Principles of Computer System Design for Stereo Perception

Michael Wagner¹, David O’Hallaron², Dimi Apostolopoulos¹, Chris Urmson¹
Email: mwagner@cmu.edu
Phone: 412-268-5354

CMU-RI-TR-02-01
January 17, 2002

Abstract

This paper presents principles for designing a computer system that supports stereo perception for an autonomous, mobile robot. These principles aid the engineer in selecting a CPU and network bus to maintain an acceptable level of robotic capability without needlessly large performance margins, which can result in excessive system power consumption. A series of configuration equations are presented that describe relationships between important system parameters such as CPU clock speed, network communication latency, computer power consumption, robot response distance, robot velocity and stereo camera parameters. Configuration equations are intended to improve the engineer’s ability to make trade-off decisions early in the design process. An application of these principles on an existing autonomous mobile robot, Hyperion, are presented both as an example and to outline future work.

1. Introduction

A critical component of an autonomous robot is its capability to compute. Computation is required to perform perception, motor control and planning, among numerous other tasks. Unfortunately, specification and design of computer systems for autonomous mobile robots is often an ad hoc process, consisting of rough estimates of performance requirements, capability margins orders of magnitude above those necessary and a severe lack of well-analyzed prior work on which to base design decisions. As a result, computer engineers developing autonomous robots cannot make educated trade-offs regarding computing capability and computer system power consumption. At best, the resulting system wastes electrical power providing excess compute cycles; at worst, computing resources become a bottleneck of robotic system performance.

In particular, principled methods of assessing the computational requirements of stereo perception are needed. Being a widespread sensing modality, stereo perception is important to many autonomous robots. Traditionally, stereo perception has consumed a significant portion of a robot’s computing resources. In the past, many researchers sought to decrease the latencies involved in stereo perception by constructing networks of parallel processors. In the late 1990’s general-purpose, single-CPU computers acquired sufficient speed and parallel data paths to perform stereo perception calculations at rates above 10 Hz. This computational speed comes often at the cost of drastically increased electrical power consumption. On some robots, the computing power requirements even approach locomotion system requirements [Shamah01]. Sufficient data is often unavailable to evaluate design trade-offs between computational resources and power consumption.

This paper investigates principled methods for designing a computing system capable of performing stereo perception to meet task requirements. One important robotic task requirement is response distance [Kelly98], which is based on the robot’s nominal operating velocity and the total latency involved in sensing the world with stereo vision. Another important task requirement is fidelity of terrain models generated by stereo perception. Inaccurate terrain models prevent a robot from avoiding dangerous obstacles. We seek to understand the relationship between providing guarantees of a given desired response distance, terrain model fidelity and steady-state electrical power consumption. To do so, the effects of computer system configuration on task requirements will be determined through testing performed on components of the Hyperion robot [Wettergreen02].

2. Relevant Stereo Perception Background

Stereo perception provides the distance from a camera to an object for pixels that overlap in two images from a pair of cameras. Finding depth involves calculating disparity, or distance in an image that an object “moves” when viewed...
from each camera. An object’s disparity is inversely proportional to its distance from the camera. The collection of disparities for all pixels shared in two images of a stereo pair is called a disparity map.

Here we analyze the performance of software using one particular stereo library, the Small Vision System (SVS) written by Kurt Konolige at Stanford Research International [Konolige97]. The SVS library provides functionality to C/C++ programs to allow dewarping of stereo image pairs, calculation of stereo disparity maps at $1/8$ sub-pixel resolution and transformation of this map to a cloud of 3D points. The library is used in the navigation system of the Hyperion robot to perform obstacle avoidance.

### 3. Approach

The approach of this paper is based on the determination of various configuration equations, which express relationships based on configuration parameters, performance parameters and mission parameters, an approach based on that described in [Apostolopoulos00]. Examples of each type of parameter in the scope of this work are summarized in Table 1 and discussed in detail in the following sections.

<table>
<thead>
<tr>
<th>Configuration Parameters</th>
<th>Performance Parameters</th>
<th>Mission Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU clock speed</td>
<td>Response distance</td>
<td>Robot velocity</td>
</tr>
<tr>
<td>Number of networked CPUs</td>
<td>Terrain model fidelity</td>
<td>Minimum obstacle height</td>
</tr>
<tr>
<td>Network communication latency</td>
<td>Computing power consumption</td>
<td></td>
</tr>
<tr>
<td>Robot width</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radius of smallest traversable steering arc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distance along the y-axis of the stereo camera to the front of the robot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stereo camera height</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stereo image resolution</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Parameters on which configuration equations are based

When engineering a robot’s computer system, the designer’s responsibility is to optimize multiple performance parameters by selecting proper configuration parameters. Mission parameters are generally known to the designer a priori and are usually determined through preliminary analysis of task requirements or terrain characteristics. Configuration parameters are chosen to be relevant early in the design process. Examples of inappropriate configuration parameters include choice of stereo disparity filtering methods, because in early design stages some implementation details of stereo perception software may not be known well enough to make reasonable trade-off decisions.

This paper describes the relationships between the parameters above so an engineer can perform the following optimization task:

Configure a robot’s computer system to perform stereo calculations fast enough to enable a small response distance while maintaining acceptable fidelity of terrain models and not consuming more electrical power than the robot can generate.

The following sections will explain configuration equations that enable the above optimization task to be reasonably performed.
4. Formulation of Configuration Equations

4.1. Response Distance Equation

The first performance parameter is response distance, which is the product of the robot’s nominal velocity and the response time:

\[ D_{\text{response}} = v_{\text{robot}} T_{\text{response}} \quad (1) \]

\( T_{\text{response}} \) is defined as the sum of many factors, such as the time to:

- Take a sensor reading
- Perceive terrain
- Plan a path
- Command actuators
- Observe an actuator response
- Observe vehicle mechanism response

For the sake of this paper, we assume that the robot’s response time is primarily determined by the perception task:

\[ T_{\text{response}} \approx T_{\text{perc}} \quad (2) \]

In our experience, stereo perception is among the most computationally expensive tasks on a robot. Another important latency to be considered is that of the path planner. Path planning is less unified on a single strategy than stereo vision, so the execution times of different approaches vary significantly. Some build statistical models of the local and global area, while others are far more reactive. This paper therefore does not attempt to characterize \( T_{\text{plan}} \), but if planning latencies are on the scale of or larger than perception latencies, this paper’s conclusions should be considered only a piece of the performance analysis puzzle.

We assume here that the robot operates in this kinematic regime, which according to [Kelly98] is the case when the robot’s maneuver coefficient, \( s \):

\[ s = \frac{v_{\text{robot}}}{2\mu g(T_{\text{response}} + T_{\text{maneuver}})} \quad (3) \]

is less than 1. If this is not true, then generally non-computational factors such as actuator time constants must be considered when calculating vehicle response distance.

We seek to model a network of \( N_{\text{cpu}} \geq 1 \) hosts performing the stereo perception task in a distributed fashion (see Figure 1). In this model each CPU receives a complete pair of stereo images simultaneously. Unlike other some distributed systems that include a data-serving node, here no data-serving overhead is required. The model is reasonably valid when using an NTSC video signal or the isochronous portion of the bandwidth allocation of a IEEE-1394 bus [IEEE96]. Each CPU calculates a portion of the complete disparity map based on \( 1/N_{\text{cpu}} \) of the number of rows in a stereo image, meaning that each CPU is responsible for \( 1/N_{\text{cpu}} \) of the complete stereo calculation. When each CPU is finished generating the disparity map, it transmits its portion back to a single host. This operation is performed \( N_{\text{cpu}} - 1 \) times. Assuming that the latency of transmitting each portion of the map is network bandwidth-limited, each transmission takes \( 1/N_{\text{cpu}} \) of the time required to transmit the entire disparity map. Furthermore we assume that the network consists of a homogeneous cluster of CPUs. Invalidating this assumption requires more complicated representations to calculate performance parameters.

Using this model, the robot’s response time is shown in equation (4):

\[ T_{\text{response}} = \frac{T_{\text{stereo}}}{N_{\text{cpu}}} + \frac{T_{\text{network}}(N_{\text{cpu}} - 1)}{N_{\text{cpu}}} \quad (4) \]

Where the latency of stereo processing is based on a sum of two terms, one which is proportional to a CPU’s clock
period (based on the coefficient $\alpha$) and one which is constant with respect to CPU speed (given by the coefficient $\beta$):

$$T_{\text{stereo}} = \alpha T_{\text{cpu clk}} + \beta$$

(5)

Putting the whole thing together as a function of mission and configuration parameters leads to our first configuration equation:

$$D_{\text{response}} = \nu_{\text{robot}} \frac{\alpha T_{\text{cpu clk}} + \beta + T_{\text{network}}(N_{\text{cpu}} - 1)}{N_{\text{cpu}}}$$

(6)

### 4.2. Terrain Model Fidelity Equations

The terms $T_{\text{network}}$, $\alpha$ and $\beta$ are linearly proportional to the size of the stereo disparity map. The disparity map size is based on both image resolution and effective field of view. These two configuration parameters must be chosen properly to maintain an acceptable level of terrain model fidelity. However, maintaining a needlessly high level of fidelity slows stereo processing. A key trade-off described in this section is therefore between response distance and terrain model fidelity.

Disparity map size is based on the resolution, or number of pixels, of a stereo image. More specifically, following the common assumption that a stereo image size fits a 4-to-3 ratio:

$$N_{\text{col}} = \frac{4}{3} N_{\text{row}}$$

(7)

the dependencies on disparity map size can be expressed as:

$$T_{\text{network}}, \alpha, \beta \propto \text{disparity map size} \propto \left(\frac{3}{4} N_{\text{col}}^2\right)$$

(8)

However, image resolution must be chosen along with the stereo camera field of view. The robot’s size, maneuverability and obstacle-climbing capability affect these choices. By relating $T_{\text{network}}, \alpha$ and $\beta$ to these fundamental characteristics of a robot through further configuration equations, the engineer can understand trade-offs that determine the fidelity of terrain models generated by the stereo perception system.

#### 4.2.1. Stereo Camera Field of View

One weak constraint, depicted in Figure 2, is that at a distance of $D_{\text{response}} + y_{\text{sensor}}$ from the front of the camera, the stereo cameras must have a sufficient effective field of view to sense objects within an area as wide as the robot. This leads to the configuration equation:
For most robots this is not a strong constraint. A more significant constraint is placed when considering maneuverability. A popular method of planning a robot’s path is by considering curved rather than straight paths, a model necessary for vehicles incapable of point turns, such as Ackermann-steered vehicles. Traversable arcs are specified by their radius of curvature, with the tightest turns having the smallest radii. To properly plan a path that avoids obstacles it is necessary to sense a large enough patch of terrain to encompass all traversable arcs. It is practically sufficient to sense the tightest arc up to 90 degrees of curvature, since robots rarely follow a constant path longer than that (see Figure 3). Not meeting this constraint causes the robot to suffer from “tunnel vision” [Kelly94]. This constraint leads to the following configuration equation:

\[
\Gamma \geq 2 \tan \left( \frac{W}{2\sqrt{H^2 + (D_{\text{response}} + y_{\text{sensor}})^2}} \right)
\]

(9)

This equation generally impacts robot design more than the previous configuration equation, because \(2R_{\text{min}}\) is often much larger than \(W\). However, systems with pointed stereo cameras can use configuration equation 9 in place of equation 10, allowing the use of lenses with a much narrower field of view, which benefits image resolution.

Figure 2. Calculating minimum effective stereo field of view, \(\Gamma\), from robot dimensions

\[
\Gamma \geq 2 \tan \left( \frac{2R_{\text{min}}}{2\sqrt{H^2 + (R_{\text{min}} - (y_{\text{R}} - y_{\text{sensor}}))^2}} \right)
\]

(10)

Figure 3. Calculating minimum effective stereo field of view, \(\Gamma\), from robot maneuverability
4.2.2. Stereo Image Resolution

Once the field of view is constrained, the designer can focus on the necessary image resolution, or number of pixels in an image. The stereo camera image resolution is chosen to resolve objects at least as large as the smallest object the robot considers an obstacle. When dealing with stereo vision, the following configuration equation relates field of view and image resolution to acuity, or the minimum perceivable object size (see Section 11.1 for a derivation):

\[ h_{\text{min}} = \frac{2}{b N_{\text{col}}} \left( \frac{d^2}{2} \right) \]  

(11)

Where \( d \) is the distance from the cameras to the object, \( b \) is the stereo camera baseline and \( N_{\text{col}} \) is the number of pixel columns in an image.

This function increases with distance, so at some downrange distance, called \( y_{\text{max stereo}} \), the robot can no longer perceive obstacles. Re-arranging terms we have yet another configuration equation:

\[ y_{\text{max stereo}} = \left[ \frac{h_{\text{obs}} b N_{\text{col}}}{2 \tan \left( \frac{1}{2} \right)} \right] \]  

(12)

Methods to set \( y_{\text{max stereo}} \) appropriately are somewhat fuzzy. It seems reasonable to expect that:

\[ y_{\text{max stereo}} \geq R_{\text{min}} - (\gamma_R - \gamma_{\text{sensor}}) \]  

(13)

This ensures that the robot can at least find obstacles within 90 degrees of arc on its tightest available turn. This is useful to help the robot avoid obstacles with a turning rather than a stopping maneuver. However, larger values of \( y_{\text{max stereo}} \) can also be helpful to generate more effective path plans.

The previous sections have explained that the robot travels a distance of approximately \( D_{\text{response}} \) during each stereo cycle. For low-speed robots with reasonably fast processors, \( D_{\text{response}} \) can be rather small and there will be significant overlap between successive stereo image projections. This can be useful because stereo vision depth information can be sparse due to poor lighting or occlusions. However, designing too much overlap can be inefficient as much of the image is redundant. From the previous configuration equation we see that \( y_{\text{max stereo}} \) increases with the square root of \( N_{\text{col}} \). However we have seen previously that \( \alpha, \beta \) and \( T_{\text{network}} \), and hence \( D_{\text{response}} \), all increase with the square of \( N_{\text{col}} \). Therefore the engineer must find an optimal combination that maximizes \( y_{\text{max stereo}} \) and minimizes \( D_{\text{response}} \).

4.3. Power Consumption Equation

The final performance parameter is computing system power consumption. The relationships here are straightforward; the power required by the computing system scales linearly with the number of CPUs:

\[ P_{\text{computing}} = P_{\text{cpu}} N_{\text{cpu}} \]  

(14)

And the power consumption of each CPU is inversely proportional to its clock period:

\[ P_{\text{cpu}} = \frac{\chi}{T_{\text{cpuclk}}} \]  

(15)

Where \( \chi \) is some scaling factor. We’ll see later on that \( \chi \) varies not only amongst various CPU models but also amongst different operating voltages of CPUs in a single processor family.

Therefore the computing power consumption configuration equation is based directly on configuration parameters, not mission parameters:

\[ P_{\text{computing}} = \frac{\chi N_{\text{cpu}}}{T_{\text{cpuclk}}} \]  

(16)
5. Benchmarking Approach

5.1. Stereo Benchmarking

The Hyperion navigation architecture defines a base class used to convert image pairs to range data. Called StereoEngine, this class was instrumented to time the following sub-components of $T_{\text{stereo}}$:

- **StereoEngine::preProcessImages()**
  This method converts stereo images into their epipolar form.
- **StereoEngine::generateDisparity()**
  This method creates a disparity map from two stereo images. This is also where filtering and other post-processing is done.
- **StereoEngine::generatePtCloud()**
  This method converts a disparity map into a cloud of 3D points, which can be used to evaluate terrain traversability by the robot’s planning software.

Other classes inherit from StereoEngine to integrate implementation-specific stereo libraries. The advantage of instrumenting the StereoEngine class is that various stereo implementations can be easily benchmarked without changing code. Furthermore, the Hyperion architecture defines a second high-level stereo class, StereoSource, to handle the capturing of stereo image pairs. This code is separate from StereoEngine, so performance can be evaluated independent of the image source.

The timing benchmarks for these three methods are then repeated 100 times. Because a method latency measurement can never be lower than the true latency (for instance, unrelated processes such as timer interrupts can only result in higher latency measurements), only the minimum latency for each function over these 100 trials is recorded. Essentially, this results in a 100-best timing measurement, as described in [Bryant01]. Of course, if one runs the stereo engine code on a non-realtime operating system (such as Linux), one cannot expect the process to have a consistent run time, and margins may be required for reliable predictions of performance.

Pre-recorded stereo pairs were used by the SVS StereoEngine subclass, and the timings of the three methods above were recorded. Images were 320 x 240 and the SVS library uses two bytes to represent disparity, therefore disparity maps are 153,600 bytes in size. These tests were performed on two Dell Inspiron 8000 computers, both with Mobile Pentium III CPUs, 256 MB RAM, 256 kB L2 cache and identical 20 GB hard drives. Both ran Red Hat Linux 7.0. However, one CPU had a 700 MHz clock speed while the other CPU had a 1 GHz clock speed. This was how the dependence of stereo perception latency on CPU clock speed was measured, so that $\alpha$ and $\beta$ could be determined. However, these tests can only find constants for Mobile Pentium III CPUs. Extrapolating the results to other architectures is invalid; doing so would require further benchmarks on these CPUs. However, in his paper Konolige reports four-fold increases in SVS performance when running on MMX-compatible CPUs versus those without multimedia extensions.

5.2. Network Benchmarking

If stereo perception is to be distributed across multiple processors, the communication latencies must be modeled. The Hyperion system uses IPC [Simmons02] to transmit data between software modules. IPC enables anonymous publication / subscription of messages using TCP/IP. A central server is executed to enable modules to connect, declare what messages they wish to receive and so on. Therefore the latency of sending data across an IPC connection was measured. Because some times measured were very small, clock skew between computers could not be eliminated. Therefore our experimental setup was as follows. Host A transmitted a buffer of size $S$ to Host B. Host B received the buffer and transmits it back to Host A. By measuring the time between when Host A receives the message and when Host A receives the message, the round-trip time can be determined. The communication latency to send a buffer of size $S$ is then half this measured time. If $S$ is equal to the size of a stereo disparity image, then this communication latency is $T_{\text{network}}$. Note that the latency not only includes the time taken to transfer a buffer of size $S$ over the network's physical layer, but also processing required in the IPC central server.
A Pentium III 600 MHz computer was networked to a Pentium III 800 MHz computer using both a 10 Mbps and 100 Mbps Ethernet hub. Both hosts ran Red Hat Linux and used IPC version 3.6.2. Communication latencies were measured for buffers between 100 bytes and 10,000,000 bytes.

5.3. CPU Power Consumption Benchmarking
A survey of current Intel CPUs was made to estimate their power consumption. Data came from the Intel Developer web site. The value of $\chi$ used here is based on Mobile Pentium III CPUs, which were used in the stereo and networking benchmarks. This approach substituted for real measurements because it is difficult to discern the power required for a CPU from the total power required for a given computer.

6. Benchmarking Results

6.1. Stereo Benchmarking
Minimum measured times for the three benchmarked methods described above on both the 700 MHz and the 1 GHz CPUs are shown in Table 2.

<table>
<thead>
<tr>
<th>$T_{cpuclk}$</th>
<th>$T_{preProcessImages}$</th>
<th>$T_{generateDisparity}$</th>
<th>$T_{generatePtCloud}$</th>
<th>$T_{stereo}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.43 ns (700 MHz)</td>
<td>3522037 cycles</td>
<td>32097136 cycles</td>
<td>18949254 cycles</td>
<td>54568427 cycles</td>
</tr>
<tr>
<td>5.0 ms</td>
<td>45.9 ms</td>
<td>27.1 ms</td>
<td>78.0 ms</td>
<td></td>
</tr>
<tr>
<td>1.00 ns (1 GHz)</td>
<td>4357789 cycles</td>
<td>32862091 cycles</td>
<td>19393974 cycles</td>
<td>56613854 cycles</td>
</tr>
<tr>
<td>4.4 ms</td>
<td>32.9 ms</td>
<td>19.4 ms</td>
<td>56.7 ms</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Stereo benchmarking results

From these data we can find $\alpha$ and $\beta$, although with only two data points it remains unclear what variation on these constants can be expected. The constants here are $\alpha = 49.53$ ms/ns and $\beta = 7.17$ ms when the disparity map size is 153,600 bytes, therefore:

$$T_{stereo} = 49.53 \text{ms/ns} \cdot T_{cpuclk} + 7.17 \text{ms}$$  \hspace{1cm} (17)

This equation can be modified for smaller disparity maps by scaling $\alpha$ and $\beta$ as previously described in Section 4.2.

6.2. Network Benchmarking
A plot of IPC network benchmarking results are shown in Figure 4. Results over both 10 and 100 Mbps connections are given. When dealing with buffers of less than 1000 bytes it appears that IPC internal data handling code is the primary contributor to transmission latency, hence the response time for networks performing distributed stereo calculations on portions of a disparity map less than 1000 bytes could not be modeled by the configuration equation (6).

It is interesting to note that data transfers via IPC are at most 71% bandwidth efficient over a 10 Mbps Ethernet hub and at most 27% bandwidth efficient over a 100 Mbps Ethernet hub. Even if a raw TCP socket had been used, tests show that only at most 31% bandwidth efficiency was possible. Therefore if a stereo perception system must use large disparity maps, distributing the stereo perception task amongst multiple CPUs may only be feasible with lower-overhead protocols such as UDP.

6.3. CPU Power Consumption Benchmarking
Intel-provided power consumption data for Pentium II, Mobile Pentium II, Pentium III and Mobile Pentium III CPUs are displayed in Figure 5. In it, one can note several discontinuous linear segments representing a model of CPU manufactured with a given process, sometimes to operate at different voltages. Therefore the $\chi$ coefficient (which is
inversely proportional to the slopes of line fits in Figure 5) does not even apply across a single model of CPU. For instance, an 800 MHz Mobile Pentium III operates at 1.4 V and 9.8 W, while an 866 MHz Mobile Pentium III operates at 1.14 V and 19.5 W. Therefore in some cases it may be possible to reduce power consumption by distributing the stereo perception task amongst multiple CPUs, since the power requirements per computational speed unit can be so nonlinear.

7. Application of Results

As a case study, the method described above was applied to the design of the stereo system of the Hyperion robot. This application uncovers some possible inefficiencies in the existing design, as well insight into some additional configuration parameters and equations that will improve the design method proposed in this paper. To draw useful comparisons, we make two important assumptions:

1. Many of Hyperion’s configuration parameters are fixed. Applying the configuration equations from first principles may argue for drastically different configuration parameter values, which would make a comparison to the existing robot difficult.
2. Hyperion’s response time is dominated by its stereo processing latency. Unfortunately, this assumption is not true. Hyperion’s navigation system involves computationally complex terrain modeling algorithms. However, if we further assume that the latencies of these algorithms scale linearly with disparity map size as stereo
processing does, initial benchmarks argue that roughly tripling the theoretical response times given by equation (6) is somewhat realistic:

\[ D_{\text{response}} = 3v_{\text{robot}} \cdot \alpha T_{\text{cpu}} + \beta + T_{\text{network}}(N_{\text{cpu}} - 1) \]

(18)

2.1. Application of Configuration Equations

Hyperion’s minimum turning radius is 2.5 m. The stereo cameras are essentially placed at the front of the vehicle, so \( y_{\text{sensor}} \) is 0 m. \( H \) is 1.44 m off the robot’s ground plane. Finally, \( y_R \) is 1.0 m. By configuration equation (10):

\[ \Gamma \geq 2 \cdot \text{atan} \left( \frac{5 \text{m}}{2 \sqrt{(1.44 \text{m})^2 + (2.5 \text{m} - (1 \text{m} - 0 \text{m}))^2}} \right) = 100.5 \text{deg} \]

(19)

Hyperion’s stereo system requires special consideration because its stereo cameras are mounted to the steering axle so it is pointed in the direction of steering. By pointing the stereo field of view, the robot can avoid the tunnel problem. Therefore the design must only meet the weaker version of the field of view configuration equation, which is based on vehicle width rather than maneuverability. Because configuration equation (9) is inversely proportional to response distance, we consider a case of zero response distance to derive the widest possible necessary field of view:

\[ \Gamma \geq 2 \cdot \text{atan} \left( \frac{1.75 \text{m}}{2 \sqrt{(1.44 \text{m})^2 + (0 \text{m} + 0 \text{m})^2}} \right) = 62.6 \text{deg} \]

(20)

The radius of Hyperion’s wheels is 0.325 m. From a purely kinematic perspective wheeled vehicles are capable of surmounting obstacles up to their wheel radius in height. We will however follow Hyperion’s existing value of \( h_{\text{obs}} \), which is conservatively set at 0.2 m. Nyquist’s Sampling Theorem forces Hyperion’s stereo perception system to detect objects 10 cm in size to reliable detect obstacles \( h_{\text{obs}} \) in size. We can use the equation (11) to examine the effect of minimum obstacle size and the number of pixel columns on \( y_{\max \text{stereo}} \) as seen in Figure 6:

![Figure 6. Effect of \( h_{\text{obs}} \) on \( y_{\max \text{stereo}} \) for Hyperion stereo system (20 cm baseline, \( \frac{1}{8} \) sub-pixel disparity) for various image resolutions.](image)

If we must set \( h_{\text{obs}} \) at 0.1 m, Figure 6 shows that a stereo image with only 80 columns could allow the SVS stereo engine, which is capable of \( \frac{1}{8} \) sub-pixel disparity, to discern an obstacle well past \( R_{\min} - (y_R - y_{\text{sensor}}) \), or 1.5 m. In
fact, such an image could discern obstacles out to 2.9 m. Note that decreasing the stereo camera height, H, could increase $y_{\text{maxsensor}}$ but at the cost of being more susceptible to severe occlusions from smaller objects and requiring a wider field of view.

Table 3 depicts the effect of $N_{\text{col}}$ on the disparity map size (assuming disparity is represented by a 2-byte short integer) $\alpha$, $\beta$ and $T_{\text{network}}$ (assuming a 100 Mb/s Ethernet network using IPC):

<table>
<thead>
<tr>
<th>$N_{\text{col}}$</th>
<th>Disparity Map Size</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$T_{\text{network}}$ (with IPC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>9,600 bytes</td>
<td>3.10 ms/\text{ns}</td>
<td>0.45 ms</td>
<td>3 ms</td>
</tr>
<tr>
<td>160</td>
<td>38,400 bytes</td>
<td>12.38 ms/\text{ns}</td>
<td>1.79 ms</td>
<td>12 ms</td>
</tr>
<tr>
<td>320</td>
<td>153,600 bytes</td>
<td>49.53 ms/\text{ns}</td>
<td>7.17 ms</td>
<td>46 ms</td>
</tr>
</tbody>
</table>

Table 3: Effect of $N_{\text{col}}$ on important constants

With such a small disparity map, the response distance configuration equation states that even with a single slow CPU, Hyperion’s response distance remains very small. Even without consideration of the power consumption configuration equation, this extremely small result supports the suggestion that Hyperion’s stereo processing be performed by a single slow CPU rather than the 500 MHz Mobile Pentium III initially selected. This older Mobile Pentium III processor board in Hyperion could theoretically generate an 80x60 disparity map in about 7 ms. However, it consumes 21 W under normal CPU utilization.

The configuration equations and approach described in this paper suggest that a much slower, and therefore presumably lower-power, system could have supported sufficient stereo perception. Hyperion’s success with its current configuration serves as an existence proof that its value of $D_{\text{response}}$ is acceptable. Hyperion’s single 500 MHz Mobile Pentium III, 320x240 disparity maps and 0.3 m/s speed lead to a small $D_{\text{response}}$ of 3.2 cm. Plotting configuration equation (6), Figure 7 shows that even a single Pentium III CPU of any speed can maintain less than a

Figure 7. From configuration equation (6), a single Pentium III CPU of any speed is sufficient to maintain a response distance of less than 3 cm
3 cm value of $D_{\text{response}}$ with an 80x60 disparity map and 0.3 m/s speed. Interestingly enough, the analysis performed here shows that rather than being the prime motivator for a fast CPU, stereo vision on Hyperion probably requires no more CPU than other components of its autonomy system.

### 2.2. Configuration Equation Evaluation

Two tasks are required to evaluate the configuration equations applied above. First, the theoretical values of $T_{\text{stereo}}$ for various CPUs must be verified. Specifically, configuration equation (6) implies that the equivalent of a 200 MHz Intel Pentium III could generate 80x60 stereo disparity maps in about 17 ms. Even taking into account Hyperion’s complex terrain modeling algorithms with equation (18), the total response time for 80x60 would only be 51 ms. At this time benchmarks of various CPUs with various disparity map sizes has not yet been carried out to verify this finding.

A second task effort must investigate the impact of the above design decisions on the fidelity of terrain models created by the stereo software. Configuration equations (11) and (12) predict that an 80x60 disparity map could detect objects that Hyperion considers obstacles at a reasonable distance. However, this prediction must be verified. Fortunately an array of images from Hyperion’s stereo cameras were logged during its expedition to the Arctic in July 2001. This valuable data set represents readings of terrain in lighting conditions relevant to Hyperion’s mission.

Five pairs of stereo images were used to see how well the Hyperion stereo perception hardware and software could perform given images of various resolutions from 320x240 (the original resolution), 160x120 and 80x60. Image resolution was lowered for testing through simple decimation. The left image from each pair is shown in Figure 8. The images represent a fair mix of flat ground, a discrete obstacle and rock fields. While exact measurements were not made, the heights of obstacles found in some of these images are roughly between 10 and 30 cm. Additionally

![Figure 8. Stereo image pairs used for evaluation, taken by Hyperion in real Arctic terrain](image-url)
because height measurements were not taken, our evaluation must assume that the terrain models resulting from the 320x240 disparity maps are of a reasonable fidelity.

The evaluation procedure was carried out as follows. Image pairs were decimated to 160x120 and 80x60 pixels. The images, along with stereo camera calibration parameters used on Hyperion’s expedition, were input to the SVS StereoEngine to create a disparity map. The disparity map was then processed by Hyperion’s Stereo Mapper software, which transforms the disparity maps into terrain models. The Stereo Mapper terrain model is a 2.5-D, grid-based map. Each grid cell is assigned a “goodness” value, which ranges from zero (a very unsafe region) to one (a very safe region). An OpenGL-based interface was used to display the Stereo Mapper terrain maps. The interface depicted safe cells as green, somewhat safe cells as yellow and unsafe cells as red.

The evaluation metric used involved visually comparing the terrain models resulting from various stereo image resolutions. If both 320x240 and 80x60 stereo images result in terrain models showing similar safe and unsafe regions, then at least for Hyperion configuration equations (11) and (12) are valid for configuring the stereo perception system. If noticeably different terrain models result, then these equations should be changed and new equations possibly added.

Figure 9 shows the terrain models created from the second image in Figure 8, which contains a single obstacle. Notice that the terrain model generated from 160x120 images is significantly smaller than generated from 320x240 images. This is because the statistical calculations of grid cell “goodness” that the Stereo Mapper performs require a minimum number of range measurements to lie in each cell. When image resolution is decreased, the overall density of range measurements decreases. At some point, the range measurement density beyond some distance becomes too low for the Stereo Mapper to calculate goodness of a grid cell. The range measurement density of 80x60 images was simply too low for the Stereo Mapper to calculate goodness at any distance. Hence, Figure 9 does not include a terrain model from the 80x60 images. One approach to overcoming this problem is to increase the terrain map’s grid cell size. Figure 10 shows a new Stereo Mapper terrain model, which has been generated from 160x120 images but uses doubles the grid cell size from 12.5 cm, as was used above in Figure 9, to 25 cm. Please see Section 12 for terrain models generated from each test stereo image pair.

This exercise argues for an additional, new configuration equation relating stereo image resolution to acuity. Section 4.2 related resolution to acuity in purely geometric terms, but it seems that modeling the sensor signal-to-noise characteristics would yield more accurate performance predictions.
3. Conclusions and Future Work

This paper has outlined several configuration equations that describe relationships involved when designing a computer system to support stereo perception on an autonomous mobile robot. The research here supports the use of configuration equations in robot computer system design, and the application of these equations to the Hyperion robot argues that a less powerful, and therefore power-hungry, CPU could suitably meet performance requirements. However, several shortcomings still exist that prevent us from quantifying the performance margins of Hyperion’s existing computer system:

1. The lack of benchmarks for Hyperion’s terrain modeling software. Simply assuming that this software requires roughly twice the CPU time of Hyperion’s stereo processing code is simplistic. The latencies involved with terrain modeling may not scale linearly with disparity map size, as does $T_{\text{stereo}}$.
2. The existing configuration equations predict that Hyperion could use very low resolution stereo images to detect obstacles. However this prediction is not supported in tests using real stereo data. Therefore additional configuration equations are needed that relate the signal-to-noise characteristics of stereo cameras to the terrain model fidelity. These equations would depict a more realistic relationship between stereo camera field of view, stereo image resolution and the resulting fidelity of stereo-based terrain models.

Furthermore, many low-power computer systems incorporate CPUs from other processor families such as PowerPC or StrongARM. Therefore benchmarking stereo software on these CPUs would increase this paper’s applicability. Further insights could also be gained by benchmarking stereo engines other than the SVS as well as popular terrain modeling and path planning software.

3. References


4. Table of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Coefficient of stereo latency configuration equation</td>
<td>ms / ns</td>
</tr>
<tr>
<td>$b$</td>
<td>Stereo camera baseline</td>
<td>m</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Coefficient of stereo latency configuration equation</td>
<td>ms</td>
</tr>
<tr>
<td>$\chi$</td>
<td>Coefficient of power consumption configuration equation</td>
<td>W ns</td>
</tr>
<tr>
<td>$d$</td>
<td>Distance from stereo camera to object</td>
<td>m</td>
</tr>
<tr>
<td>$D_{response}$</td>
<td>Robot response distance</td>
<td>m</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>Effective field of view of stereo camera pair</td>
<td>rad</td>
</tr>
<tr>
<td>$g$</td>
<td>Acceleration on the due to gravity</td>
<td>m / s$^2$</td>
</tr>
<tr>
<td>$H$</td>
<td>Height of stereo camera from the ground</td>
<td>m</td>
</tr>
<tr>
<td>$h_{obs}$</td>
<td>Minimum object height that the robot considers an obstacle</td>
<td>m</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Coefficient of friction</td>
<td></td>
</tr>
<tr>
<td>$N_{col}$</td>
<td>Number of pixel columns in a stereo image</td>
<td></td>
</tr>
<tr>
<td>$N_{cpu}$</td>
<td>Number of CPUs</td>
<td></td>
</tr>
<tr>
<td>$N_{row}$</td>
<td>Number of pixel rows in a stereo image</td>
<td></td>
</tr>
<tr>
<td>$P_{computing}$</td>
<td>Power required by entire computing system</td>
<td>W</td>
</tr>
<tr>
<td>$P_{cpu}$</td>
<td>Power required by a single CPU</td>
<td>W</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>Tilt of stereo camera</td>
<td>rad</td>
</tr>
<tr>
<td>$R_{min}$</td>
<td>Minimum traversable arc radius</td>
<td>m</td>
</tr>
<tr>
<td>$T_{cpuclk}$</td>
<td>CPU clock period</td>
<td>s</td>
</tr>
<tr>
<td>$T_{maneuver}$</td>
<td>Robot mechanism maneuver latency</td>
<td>s</td>
</tr>
<tr>
<td>$T_{network}$</td>
<td>Latency of network communication for a given disparity map size</td>
<td>s</td>
</tr>
<tr>
<td>$T_{perc}$</td>
<td>Perception system latency</td>
<td>s</td>
</tr>
<tr>
<td>$T_{plan}$</td>
<td>Planning software latency</td>
<td>s</td>
</tr>
<tr>
<td>$T_{response}$</td>
<td>Robot response time</td>
<td>s</td>
</tr>
<tr>
<td>$T_{stereo}$</td>
<td>Latency of stereo calculations for a given disparity map size</td>
<td>s</td>
</tr>
<tr>
<td>$v_{robot}$</td>
<td>Nominal robot velocity</td>
<td>m / s</td>
</tr>
<tr>
<td>$W$</td>
<td>Robot body width</td>
<td>m</td>
</tr>
<tr>
<td>$y_{maxsensor}$</td>
<td>Maximum downrange (y-axis) distance that the stereo system can perceive obstacles</td>
<td>m</td>
</tr>
<tr>
<td>$y_{r}$</td>
<td>Distance from the front of the robot to the origin of the robot’s coordinate frame</td>
<td>m</td>
</tr>
<tr>
<td>$y_{sensor}$</td>
<td>Distance from front of robot to stereo camera along the y-axis</td>
<td>m</td>
</tr>
</tbody>
</table>
5. Additional Derivations

5.1. Stereo Distance Resolution Equation

\[ d + \Delta d = \frac{bf}{\text{disp} + \Delta\text{disp}} \]

\[ \Delta d = \frac{bf}{\text{disp} + \Delta\text{disp}} - d \]

\[ \Delta d = \frac{bf}{d + \Delta\text{disp}} - d \]

\[ \Delta d = \frac{1}{\frac{bf}{d + \Delta\text{disp}}} - d \]

\[ \left( \frac{\Delta d}{bf} \right) \left( \frac{bf}{d + \Delta\text{disp}} \right) = 1 - d \frac{bf}{bf + \Delta\text{disp}} \]

\[ \frac{\Delta d + \Delta d\Delta\text{disp}}{bf} = \frac{-d\Delta\text{disp}}{bf} \]

\[ \frac{\Delta d}{d} = -(\Delta d + d)\frac{\Delta\text{disp}}{bf} \]

\[ \Delta d = -(\Delta d + d)\frac{\Delta\text{disp}}{bf} \]

\[ \Delta d \equiv -d\frac{\Delta\text{disp}}{bf} \]

\[ \Delta d \equiv -d^2\frac{\sqrt{\frac{\Gamma}{2}}}{bN_{col}} \]

\[ h_{min} \equiv d^2\frac{\sqrt{\frac{\Gamma}{2}}}{bN_{col}} \]
6. Terrain Models from Stereo Image Pairs Used for Evaluation

6.1. Terrain Models from 320x240 Stereo Image Pairs

**Image0000:** here the robot is viewing essentially flat terrain. This is reflected in the terrain map, which is full of traversable grid cells out nearly 7 m in front of the Hyperion’s coordinate frame origin, or 6.2 m away from the stereo cameras.

**Image0001:** here the robot is viewing a single, discrete object that is considered an obstacle. It shows up clearly in the terrain map as a single unsafe, red region. The density of data is similar to the image0000 map above and the traversability of grid cells are known beyond 6 m from the stereo cameras.

**Image0002:** a rock field is viewed in this image. In the terrain map Hyperion perceives several areas of imperfect but passable terrain and a single large obstacle. The height and distance of the obstacle is such that it casts a shadow of occlusion in the terrain map. Other than this, the density and distance of valid grid cells is similar to the other maps based on 320x240 images.
Image0004: this is a third patch of the rock field. Several passable but difficult regions are shown in yellow, as is a single obstacle. Once again, density and distance of valid grid cells is similar to the other maps based on 320x240 images.

Image0003: another patch of the rock field is viewable here. Now Hyperion perceives several obstacles, resulting in many red and yellow regions. Still, the density and distance of valid grid cells is similar to the other maps based on 320x240 images.
6.2. Terrain Models from 160x120 Stereo Images Pairs

6.2.1. Using 12.5 x 12.5 cm Grid Cells

Image0000: again the robot is viewing essentially flat terrain but with 160x120 images the terrain map can calculate traversability only about 4 m in front of Hyperion, or only 3.3 m away from the stereo cameras. This is because at a lower resolution, the Stereo Mapper is given too few range measurements in distant cells to calculate a traversability.

Image0001: in this map the single discrete does show up as a red region. As above, the region of known terrain is very small, only extending to a 3.3 m distance from the cameras. Also note that the obstacle region here is slightly larger than that of the 320x240-based terrain map. Also, the obstacle's position is different by around 30 cm. These differences will be meaningless when the planner performs C-space expansion.

Image0002: this terrain map does not contain the obstacle seen in the 320x240 image, because the obstacle lies too far away.
Image0003: again the terrain map is too small to contain the two obstacles that are seen in the corresponding 320x240-based terrain map. However, even with 160x120 images and 12.5 x 12.5 cm grid cells, the closer obstacle can be seen.

Image0004: although an obstacle in the scene should be within the range of this terrain map, only a yellow (“somewhat unsafe”) indication is given in the region where the obstacle existed in the 320x240-based terrain map.
6.2.2. Using 25 x 25 cm Grid Cells

**Image0000**: here the robot is viewing essentially flat terrain. This is reflected in the terrain map, which is full of traversable grid cells out nearly 7 m in front of the Hyperion’s coordinate frame origin, or 6.2 m away from the stereo cameras. Note that in this terrain map, each cell represents a 25 x 25 cm area. Therefore when using decimated stereo image pairs, we need to decrease mapping resolution to maintain the distance at which objects can be detected.

**Image0001**: here the robot is viewing a single, discrete object that is considered an obstacle. It shows up clearly in the terrain map as a single unsafe, red region. The size and density of the map is similar to the image0000 map above. Additionally, notice that the obstacle appears in the same position it did when using 12.5 x 12.5 cm, 160x120 images.

**Image0002**: a rock field is viewed in this image. Although the 320x240-based terrain maps clearly detect an obstacle near the center of the image, here the obstacle is not seen. Instead, a small patch of yellow is generated. This example shows that the 25 x 25 cm, 160x120-based terrain maps have less fidelity than the 320x240-based maps.
Image0003: another patch of the rock field is viewable here. While two obstacles exist in this scene, the terrain map only correctly marks one unsafe region. However, the second obstacle is about 6 m away from the camera, at the distant edge of the terrain map. We can therefore assume that it would be detected in subsequent maps as the robot drove further ahead.

Image0004: this is a third patch of the rock field. The 320x240-based maps show a single obstacle marked as red and several potentially dangerous regions marked yellow. In this map the obstacle appears yellow rather than red. Again, the fidelity of this map may be unacceptably low, potentially allowing the robot to drive into an obstacle.