Panoramic stereo vision and depth-assisted edge detection

Aris Economopoulos, Drakoulis Martakos
National and Kapodistrian University of Athens, Department of Informatics & Telecommunications
Multimedia Technology Laboratory
Panepistimiopolis, Ktiria Pliroforikis, 15784 Athens, Greece
{pathway, martakos}@di.uoa.gr

Peter Peer*, Franc Solina
University of Ljubljana, Faculty of Computer and Information Science
Computer Vision Laboratory
Tržaška 25, 1000 Ljubljana, Slovenia
{peter.peer, franc.solina}@fri.uni-lj.si

Abstract

This article deals with the fusion of two initially independent systems. The first system utilizes panoramic images and rotational geometry to create stereo image pairs and recover depth information using a single camera. The second system utilizes dense depth estimates in order to drive a process, which improves edge detection by factoring in the distance (scale) from the camera at each point. Distance data is fed to a smoothing pre-processor, which splits the image into layers based on pixel distance and independently smoothes each one, utilizing a different sigma. This variation of sigma across image depth ensures that noise is eliminated with minimal loss of overall detail. Significant improvements to the results of subsequent edge detection operations are observed for images with great depth variations. The methods utilized by both systems are analysed and their benefits and drawbacks are exposed. The process of integration is also detailed, outlining problems and solutions that were adopted. The applications and constraints of the system are also discussed.

Keywords: panoramic image, depth image, edge detection

*Corresponding author: +386 1 4768 381 (tel.), +386 1 4264 647 (fax)

1. Introduction

This paper deals with two discrete subjects. Two separate systems are presented, one dealing with panoramic stereo vision and the other with stereo-assisted edge detection. Both systems were developed independently.

1.1. Motivation

Edge detection is one of the most broadly used operations in the analysis of images. There has been a plethora of algorithms presented within this context and several different approaches to the subject have been well documented [4, 6, 10, 13, 21].

In the vast majority of cases, edge detection has been considered to be an early step in the process of analysis. This has allowed researchers to effectively utilize edge data to guide higher-level operations. In addition to its utilization in other areas, edge detection has also been used in several applications of stereoscopy in order to improve stereo correlation algorithms and to aid in overall system robustness [9, 11, 21].

There is very little research however, that looks at the relationship of stereopsis and edge detection from a different standpoint. Specifically, we submit that in humans, whose vision system our discipline is after all trying to model, the presence of robust and efficient stereoscopy may actually be implicitly aiding the process of edge detection, rather than the other way around. Modern medicine and neuroscience are as of yet unable to provide a precise blueprint of the algorithms inherent in human visual perception. We have precious little information on how the brain processes visual stimuli, and the order in which operators are applied and the possible interdependencies that they exhibit are not among it. To explore this hypothesis, and to establish whether it could hold true as well as whether it could be applied to computer vision problems, we decided to analyze the workings of an existing edge detection operator and enhance it so as to make use of depth data.

The assumption made here, of course, is that a robust stereo vision system that does not depend on edge detection techniques is available. Such a system is presented in [23]. An alternative approach that is able to produce dense depth maps, and hence drive the aforementioned process, is panoramic stereo imaging. A standard camera has a limited field of view, which is usually smaller than the human field of vision. Because of that fact, people have always tried to generate images with a wider field of view, up to a full 360 degrees panorama [17]. Points and lines should be visible on all scene images. This is a property of panoramic cameras and represents our fundamental motivation for the generation of depth images using this technique. If we would try to build two panoramic images simultaneously by using two standard cameras, we would have problems since the scene would not be static.

In this article, we attempt to show how knowledge of a scene's stereo geometry and corresponding disparities can be used to assist in the task of edge detection. To test our theories, we chose to work with the Canny algorithm for edge detection [2]; an operator that has received much attention, and rightly so, for it far outperforms most of its competitors. However, the Canny edge detector, like many others, has a serious drawback, if not limitation. To achieve usable results, the process of edge detection must be preceded by the application of a Gaussian smoothing filter, the standard deviation σ of which severely affects the quality of the output image.

More specifically, it is known that the process of eliminating noise without also eliminating crucial image detail is a delicate one. The more an image is smoothed, the more its finer details blend into the background and fail to be picked up by the edge detector. Thus, a balance must be achieved here; traditionally, this balance has been achieved by varying σ until the results of the operation are satisfactory.

1.2. Structure of the article

In the next section, we present our mosaic-based panoramic stereo system. The geometry of the system, the epipolar geometry involved, and the procedure of stereo reconstruction are explained in detail. Furthermore, we describe the contribution of our work, and offer an overview of related work. The focus of the section is on analyzing the system's capabilities.

Section 3 gives a detailed description of our method for depth-assisted edge detection via layered scale-based smoothing. This method essentially improves the results of an existing edge detection operator by incorporating into the process the depth data offered by the stereo system presented in the previous section.

Section 4 gives experimental result for both systems and describes their integration.

The final section of the article is used to summarize the main conclusions.

2. Panoramic stereo system

2.1. System hardware

In Figure 1 you can see the hardware part of our system: a rotational arm is rotating around the vertical axis; a pole is fixed on the rotational arm, enabling the offset of the optical center of the camera from the system's rotational center; on it, we have fixed one standard color camera looking outwards from the system's rotational center. Panoramic images are generated by moving the rotational arm by an angle corresponding to one column of the captured image.

It can be shown that the epipolar geometry is very simple if we are doing the reconstruction based on a symmetric pair of

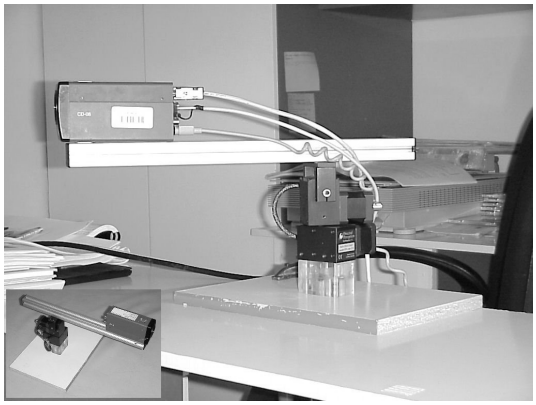


Figure 1. System hardware.

stereo panoramic images. We get a symmetric pair of stereo panoramic images when we take symmetric stripes on the left and on the right side from the captured image center column.

2.2. Related work

We can generate panoramic images with the help of special panoramic cameras or with the help of a standard camera and mosaicing standard images in panoramic images. If we want to generate mosaiced 360 degrees panoramic images we have to move the camera on a closed path, which is in most cases a circle.

One of the best known commercial packages for creating mosaiced panoramic images is QTVR (QuickTime Virtual Reality). It works on the principle of sewing together a number of standard images captured while rotating the camera [3]. Peleg et al. [16] introduced the method for creation of mosaiced panoramic images from standard images captured with a handheld video camera. A similar method was suggested by Szeliski and Shum [20] which also do not strictly constrain the camera path but assume that there is no great motion parallax effect present. All the methods mentioned so far are used only for visualization purposes since the authors did not try to reconstruct the scene.

Ishiguro et al. [9] suggested a method which enables the reconstruction of the scene. They used a standard camera rotating on a circle. The scene is reconstructed by means of mosaicing together panoramic images from the central column of the captured images and moving the system to another location where the task of mosaicing is repeated. Two created panoramas are then used as input in stereo reconstruction procedure. The depth of an object was first estimated using projections in two images captured on different locations of the camera on the camera's path. But since their primary goal was to create a global map of the room, they preferred to move the system attached to the robot about the room.

Peleg and Ben-Ezra [14, 15] introduced a method for the creation of stereo panoramic images. Stereo panoramas are

created without actually computing the 3D structure — the depth effect is created in the viewer's brain.

In [19], Shum and Szeliski described two methods used for creation of panoramic depth images, which are using standard procedures for stereo reconstruction. Both methods are based on moving the camera on a circular path. Panoramas are built by taking one column out of the captured image and mosaicing the columns. They call such panoramas *multiperspective panoramas*. The crucial property of two or more multiperspective panoramas is that they capture the information about the motion parallax effect, while the columns forming the panoramas are captured from different perspectives. The authors are using such panoramas as the input for the stereo reconstruction procedure.

However, multiperspective panoramas are not something entirely unknown to the vision community [19]. They are a special case of *multiperspective panoramas for cell animation* [22], and are very similar to images generated with by the following procedures: *multiple-center-of-projection* [18], *manifold projection* [16] and *circular projection* [14, 15]. The principle of constructing the multiperspective panoramas is also very similar to *the linear pushbroom camera* principle for creating panoramas [7].

In articles closest to our work [9, 19], we missed two things: system capabilities analysis and searching for corresponding points using the standard correlation technique. This is why in this article we focus on these two issues. While in [9], the authors searched for corresponding points by tracking the feature from the column building the first panorama to the column building the second panorama, the authors in [19] used an upgraded *plain sweep stereo* procedure.

2.3. System geometry

Let us begin this section with a description of how the stereo panoramic pair is generated. From the captured images on the camera's circular path we always take only two columns which are equally distant from the middle column. The column on the right side of the captured image is then mosaiced in the left eye panoramic image and the column on the left side of the captured image is mosaiced in the right eye panoramic image. Thus, we are building the panoramic image from only one column of the captured image. Thus, we get a symmetric pair of panoramic images.

The geometry of our system for creating multiperspective panoramic images is shown in Figure 2. When created, they are used as an input to create panoramic depth images. Point C denotes the system's rotational center around which the camera is rotated. The offset of the camera's optical center from the rotational center C is denoted as r, describing the radius of the circular path of the camera. The camera is looking outwards from the rotational center. The optical center of the camera is marked with O. The selected column of pixels that will be sewn in the panoramic image contains the projection of point

P on the scene. The distance from point P to point C is the depth l and the distance from point P to point O is denoted by d. θ defines the angle between the line defined by point C and point O and the line defined by point C and point P. In the panoramic image, θ gives the horizontal axis describing the path of the camera. By φ we denote the angle between the line defined by point O and the middle column of pixels of the captured image and the line defined by point O and selected column of pixels that will be mosaiced in the panoramic image. Angle φ can be thought of as a reduction of the camera's horizontal view angle α.

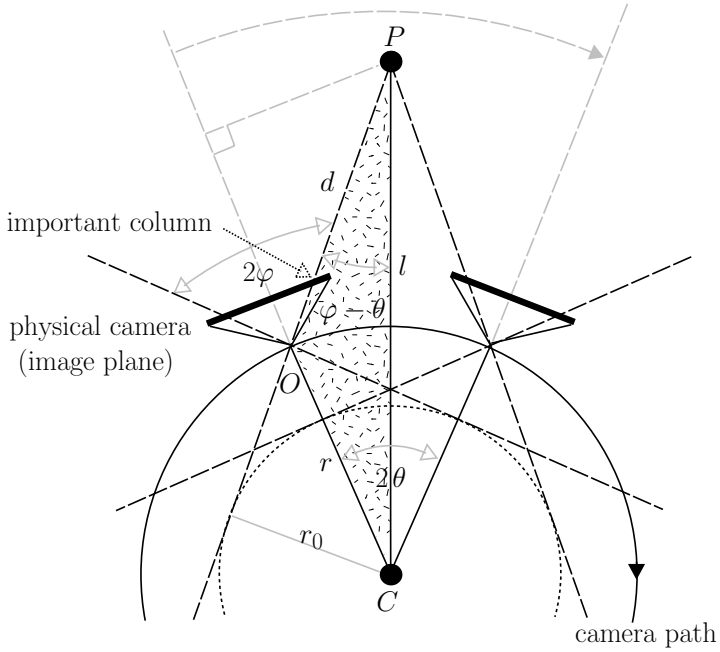


Figure 2. System geometry for construction of a multiperspective panorama. Note that a ground-plan is presented. The optical axis of the camera is kept horizontal.

The system in Figure 2 is obviously a non-central since the light rays forming the panoramic image are not intersecting in one point called the viewpoint, but instead are tangent ($\varphi \neq 0$) to a cylinder with radius r_0 called the viewing cylinder. Thus, we are dealing with panoramic images formed by a projection from a number of viewpoints. This means that a captured point on the scene will be seen in the panoramic image only from one viewpoint.

For stereo reconstruction we need two images. If we are looking at only one circle on the viewing cylinder (Fig. 2) then we can conclude that our system is equivalent to a system with two cameras. In our case two virtual cameras are rotating on a circular path, i.e. viewing circle, with radius r_0. The optical axis of a virtual camera is always tangent to the viewing circle. The panoramic image is generated from only one pixel from the middle column of each image captured by a virtual

camera. This pixel is determined by the light ray which describes the projection of the scene point on the physical camera image plane. If we observe a point on the scene P, we see that both virtual cameras which see this point, form a traditional stereo system of converging cameras.

2.4. Epipolar geometry

In this section we will only illustrate the procedure of the proof that epipolar lines of the symmetrical pair of panoramic images are image rows. This statement is true for our system geometry. For proof see [8].

The proof is based on the radius r_0 of the viewing circle (Figure 2). We can express r_0 in terms of known quantities r and φ as: $r_0 = r \cdot \sin\varphi$. We can treat value r_0 as the radius of the captured panoramic image while we get the same panoramic image in the case where we rotate a line camera on a circular path with radius r_0 and with a line camera's optical axis tangent to this circle.

We can carry out the proof in three steps: *first*, we have to execute the projection equation for the line camera, *then* we have to write the projection equation for the multiperspective panoramic image and in the *final* step we can prove the property of the epipolar lines for the case of a symmetrical pair of panoramic images. In the first step we are interested in how the point on the scene is projected to the camera's image plane [6] which has in our case, while we are dealing with a line camera, a dimension of $n \times 1$ pixels. In the second step we have to write the relations between different notations of a point on the scene and projection of this point on the panoramic image: notation of the scene point in Euclidean coordinates of the world coordinate system and in cylindrical coordinates of the world coordinate system, notation of the projected point in angular coordinates of the (2D) panoramic image coordinate system and in pixel coordinates of the (2D) panoramic image coordinate system. When we know the relations between mentioned coordinate systems we can write the equation for the projection of the scene points on the image plane (cylinder) of the panorama. Based on angular coordinates of the panoramic image coordinate system property, we can in the third step show that the epipolar lines of the symmetrical pair of panoramic images are actually rows of panoramas. The basic idea for the last step of our proof is as follows: If we are given an image point on one panoramic image, we can express the optical ray defined by a given point and the optical center of the camera in the 3D world coordinate system. If we project this optical ray described in the world coordinate system onto the second panoramic image, we get an epipolar line corresponding to given image point on the first panoramic image.

2.5. Stereo reconstruction

Let us go back to Figure 2. Using trigonometric relations evident from the sketch we can write the equation for depth estimation l of point P on the scene. Using the basic law of sines for triangles, we have:

$$\frac{r}{\sin(\varphi-\theta)} = \frac{d}{\sin\theta} = \frac{l}{\sin(180^{\mathrm{o}}-\varphi)},$$

and from this equation we can express the equation for depth estimation l as:

$$l = \frac{r\cdot\sin(180^{\mathrm{o}}-\varphi)}{\sin(\varphi-\theta)} = \frac{r\cdot\sin\varphi}{\sin(\varphi-\theta)}. \tag{1}$$

From eq. (1) follows that we can estimate the depth l only if we know three parameters: r, φ and θ. r is given. Angle φ can be calculated with regard to the camera's horizontal view angle α as:

$$2\varphi = \frac{\alpha}{W}\cdot W_{2\varphi}, \tag{2}$$

where W is the width of the captured image in pixels and $W_{2\varphi}$ is the width of the captured image between columns forming the symmetrical pair of panoramic images, given also in pixels. To calculate the angle θ we have to find corresponding points on panoramic images. Our system works by moving the camera for the angle corresponding to one column of captured image. If we denote this angle with θ_0, we can write angle θ as:

$$\theta = dx\cdot\frac{\theta_0}{2}, \tag{3}$$

where dx is the absolute value of difference between corresponding points image coordinates on horizontal axis x of the panoramic images.

We are using a procedure called "normalized correlation" to search for corresponding points. To increase the confidence in estimated depth we are using a procedure called "back-correlation" [6]. With the back-correlation we are also solving the problem of occlusions.

2.6. System capabilities analysis

2.6.1. Time complexity of creating a panoramic image

The biggest problem our system is facing is that it can not work in real time since we are creating panoramic images by rotating the camera for a very small angle. Because of mechanical vibrations of the system, we also have to be sure to capture an image when the system is completely still. The time that the system needs to create a panoramic image is much too long, so there is no feasibility to make it work in real time.

In one circle around the system vertical axis our system constructs 11 panoramic images (5 symetrical pairs and a panoramic image from the middle columns of the captured images). It captures 1501 images with resolution of 160×120 pixels, where radius is $r = 30$ cm and the shift angle is $\theta_0 = 0.2^{\circ}$. The process takes a bit more than 15 minutes on a 350 MHz Intel PII PC.

2.6.2. Constraining the search space on the epipolar line

Knowing that the width of the panoramic image is much bigger than the width of the captured image, we would have to search for corresponding points along a very long epipolar line. That is why we would really like to constrain the search space on the epipolar line as much as possible. A side effect is also an increased confidence in estimated depth and a faster execution of the stereo reconstruction procedure.

If we derive from eq. (1), we can ascertain two things which nicely constrain the search space:

1. If θ_0 presents the angle for which the camera is shifted, then $2\theta_{\min} = \theta_0$. This means that we have to make at least one basic shift of the camera to get a scene point projected in a right column of the captured image forming the left eye panorama, to be seen in the left column of the captured image forming the right eye panorama. Based on this fact, we can search for the corresponding point in the right eye panorama starting from the horizontal image coordinate $x + \frac{2\theta_{\min}}{\theta_0} = x + 1$ forward, where x is the horizontal image coordinate of the point in the left eye panorama for which we are searching the corresponding point. Thus, we get the value +1 since the shift for angle θ_0 describes the shift of the camera for one column of the captured image.

2. Theoretically, the estimation of depth is not constrained upwards, but from eq. (1), it is evident that the denominator must be non-zero. We can write this fact as: $\theta_{\max} = n \cdot \frac{\theta_0}{2}$, where $n = \varphi$ div $\frac{\theta_0}{2}$ and φ mod $\frac{\theta_0}{2} \neq 0$.

   If we write the constraint for the last point, that can be a corresponding point on the epipolar line, in analogy with the case of determining the starting point that can be a corresponding point on the epipolar line, we have to search for the

corresponding point on the right eye panorama to including horizontal image coordinate $x + \frac{2\theta_{\max}}{\theta_0} = x + n$. x is the horizontal image coordinate of the point on the left eye panorama for which we are searching the corresponding point.

In the following sections we will show that we can not trust the depth estimates near the last point of the epipolar line search space, but we have proven that we can effectively constrain the search space.

To illustrate the use of the specified constraints on real data, let us write the following example which describes the working process of our system: while the width of the panorama is 1501 pixels, we have to check only $n = 149$ pixels in a case of $2\varphi = 29.9625^{\circ}$ and only $n = 18$ in the case of $2\varphi = 3.6125^{\circ}$, when searching for a corresponding point.

From the last paragraph we could conclude that the stereo reconstruction procedure is much faster for a smaller angle φ. But we will show in the next section that a smaller angle φ does, unfortunately, also have a negative property.

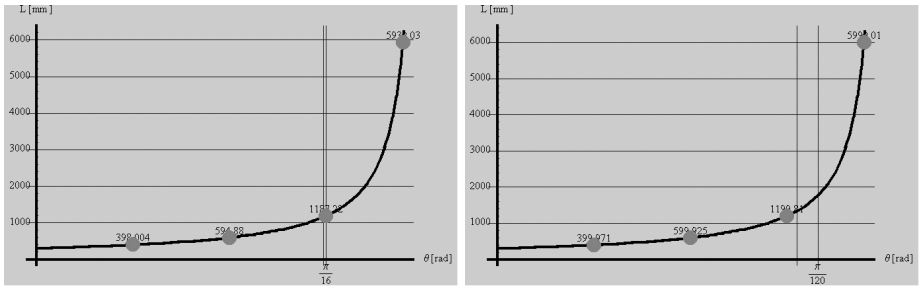
2.6.3. The meaning of the error for a pixel in the estimation of angle θ

Let us first define what we mean by the term 'error' for a pixel. The images are discrete. Therefore, we would like to know what is the value of the error in the depth estimation if we miss the right corresponding point for only a pixel. And we would like to have this information for various values of the angle φ.

Before we illustrate the meaning of the error for a pixel in the estimation of angle θ, let us take a look at the graphs in Figure 3. These graphs are showing a dependence of the depth function l from angle θ while using different values of angle φ. From the graphs it is evident that the depth function l is rising slower in the case of a bigger angle φ. This property decreases the error in depth estimation l when using a bigger angle φ, but this decrease in the error becomes even more evident if we know that the horizontal axis is discrete and the intervals on the axis are $\frac{\theta_0}{2}$ degrees wide (see Figure 3). If we compare the width of the interval on both graphs with respect to the width of the interval that θ is defined on ($\theta \in [0, \varphi]$), we can see that the interval whose width is $\frac{\theta_0}{2}$ degrees, is much smaller when using a bigger angle φ. This subsequently means that the error for a pixel in the estimation of angle θ is much smaller when using a bigger angle φ, since a shift for angle θ_0 describes the shift of the camera for one column of pixels.

Because of discrete horizontal axis θ (Figure 3) with intervals, which are $\frac{\theta_0}{2}$ degrees wide (in our case $\theta_0 = 0.2^{\circ}$), the number of possible depth estimation values is proportional to angle φ: we can calculate $(\varphi \text{ div } \frac{\theta_0}{2} =) 149$ different depth values if we are using the angle $2\varphi = 29.9625^{\circ}$ and only 18 different depth values if we are using the angle $2\varphi = 3.6125^{\circ}$. And this is the negative property of using a smaller angle φ.

Results in table 1 give the values of the error for a pixel in the estimation of angle θ for different values of parameters θ and φ.



a) $2\varphi = 29.9625^{\circ}$ b) $2\varphi = 3.6125^{\circ}$

Figure 3. Graphs showing dependence of depth function l from angle θ while radius $r = 30$ cm and using different values of angle φ. To ease the comparison of the error for a pixel in estimation of angle θ we showed the interval of width $\frac{\theta_0}{2} = 0.1^{\circ}$ between the vertical lines around the third point.

From the results in table 1 we can conclude that the error is much bigger in case of a smaller angle φ than in the case of a bigger angle φ. The second conclusion is that the value of the error is getting bigger when the value of the angle θ is getting closer to the value of the angle φ. This is true regardless of the value of the angle φ. The speed of the reconstruction process is inversely proportional to the accuracy of the process.

By varying the parameters θ_0 and r we are changing the size of the error:

- By increasing the resolution of the captured image we are decreasing the angle θ_0 and subsequently decreasing the rotational angle of the camera between two successively captured images forming the panoramas. For nearly the same factor as we increase (decrease) the resolution of captured image, we decrease (increase) the value of the error Δl, while the reconstruction process takes for nearly the same factor more (less) time to end. We can treat the increase in resolution of the captured image as the sub-pixel accuracy procedure.

- For the same factor that we increase (decrease) radius r, we increase (decrease) the (biggest possible and sensible) depth estimation l and the size of the error Δl. If we vary the parameter r, the process of reconstruction will not be any faster or slower. In practice bigger r means that we can reconstruct bigger scenes (rooms). The geometry of our system is adequate of reconstructing (smaller) rooms and is not suitable for reconstruction of an outdoor scene. This is due to the property of the system: we do not trust in the estimated depth l of far away objects on the scene were the size of the error Δl is too big.

2.6.4. Definition of maximal depth in which we trust

In section 2.6.2 we defined the minimal possible depth estimation $l_{\min}$ and maximal possible depth estimation $l_{\max}$, but we did not write anything about the meaning of the error for a pixel in estimation of angle θ for these two estimated depths. Let

	$\theta - \frac{\theta_0}{2}$	θ	$\theta + \frac{\theta_0}{2}$
l [mm]	394.5	398	401.5
Δl [mm] (error)	3.5		
		3.5	

a) $\theta = \frac{\varphi}{4}$, $2\varphi = 29.9625^{\circ}$

	$\theta - \frac{\theta_0}{2}$	θ	$\theta + \frac{\theta_0}{2}$
l [mm]	372.5	400	431.8
Δl [mm] (error)	27.5		
		31.8	

b) $\theta = \frac{\varphi}{4}$, $2\varphi = 3.6125^{\circ}$

	$\theta - \frac{\theta_0}{2}$	θ	$\theta + \frac{\theta_0}{2}$
l [mm]	2252.9	2373.2	2507
Δl [mm] (error)	120.3		
		133.8	

c) $\theta = \frac{7\varphi}{8}$, $2\varphi = 29.9625^{\circ}$

	$\theta - \frac{\theta_0}{2}$	θ	$\theta + \frac{\theta_0}{2}$
l [mm]	1663	2399.6	4307.4
Δl [mm] (error)	736.6		
		1907.8	

d) $\theta = \frac{7\varphi}{8}$, $2\varphi = 3.6125^{\circ}$

Table 1. The meaning of the error for a pixel in estimation of angle θ (equation (3)).

us examine the size of the error Δl for these two estimated depths: we calculate $\Delta l_{\min}$ as the absolute value of difference between the depth $l_{\min}$ and the depth l for which the angle θ is bigger than angle $\theta_{\min}$ for angle $\frac{\theta_0}{2}$:

$$\Delta l_{\min} = |l_{\min}(\theta_{\min}) - l(\theta_{\min} + \frac{\theta_0}{2})| = |l_{\min}(\frac{\theta_0}{2}) - l(\theta_0)|.$$

Similarly, we calculate the error $\Delta l_{\max}$ as the absolute value of difference between the depth $l_{\max}$ and the depth l for which the angle θ is smaller than angle $\theta_{\max}$ for angle $\frac{\theta_0}{2}$:

$$\Delta l_{\max} = |l_{\max}(\theta_{\max}) - l(\theta_{\max} - \tfrac{\theta_0}{2})| = |l_{\max}(n\tfrac{\theta_0}{2}) - l((n-1)\tfrac{\theta_0}{2})|,$$

where by the variable n we denote a positive number following from the equation: $n = \varphi$ div $\frac{\theta_0}{2}$.

	$2\varphi = 29.9625^{\circ}$	$2\varphi = 3.6125^{\circ}$
$\Delta l_{\min}$	2 mm	19 mm
$\Delta l_{\max}$	30172 mm	81587 mm

Table 2. The meaning of the error (Δl) for one pixel in the estimation of angle θ for the minimal possible depth estimation $l_{\min}$ and the maximal possible depth estimation $l_{\max}$ regarding the angle φ.

In table 2 we gathered the error sizes for different values of angle φ. The results confirm statements in Section 2.6.3. We can also add two additional conclusions:

1. The value of the error $\Delta l_{\max}$ is unacceptably high and this is true regardless of the value of angle φ. This is why we have to sensibly decrease the maximal possible depth estimation $l_{\max}$. This conclusion in practice leads us to define the upper boundary of allowed error size (Δl) for one pixel in the estimation of angle θ and with it, we subsequently define the maximal depth in which we trust.

2. Angle φ always depends upon the horizontal view angle α of the camera (equation (2)). And while the angle α is limited to around 40° considering standard cameras, our system is limited to an angle α when estimating the depth, since in the best case we have: $\varphi_{\max} = \frac{\alpha}{2}$. Thus our system can really be used only for the reconstruction of small rooms.

3. Depth-assisted edge detection

3.1. Scale, size and loss of detail

Theoretically, it could be argued that the application of a two-dimensional filter like the Gaussian to an image representing a three-dimensional scene heralds implicit problems. When a three-dimensional scene is captured using a camera, it is logical to assume that the available detail that the resulting image can yield at any point depends partially on that point's original distance from the lens. The farther the feature, the smaller its representation in the image will be (scale). This difference in scale will make it less likely that a distant feature we are interested in will retain all of its detail after being sampled and quantized. This is illustrated in Figure 4.

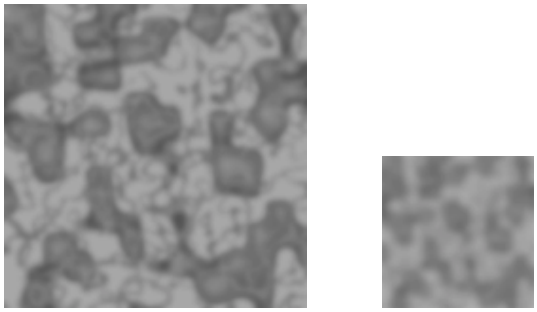


Figure 4. Loss of detail with half-size texture (σ=2).

One can then see that the application of a uniform Gaussian filter with a constant standard deviation σ will cause a disproportionate loss of detail across the image, essentially resulting in low operator performance in the areas of the picture

farthest from the lens. This is more evident in outdoor and synthetic scenes, where depth can vary greatly from point to point. This loss of quality may be countered if the filter's scale can be varied according to its point of application in the image.

An intuitive solution here, then, would be to establish a relation between the filter's scale (controlled by σ), and the real-world size of the object being smoothed. In order to achieve this kind of adjustment at the level of processing that precedes edge detection, one would need to somehow be aware of the relative size of objects at every part of the image. It is known, however, that there is a direct relationship between the size (in pixels) of the representation of an object in an image and that object's original, real-world distance to the camera's lens.

Assuming, therefore, the availability of a robust stereo system, we can compute the distance at any point in the image, thus establishing a variable that will allow us to adjust the Gaussian's σ accordingly.

3.2. Algorithm overview

The algorithm that deals with the problems described above utilizes pre-computed pixel distance information in order to segment the image into several different layers. In this capacity, the algorithm can be considered to be depth-adaptive. These layers are then smoothed independently and are recombined before being fed to an edge-detector. The entire process consists of three main phases, each of which will be presented in detail below.

1. *Layering* — The image is segmented into discrete layers based on pixel distance data and a user-supplied parameter that denotes the desired number of layers to be produced.

2. *Piecewise Smoothing* — The resulting sub-images are smoothed independently, utilizing a filter scale appropriate to the relevant depth of the layer in the image.

3. *Compositing* — The smoothed sub-images are combined into one image again. That image will be the input for the edge-detector.

4. *Edge-detection* — The composited image is fed into an edge-detector.

3.3. Layering

The first step in the realization of this algorithm is to split a given image into multiple layers. Each layer corresponds to a certain range of distances and consists only of image pixels with relevant distance characteristics. The pixels that make up each layer are subsequently supplemented by artificially background-valued pixels in order to form a sub-image that is ready

for smoothing. The algorithm does not currently impose any continuity constraints, and it is not uncommon for a layer to contain non-neighboring pixels.

Note that most trial runs were performed using dense depth maps. To accommodate different measurement schemes, the algorithm detects the minimum and maximum depth values and normalizes all pixel distances to a relative percentage. The pixel(s) with the smallest distance to the camera are given a value of 0 percent, while those farthest from the camera are given a value of 100 percent.

A user-supplied parameter controls the number of layers that will be produced. Steps are being taken to optimize and automate this process, but meanwhile, tests have shown that five layers are often a good choice in synthetic scenes with noticeable depth differences. The process of splitting the original image into various layers is illustrated in Figure 5.

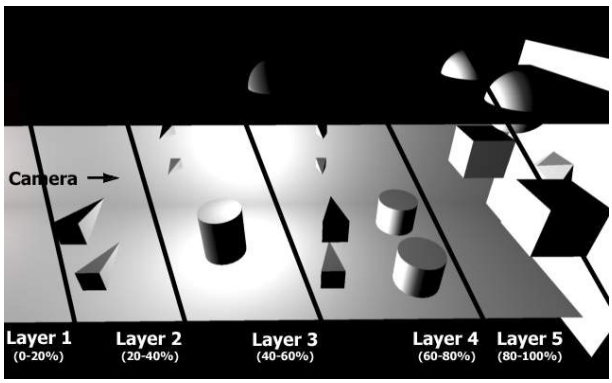


Figure 5. Depiction of layers in a 3D scene.

In the case depicted above, the user has asked for five layers to be produced. The first layer (closest to the camera) will include all pixels with normalized distances of 0 to 20 percent. The second layer will include pixels with distances of 20 to 40 percent, the third 40 to 60 percent, the fourth 60 to 80 percent, and the last layer (farthest from the camera), will include pixels with normalized distances of 80 to 100 percent. Note that these percentages are irrelevant to the size of a layer. The maximum and minimum distances that define a layer are termed $d_{\max}$ and $d_{\min}$ respectively.

Thus, given an image I of height h and width w, the algorithm will produce a set of sub-images:

$$I_s = \{I_1, I_2, \ldots, I_n\},$$

where n is the number of layers as specified by the user-supplied parameter. Each sub-image will retain a height of h and a width of w and will consist of a set of pixels P_n such that:

$$\forall p \in P_n \begin{cases} V_p = V_I, & d_{\max} > D_p > d_{\min} \\ V_p = B, & \text{otherwise,} \end{cases}$$

where p is any pixel in the sub-image, V_p is the value of that pixel in the sub-image, V_I is the value of the same pixel in the original image, D_p is the normalized distance of the pixel in the original image, and B is the background gray value. Hence, a sub-image I_n consists of a number of pixels whose gray values are inherited by the original image, and of a number of "artificial" pixels, which are used in order to make the process of smoothing easier and to avoid the creation of false edges through intensity variations.

3.4. Piecewise smoothing

At this point, the algorithm is ready to apply a Gaussian filter to each of the sub-images that were created during the previous step. Each sub-image should be smoothed using a different σ. In addition, σ should be greatest in the first layer (closest to the camera) and smallest in the last layer (farthest from the camera). Currently, the values of σ for each layer are chosen manually. The algorithm suggests default values after σ has been specified for the first layer, but user intervention is still occasionally required in order to strike a good balance between the elimination of noise and the preservation of detail.

This process should ensure that all sub-images are convolved using a scale appropriate to the pixel-wise size of their contents.

3.5. Compositing

Once the smoothing procedure is finished, the final step is to re-combine the various sub-images into one solid image once more. This resultant image will be the input to the edge-detector. It will essentially be a composite of all the sub-images produced earlier by the layering procedure.

The composition process will result in a final image I_C, in which all original pixels will be substituted by the pixels of the various sub-images, according to the following algorithm.

1. Create a new empty image I_C, with height h and width w.
2. From the set of pre-smoothed sub-images, choose the one with the greatest maximum pixel distance $d_{\max}$ (i.e. the layer farthest from the camera).
3. Substitute all pixels of I_c with all non-artificial pixels of the sub-image.

4. Discard the current working sub-image and repeat steps 1 to 4 until no more sub-images remain.

The above process will ensure that I_c is made up of a number of regions, which have been independently smoothed. The filter scale used in the smoothing of each region will be one that is appropriate to the pixel-wise size of its contents. The visible changes in the amount of smoothing from region to region often vary along the depth discontinuities in the image.

4. Integration and experimental results

4.1. Integration

Although the two systems were developed independently, the integration process was not especially problematic. Essentially, it consisted merely of using real-world data (captured images and depth maps) generated by the panoramic stereo system to drive the depth-assisted edge detection process.

There were two main problems enountered. The most serious of these problems was the limitation of the panoramic stereo system regarding the distances that it can accurately measure. As mentioned above, the depth-assisted edge detection technique is especially useful in images with large depth variations. The images and associated depth maps produced by the panoramic stereo system do not exhibit great depth variations. This, of course, somewhat limits the obvious benefits of the smoothing pre-processor.

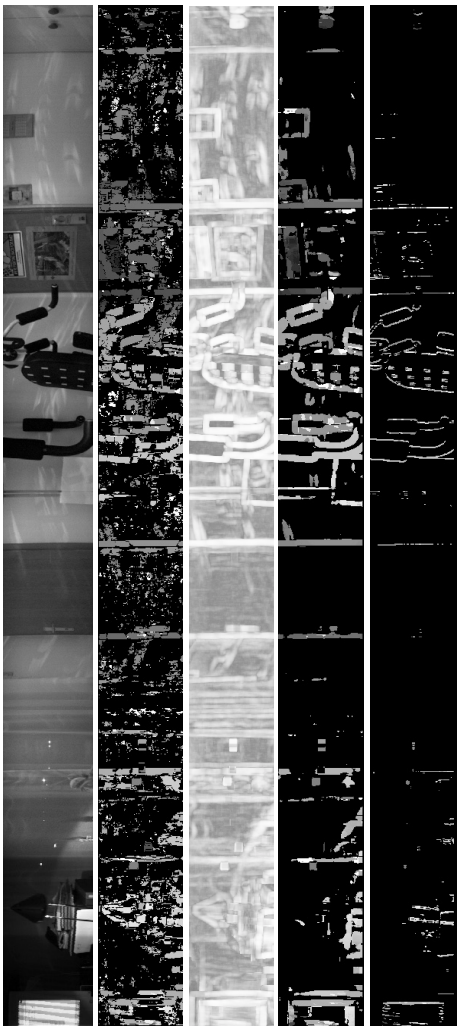
The second problem that was encountered was the non-continuity of the depth map. Due to both noise and the layout of the room, the depth distribution in the test data was extremely hectic and discontinuous. This was in strong contrast to the synthetic scenes, where the depth essentially varied smoothly. This problem was expected, of course, and although the enhancement to the edge detection process was visible, it showcased the need for additional refinement.

Experimental results for both systems are presented below, along with commentary.

4.2. Generation of real world depth images

Figure 6 shows some results of our system. In the case denoted by b), we constructed a dense panoramic image, which means that we tried to find a corresponding point in the right eye panorama for every point in the left eye panorama. Black color marks the points on the scene with no associated depth estimation. Otherwise, the nearer the point on the scene is to the rotational center of the system, the lighter the point appears in the depth image.

In the case denoted by d), we used the information about the confidence in estimated depth (case c)), which we get from normalized correlation estimations. In this way, we eliminate from the dense depth image all the associated depth estimates



a) b) c) d) e)

Figure 6. Some results of stereo reconstruction when creating the depth image for the left eye while angle $2\varphi = 29.9625^{\circ}$: a) left eye panorama, b) dense depth image / using back-correlation / reconstruction time: 6:42:20 (h:m:s), c) the information about the confidence in estimated depth, d) dense depth image after the weighting / not using back-correlation / reconstruction time: 3:21:56, e) sparse depth image / not using back-correlation / reconstruction time: 0:0:38.

which do not have a high enough associated confidence estimation.

In the case marked by e), we created a sparse depth image by searching only for the corresponding points of features on input panoramas. The features can be presented as vertical edges on the scene, which we can derive very fast if we filter the panoramas with the Sobel filter for searching the vertical edges [6, 9]. If we would use a smaller value for angle φ, the reconstruction times would be up to eight times smaller from the presented ones.

All results were generated by using a correlation window of size $2n + 1 \times 2n + 1$, n=4. We searched for corresponding

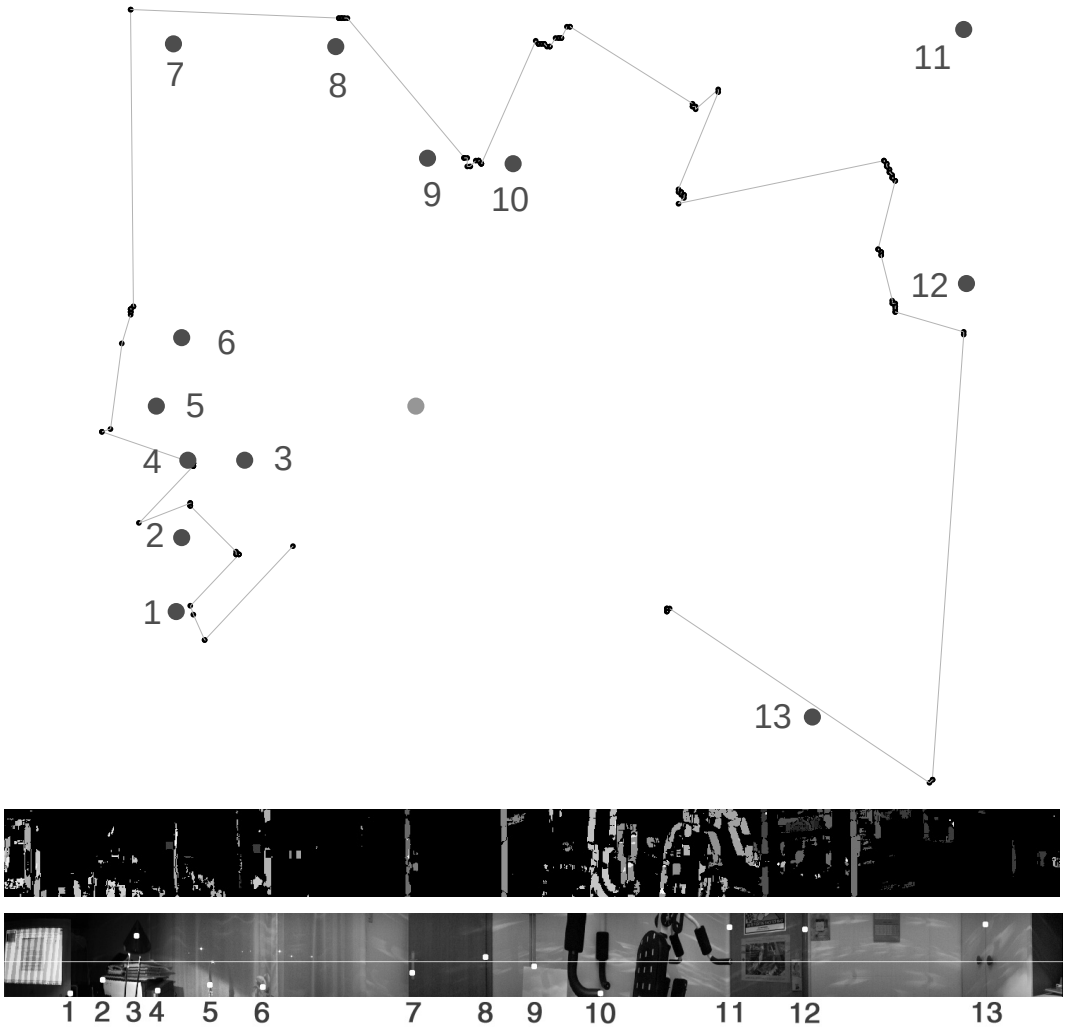


Figure 7. On top is a ground-plan showing the results of the reconstruction process based on the 68th row of the depth image. We used back-correlation and weighting for angle $2\varphi = 29.9625^{\circ}$. The corresponding depth image is shown in the middle picture. For orientation, the reconstructed row and the features on the scene for which we measured the actual depth by hand are shown on the bottom picture. The features on the scene marked with big dots and associated numbers are not necessarily visible in this row.

points only on the panoramic image row which was determined by the epipolar geometry.

Since it is hard to evaluate the quality of generated depth images given in Fig. 6, we will present the reconstruction of the room from generated depth image. Then we will be able to evaluate the quality of generated depth image and consequently the quality of the system.

The result of the (3D) reconstruction process is a ground-plan of the scene. The following properties can be observed on Figure 7:

- Big dots denote features on the scene for which we measured the actual depth by hand.
- Big dot near the center of the reconstruction shows the center of our system.
- Small black dots are reconstructed points on the scene.
- Lines between black dots denote links between two successively reconstructed points.

The result of the reconstruction process based on the 68th row of the depth image when we used back-correlation and weighting is given in Figure 7. Black dots are reconstructed on the basis of estimated depth values, which are stored in the same row of the depth image. The features on the scene marked with big dots are not necessarily visible in the same row.

With respect to the presented reconstruction times we could conclude that the reconstruction procedure could work in nearly real time, if we would work with 8-bit grayscale images (and not with the color images as in our case), with lower resolution and/or if we would create the sparse depth image of only part of the scene. This could be used for robot navigation [9].

4.3. Edge detection on a synthetic image

The algorithm for depth-assisted edge detection was implemented using the architecture described and presented in [5]. A set of sample images is presented below, documenting a trial run of the algorithm on a synthetic image. The entire algorithm has been coded using components. These components were written in Visual Basic 6.0 and Visual C++ 6.0. Pixel distances for the synthetic scene presented in this section were calculated using a custom 3D engine based on DirectX. All pictures and visualizations were produced in bitmap format using Windows GDI functions.

The synthetic scene shown in Figure 8 was created to illustrate how the algorithm presented in this section could be applied in order to preserve necessary detail. There are two “Some Edges” labels in this picture. One is very close to the

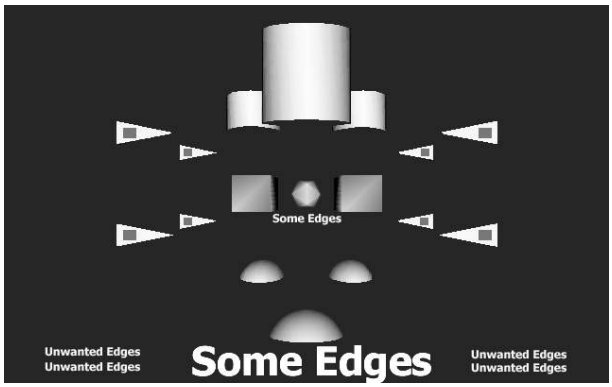


Figure 8. The synthetic 3D scene used for the tests.

camera, while the other is further away from it. These labels represent data that we are interested in. On the other hand, four "Unwanted Edges" labels figure in the image as well. These labels represent noise and unwanted edge data.

The goal of the pre-processor here would be to preserve the details we are interested in (namely the "Some Edges" labels), while attempting to get rid of any noise and unwanted information in the image (namely the "Unwanted Edges" labels).

The standard implementation of the Canny edge-detector does not fare too well in this task. If the σ is too small, the unwanted information persists. As soon as the σ starts getting large enough to partly or fully eliminate the unwanted information, the edge-detector fails to detect relevant detail, which the smoothing filter has obscured (Figure 9).

Our algorithm, on the other hand, begins by reading a z-buffer dump of the synthetic scene. The values read represent the distance of each pixel from the camera. After normalization, the system produces a visualization of the depth map for convenience (Figure 10). 256 gray values are used. Pixels closest to the camera are dark, while those that are further away are light. Notice the barely visible polyhedron in the middle of the far white wall.

The parameter supplied to the layering part of the algorithm is 5. The layers produced are transformed into sub-images and smoothed independently. The composition of these sub-images into a single image follows, and generates the image presented in Figure 11.

Notice how the blurring effect of the filter varies according to the depth of the picture. The parts of the picture that are closest to the camera are blurred the most, but the important details are not affected due to the large scale. On the other hand, the points that are far away from the camera are smoothed less. Detail is preserved. Depending on the distribution of depth in the image, one could achieve a near-perfect balance between noise reduction and detail preservation at every level of depth in the image.

The above is then fed into the edge-detecting module, which produces the edge image presented on Figure 12.

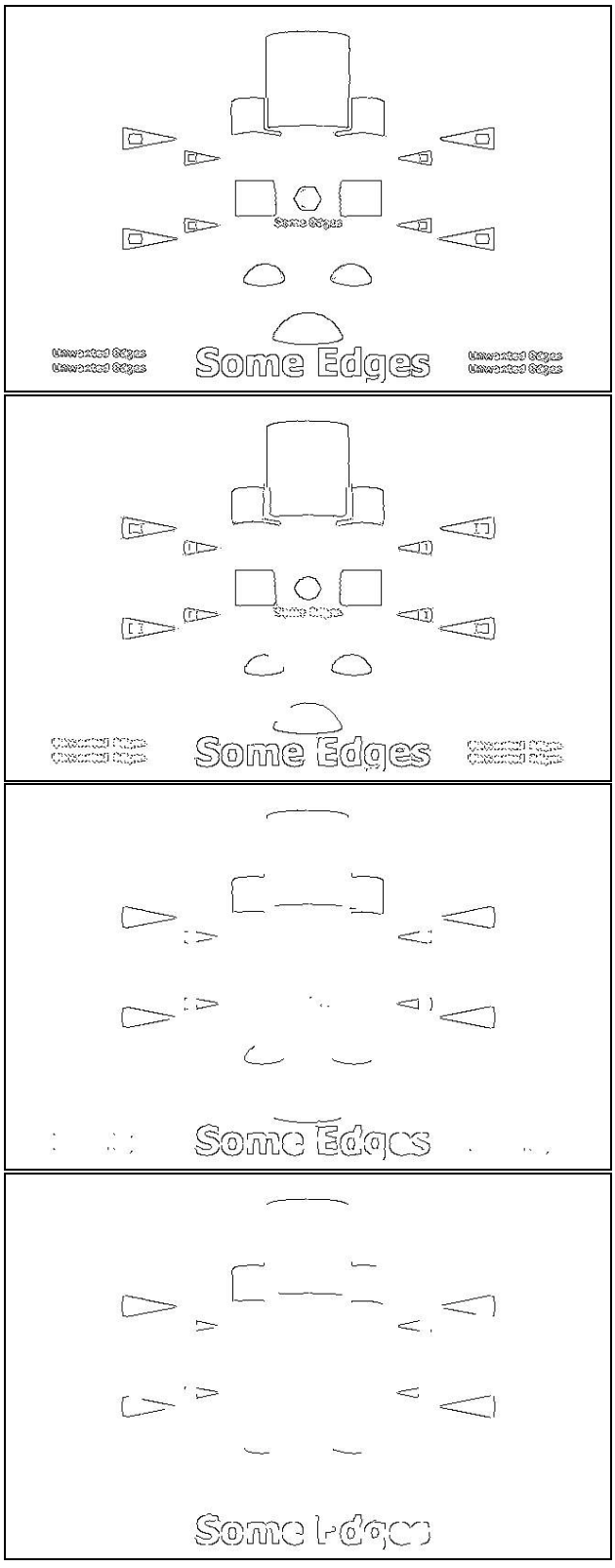


Figure 9. From top to bottom — Canny with σ=1, σ=2, σ=3 and σ=4.

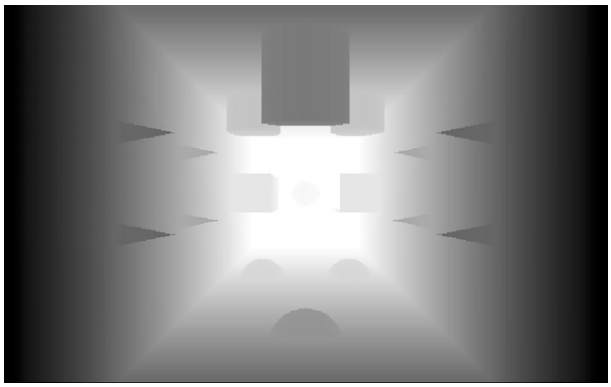


Figure 10. Depth map visualization.

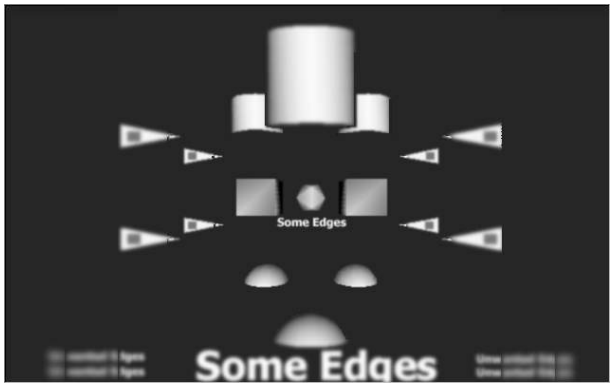


Figure 11. Composited image.

While the unwanted detail has not been eradicated completely, the detail has been preserved properly where the depth is greatest. Notice that the “Some Edges” labels have been changed only slightly. Notice also how the squares inside the pyramids on the left- and right-hand side are well-defined. The polyhedron and the cubes that lie against the far wall also appear quite clearly. Contrast this with the images the Canny produced above with a σ of 3 and 4. Should we wish to do so, we could use a disproportionately large σ in the first layer in order to completely wipe out the “Unwanted Edges” labels. In addition, note that the labels were only used to illustrate a point. In practice, noise would never have the kind of coherency that these letters retain.

4.4. Edge detection on a real world image

Using the same methods outlined above, we applied the enhanced Canny edge detector to real data produced by the panoramic stereo system (Figure 13).

On Figure 14 you can see the depth map that generated by the system, based on the distance data that was provided by the

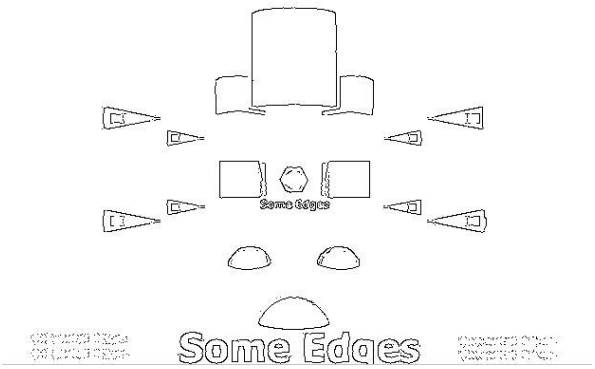


Figure 12. Edge image after scale-based smoothing.

stereo system.

Running an unmodified version of Canny on the image, we can see in the Figure 15 that the edge detector performs pretty well. There are essentially two main problems with the image. The first problem is that the pattern appearing on the monitor on the left-hand side has created some edges that we are probably not interested in. The second problem is that the edge detector has picked up some false edges around what seems to be bright spots. Again, as we vary σ, we see that these imperfections can be culled, but only at the cost of detail in the rest of the image.

As mentioned above, the depth variations in the image were both small and hectic. Thus, it was decided that only two layers would be used, and that the closest layer would be heavily smoothed. Figure 16 shows the result of the Canny edge detector after the image was piped through the smoothing pre-processor. You can see that most of the aforementioned imperfections are gone, while image detail has been preserved.

5. Conclusions

In this article, we presented an analysis of the system for the construction of depth panoramic images using only one standard camera. We proved the following: the procedure for creating panoramic images is very long and can not be executed in real time under any circumstances (using only one camera); epipolar lines of symmetrical pair of panoramic images are image rows; based on the equation for estimation of depth l, we can constraint the search space on the epipolar line; confidence in estimated depth is changing: bigger the slope of the function l curve, smaller the confidence in estimated depth; if we observe only the reconstruction time, we can conclude that the creation of dense panoramic images is very expensive. We can write the essential conclusion as: such systems can be used for 3D reconstruction of small rooms.

A further time reduction in panorama building can be achieved: Instead of building the panorama from only one column

Figure 13. Real data.

Figure 14. Depth map visualization.

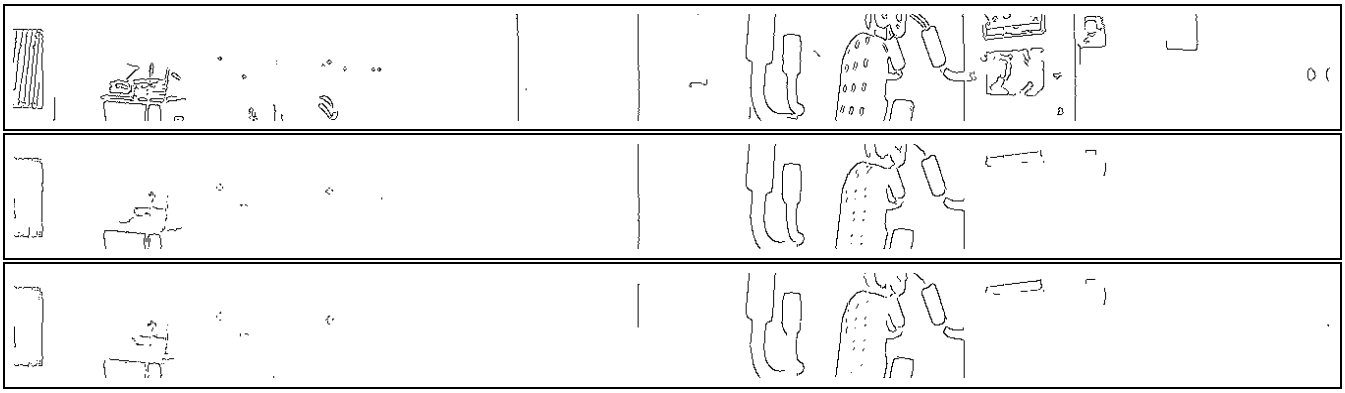


Figure 15. From top to bottom — Canny with σ=1, σ=2, and σ=3.

Figure 16. Edge image after scale-based smoothing.

of the captured image, we could build the panorama from the wider stripe of the captured image, [17]. Thus, we would increase the speed of the building process. If we would use this idea in our system, we know that within the stripe the angle φ is changing. However, the question is how this influences the reconstruction procedure?

The hypothesis that depth information provided by stereopsis (or other mechanisms) may be utilized in order to improve the efficiency of edge detection operators was also presented. In support of this hypothesis, a non-optimal algorithm that alters the pre-processing stage of the Canny edge detector in order to take advantage of pixel distance data was detailed. Through the distance-based segmentation of the input image into different layers, the algorithm that was presented was able to apply custom-fitting smoothing filters to each layer. This allowed the algorithm to eliminate noise, from the image, while at the same time keeping the available overall detail to acceptable constant levels over the entire image. While the algorithm has performed very well on synthetic images with large and continuous depth variations, it still needs a lot of work and refinement for proper operation on real-world images.

While the integration of this algorithm with the aforementioned panoramic stereo system was relatively straightforward, functional limitations specific to the panoramic approach limit this algorithm's usefulness in that context.

The optimization, improvement and the possible redesign of this algorithm are currently subjects of further research. It is hoped that this research will contribute an alternative approach to edge detection — and possibly other low-level image operations — by exposing how stereo data may be used to assist such endeavors.

The next step in pursuing this path of research is to optimize the layering process so that human intervention is not required. Statistical depth distributions are being considered for this purpose. In addition, a lot of work can be performed in automatically determining the proper σ for each layer.

References

[1] I. E. Abdou and W. K. Pratt. Quantitative design and evaluation of enhancement/thresholding edge detectors. In *Proceedings of the IEEE*, volume 67, pages 753–763, 1979.

[2] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[3] S. Chen. Quicktime VR — an image-based approach to virtual environment navigation. In *ACM SIGGRAPH*, pages 29–38, Los Angeles, USA, 1995.

[4] E. R. Davies. *Machine Vision — Theory, Algorithms, Practicalities*. Academic Press, USA, 1997.

[5] A. Economopoulos and D. Martakos. Component-based architectures for computer vision. In *Proceedings of WSCG: International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, number 2, pages 117–125, 2001.

[6] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts, London, England, 1993.

[7] R. Gupta and R. I. Hartley. Linear pushbroom cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):963–975, September 1997.

[8] F. Huang and T. Pajdla. Epipolar geometry in concentric panoramas. Technical Report CTU-CMP-2000-07, Center for Machine Perception, Czech Technical University, Pargue, Czech Republic, March 2000.

[9] H. Ishiguro, M. Yamamoto, and S. Tsuji. Omni-directional stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):257–262, February 1992.

[10] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw-Hill, Singapore, 1995.

[11] R. Klette, K. Schluens, and A. Koschan. *Computer Vision — Three-dimensional Data from Images*. Springer, Singapore, 1998.

[12] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):117–154, 1998.

[13] J. R. Parker. *Algorithms for Image Processing and Computer Vision*. Wiley Computer Publishing, USA, 1997.

[14] S. Peleg and M. Ben-Ezra. Stereo panorama with a single camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 395–401, Fort Collins, USA, 1999.

[15] S. Peleg, Y. Pritch, and M. Ben-Ezra. Cameras for stereo panoramic imaging. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 208–214, Hilton Head Island, USA, 2000.

[16] S. Peleg, B. Rousso, A. Rav-Acha, and A. Zomet. Mosaicing on adaptive manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1144–1154, October 2000.

[17] B. Prihavec and F. Solina. User interface for video observation over the internet. *Journal of Network and Computer Applications*, 21:219–237, 1998.

[18] P. Rademacher and G. Bishop. Multiple-center-of-projection images. In *Computer Graphics (ACM SIGGRAPH)*, pages 199–206, Orlando, USA, 1998.

[19] H. Y. Shum and R. Szeliski. Stereo reconstruction from multiperspective panoramas. In *IEEE 7^{th} International Conference on Computer Vision*, pages 14–21, Kerkyra, Greece, 1999.

[20] R. Szeliski and H. Y. Shum. Creating full view panoramic image mosaics and texture-mapped models. In *Computer Graphics (ACM SIGGRAPH)*, pages 251–258, Los Angeles, USA, 1997.

[21] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice-Hall, USA, 1998.

[22] D. Wood, A. Finkelstein, J. Hughes, C. Thayer, and D. Salesin. Multiperspective panoramas for cel animation. In *Computer Graphics (ACM SIGGRAPH)*, pages 243–250, Los Angeles, USA, 1997.

[23] C. L. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7), 2000.