system. Visual images can obviously be captured with a camera, but obtaining three-dimensional spatial data is not as straightforward, since no single method has yet established itself as the common standard (although this seems to be about to change). I considered four different ways of 3d imaging for the system. The options were to use a time-of-flight range camera, a scanning laser rangefinder, structured light scanning or stereoscopic vision with two normal cameras. The ToF range camera is definitely the most advanced of the methods. It is a device which records depth information like pixels in an image using, for example, a radio frequency modulated infrared light source, with the depth values interpreted from the phase differences between the transmitted beam and the light reflected to each pixel. What makes this kind of a device especially desirable is that it records the entire scene in its view at once, several frames per second. This would be very useful for a moving robot in a dynamic environment. When evaluating the scanner options at the beginning of the course I found out that existing models of this kind of cameras were too expensive, although some companies had future plans for consumer-priced devices. I decided to concentrate on the remaining options and ended up with a design that combines all three. A line scanning laser rangefinder already available at the lab would be used for measuring distance to the walls and room corners and an assembly of two webcams and a patterned laser pointer would provide stereoscopic vision assisted by structured light. The laser scanner would preferrably not be used for actual 3d scanning, because it only scans in a two-dimensional line and would have to be tilted mechanically to get a three-dimensional image. Unless the tilting mechanism was extremely accurate and the scene and robot were quite stationary during the scan, it would be difficult to align the individual scanlines to form a reliable measurement. The essential part of task-related (as opposed to mapping-related) depth information would therefore be provided by the stereo camera system.

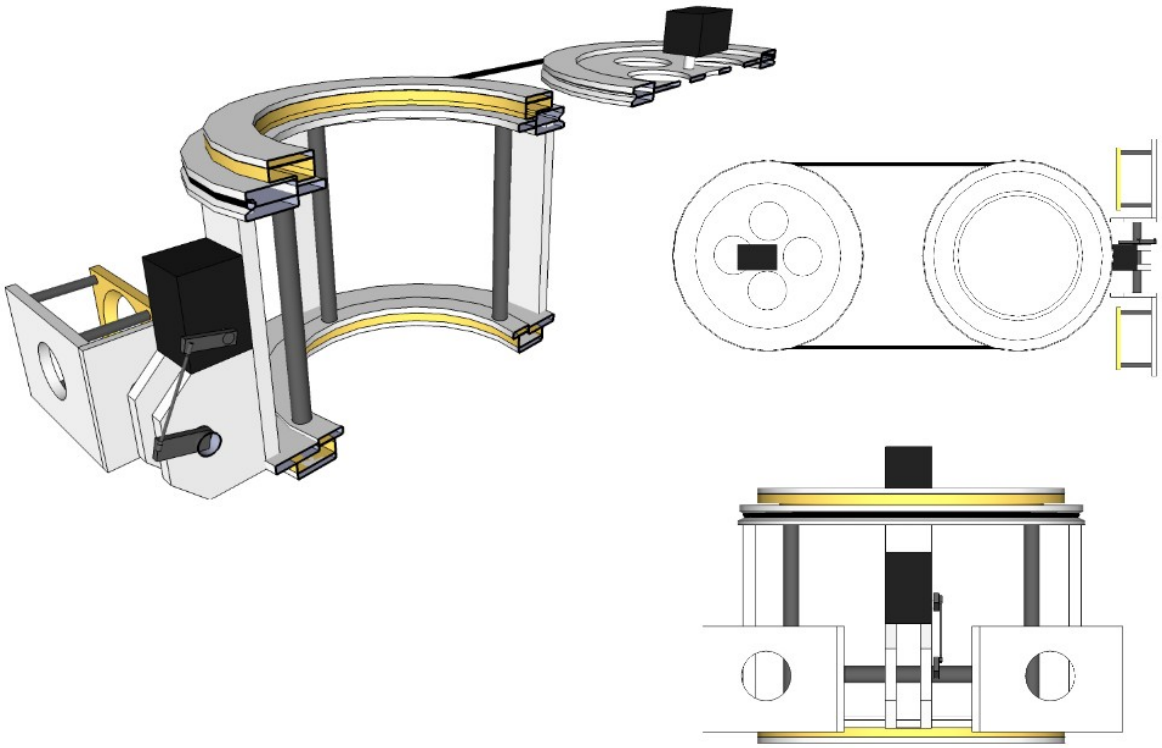
Stereoscopic processing

The performance of a stereoscopic vision system is almost entirely defined by the performance of the algorithm used for solving the corresponding points problem. Creating a good algorithm for this is nontrivial and the subject of ongoing research. I tried out some simple approaches in a Java program to see if I could find a solution that would be good enough, or at least consistently bad. I used stereo images downloaded from the web as test data and a simple algorithm which compares small areas of the images pixel by pixel and looks for the least difference. The results were somewhat confusing. For a couple of the image pairs I used, the program managed to parse the depth information remarkably well, but for most of the images it would only produce a random-looking jumble. The best results were, unsurprisingly, achieved when parsing a mathematically generated random-dot stereogram. One particular photograph got close to this, but the rest gave no real results. The reason seemed to be image rectification. My algorithm only searched

horizontally, assuming the images were taken at the same height and rotation, and if this wasn't the case, the search never passed the correct feature in the target image. Since the algorithm chose the point with the least difference anyway, the result was a collection of random depth values. I didn't want to put too much effort in the stereo processing at this point so I left it at that. The OpenCV computer vision library contains functions for image rectification and depth parsing, so I decided to use those later on. I also decided to install a laser pointer which would project a clearly visible pattern of lines or dots onto the scene. This pattern should be quite easy to extract from the camera images and could be used to calculate a quick and rough preview of the scene.

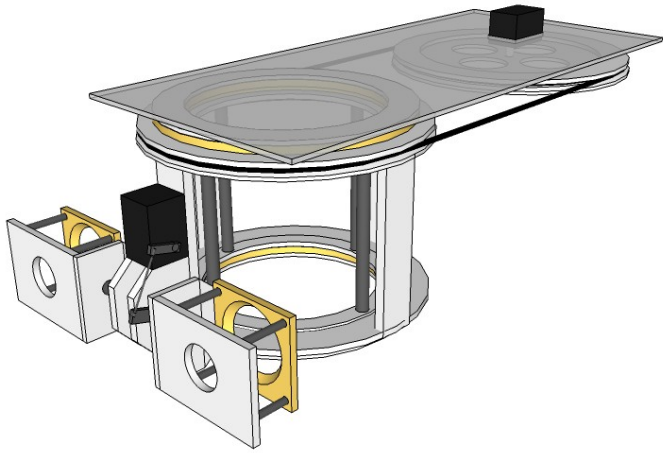


The mounting gimbal



To efficiently monitor the environment, the scanner and cameras need to be mounted on a gimbal which can tilt and rotate. In other words, servos are required for the yaw and pitch angles, and the roll angle can remain fixed with the cameras in the "natural" position and the scanning line positioned horizontally. The scan orientation is horizontal because that way the wall positions can be scanned easily. An important design constraint for the gimbal is that the robot will have a manipulator arm mounted on its underside, same as the cam/scan assembly. Obviously the robot must be able to see in all directions without the manipulator getting on the way. The options in this

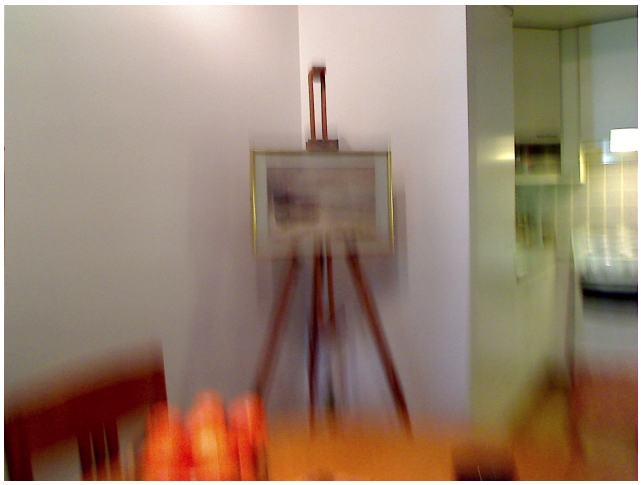
case are either to make the whole robot body rotating or mount the manipulator arm THROUGH the cam/scan assembly, which would then rotate around it. Since I wasn't responsible for the design of the robot body, I chose the latter alternative. The sensor hardware would be mounted on a sort of a round cage, which is supported by slide bearings on an inner cage, which in turn is fixed on the robot body. The manipulator arm would be mounted in the space inside the inner cage. The rotating outer cage would be turned by means of a driving belt, with the turning motor mounted on the robot body next to the cage assembly. The pitch control servo would be mounted on the outer cage with the sensors. There are certain more or less obvious problems with this design. For one, the equipment on the rotating outer cage require electrical cables. This means that the cage can't rotate continuously. Secondly, the assembly does need to rotate at least the whole circle, preferrably a bit more. It follows that the motor turning it must be something other than a standard servo. A stepped motor would suffice, but with that we could lose the current orientation of the assembly, if it happens to turn on its own for some reason or if it sticks badly enough for the motor to get out of step trying to turn it. Of course, knowledge of the orientation of the sensors is important for the mapping to make sense. A position encoder would need to be added to measure the orientation, making the system one big, complicated build-it-yourself servo. Luckily it appears that such big complicated servos already exist, and they aren't even very expensive. A company called Robotis manufactures a range of servos called Dynamixel, especially meant for robotics applications. The smallest servo in their range, the Dynamixel AX-12, is perfectly suitable for our purpose. The Dynamixel servos are capable of either continuously turning operation or normal proportional servo operation. You can choose whether to set the servo position or to set the travel speed, and the current position of the servo can be read at any time. The servo is controlled through a serial bus, for which there exist an usb adapter and software development libraries for linux and windows. The torque of the AX-12 model is 16.5 kgcm at an operating voltage of 10V, which should be enough. I decided to use these kind of servos for both pitch and yaw. I later found out that these servos are in fact less than perfect for our application, because apparently you can't set the center position of the proportional steering freely. The maximum proportional steering angle of 300 degrees is defined about a fixed central point, which means that the remaining 60 degrees must be handled using the continuous travel mode. The specification document of the servo does not tell whether the position angle is readable outside the maximum angle region. It might be possible to get around this problem by up-gearing the transmission from the servo to the gimbal by a 12:10 gear ratio, so that a 300 degree turn of the servo wheel would result in a 360 degree turn of the gimbal. If the servo does not have enough torque to turn the up-geared assembly, an extra position encoder might be needed anyway.

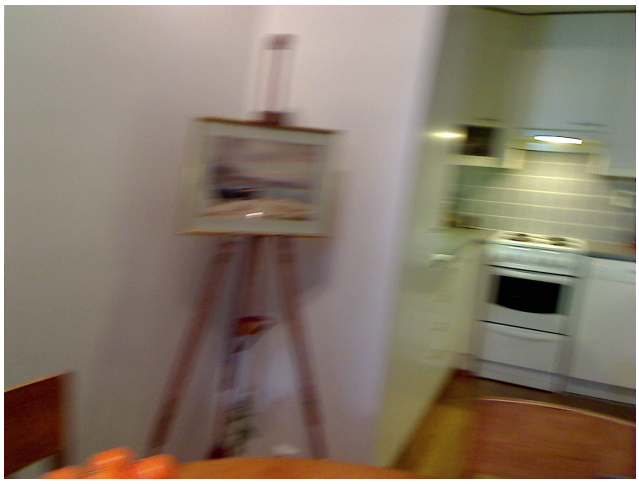


Webcam problems

Two Logitech Webcam C600 cameras were chosen for the vision part. The cameras were advertised as having good optics, resolution and light sensitivity, and also would probably work

with the Linux operating system we had chosen for the robot. Unfortunately I had no chance to test the selected camera model before ordering. When the cameras had arrived and I was testing them, I became aware that they suffered greatly from an effect called "rolling shutter". The rolling shutter effect is caused by the way the camera reads its sensor. The image is read from the sensor pixel by pixel, from left to right, top to bottom. The reading operation is not particularly fast, especially in poor light, so the pixels arrive a significant time apart. If anything in the image moves or if the camera itself is moved, the scene does not stay constant for each pixel. This results in a strangely warped image. Specifically, if the camera is moved sideways, the bottom part of the image seems to bend away from the direction the camera is moving, and moving the camera up or down causes the image to stretch or squash respectively. It appears that nearly all webcams exhibit this effect and it isn't stated in their specifications. Cameras like this might not be entirely unusable for robot vision, but it is definitely a limitation for the robot to have to stop every time it needs to view its surroundings. Extracting stereoscopic data from warped images is rather hopeless and motion-to-structure analysis becomes mostly impossible. The usefulness of motion detection is also undermined. It might be possible to correct the images with algorithms, but this feels like a waste of effort given that properly working cameras do exist.





Microsoft Kinect

When in the beginning of the term I evaluated the different 3d data aquisition methods, it seemed that several companies were developing inexpensive 3d distance cameras, and were about to publish commercial models of them in the near future. There was even anticipation that Microsoft would release a consumer-priced game controller for the Xbox based on this technology. At approximately the same time that I found out our webcams were suffering of the rolling shutters, Microsoft indeed released their Kinect game controller cum 3d camera. Almost immediately an open source library for exploiting this device's capabilities (both normal and 3d camera) was also published (not, obviously, by Microsoft). Upon these news, there was a general consensus among the ceilbot project's crew to replace the cumbersome line scanner and double webcam assembly by a Kinect sensor, costing less than one tenth of the price of the line scanner alone. The sensor has now been ordered and the design of the gimbal needs to be altered accordingly.


XBOX 360