A Depth Recovery Algorithm Using Defocus Information

Ten-lee Hwang James J. Clark Alan L. Yuille

Division of Applied Sciences
Harvard University
Cambridge, MA 02138

Abstract

Pentland recently proposed an algorithm for sensing scene depth by measuring the amount of defocus of an image. The algorithm is interesting in the sense that no correspondence problems are involved. In this paper[1], we first discuss the difference between shape from defocus and shape from focus. We then present a two-phase algorithm where the defocus process is modeled as a two-dimensional Gaussian point spread function. During the first phase (calibration phase), a camera system parameter is determined off-line. In the next phase (depth-recovery phase), this parameter is used to recover on-line the scene depth by taking two images of the same scene, but with a different amount of defocus. Some implementation issues are addressed and test results on real images are provided.

1 Introduction

Depth perception is a very important low-level task for enabling a mobile robot system to understand the three dimensional relationship of the world space objects. There are many different approaches to solve the depth perception problem, e.g. shape from shading and stereo. Different approaches are based on different assumptions and work best in different situations. In stereo algorithms, finding the corresponding pixels in two images of the scene has been recognized as a difficult problem. Pentland[9] was able to recover the depth by defocussed images without the correspondence problem in stereo. He noticed the fact that most biological lens systems are exactly focused at only one distance along each radius from the lens into the scene. As the distance between the imaged point and the surface of exact focus increases or decreases, the imaged objects become progressively more defocused. In addition, some other researchers [3][4][8][10][11] have used this phenomenon to derive algorithms for recovering depth information.

As shown in Figure 1, the blurring of the image due to defocussing is best described by a point spread function. If the image is almost focused, the point spread function is a square sum of a infinite number of Bessel functions[1][9]. For the far

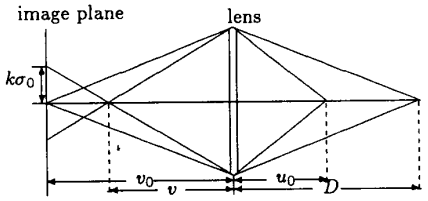

Figure 1: Geometry of Imaging.

field, i.e., far from focus, we would get a sum of an infinite number of some functions again. In the spirit of the central limit theorem, the net effect is almost certainly best described by a two-dimensional Gaussian function $G(r,\sigma_0)$ with a spatial constant σ_0, i.e., $G(r,\sigma_0) = \frac{1}{2\pi\sigma_0^2} e^{-\frac{r^2}{2\sigma_0^2}} = \frac{1}{2\pi\sigma_0^2} e^{-\frac{x^2+y^2}{2\sigma_0^2}}$, where $r^2 = x^2 + y^2$. Nonlinearities, however, may occur as a result of camera imperfections. For the moment, we ignore these effects.

1.1 Pentland's algorithm

Pentland[9] showed that the distance D to an imaged point is related to the parameters of the lens system and the amount of defocus by the following equations:

$$D = \frac{Fv}{v - F - 2k\sigma_0 f}, D > u_0; \quad (1)$$

$$D = \frac{Fv}{v - F + 2k\sigma_0 f}, D < u_0 \quad (2)$$

where v_0 is the distance between the lens and the image plane, F the focal length of the lens system, f the f-number of the lens system($f = \frac{F}{d}$, d the lens diameter) [2], σ_0 the spatial constant of the point spread function, k the proportionality between the blur circle radius and σ_0 (k is assumed constant but it may spatially vary depending on the camera), and where u_0 is the distance between the lens and the position of perfect focus. In the position of perfect focus, σ_0 is zero, so $u_0 = \frac{Fv_0}{v_0 - F}$.

Pentland[9] was able to compute the blur circle radius from a sharp discontinuity. As far as arbitrary scenes are concerned, Pentland proposed another algorithm to find the blur circle ra-

[1]This research was supported by the Center for Intelligent Control Systems Brown/Harvard/MIT through the U.S. Army Research Office grant no. DAAL03-86-K-0171.

[2]We use the conventional definition[6] for f, instead of Pentland's original definition $f = \frac{2F}{d}$[9].

dius by changing the degree of defocus in input images. He employed the Fourier transform in his mathematical derivation, but he simplified his algorithm by using Laplacian and Gaussian filters to estimate local high frequency content in his implementation. In his algorithm, he had to make unnecessary assumptions and constraints about the blur circle radius. Therefore, he could only get a very rough depth estimate on which it is almost impossible to conduct any practically useful error analysis.

With the new algorithm described in the next section, we can obtain a much better depth estimate and we can determine a unique depth estimate from two images obtained by changing v_0.

1.2 Other algorithms

Besides Pentland's algorithm, there are several algorithms based on depth from defocus. The algorithms were proposed from different models of the imaging systems or on different assumptions of the the point spread functions. For instance, Grossmann evaluated the blur measure for image primitives, e.g. edges, and then converted the blur measure to the relative depth[4]; Subbarao and Gurumoorthy also recovered the depth of the blurred edges by a different technique[10]; and Subbarao proposed a general framework for parallel depth recovery[11]. All these algorithms recover the scene depth directly from defocussed images.

On the other hand, there are several algorithms[8] based on depth from focus, rather than depth from defocus. Some of them use active range finding with infrared or sonar sensors; some of them require special hardware, e.g. beam-splitters. If general-purpose cameras are used, the algorithms usually define some criterion function to measure the sharpness of focus, for example, the magnitude of the gradient. Locating the mode of the criterion function would be equivalent to getting the sharpest focus. After the object is in focus, we can then recover the depth of the object. A pyramid based algorithm by Darrell and Wohn[3] can even recover a multi-resolution depth pyramid.

Most active-focusing algorithms are not uniform in the sense that the imaging system has to select a window, focus the object points within the window, and recover the depth only within the small window. Even for uniform algorithms recovering the depth from focus, the imaging system has to take some time to focus some portion of the scene, the corresponding object points might move during the focusing process. However, as one of the depth-from-defocus algorithms, our proposed algorithm is uniform, local, fast, without any tracking problems, and easily implemented on SIMD image processors or parallel machines. As we will see later in the paper, the algorithm gives relatively poor depth estimates. Since the depth recovery of the active-focusing algorithms is pretty accurate (2.5 percent precision[8]), our algorithm can be used as a preprocessing stage of the active-focusing algorithms in choosing the initial search interval of the criterion function. It could also be used to guide a binocular stereo matching process.

2 A Controlled-defocus Algorithm

Assume that a defocused local image patch $E(x,y)$ is projected from a local portion of the scene with a constant depth. It can be expressed as

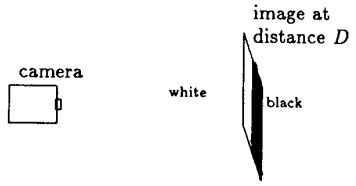

Figure 2: Calibration Setup.

$$E(x,y) = G(r,\sigma_0) \otimes E_0(x,y)$$

where $E_0(x,y)$ is the geometrically ideal projected image patch, $G(r,\sigma_0)$ summarizes all nonideal and defocusing behaviors, and $\otimes$ indicates the convolution operation. σ_0 satisfies equations (1) or (2).

The new algorithm is designed to remedy the major drawbacks in Pentland's algorithm. The algorithm consists of two phases, the k-calibration phase and the depth-recovery phase. The first phase tries to calibrate the system parameter k, which is the proportionality factor between the blur circle radius and σ_0, by analyzing a simple known picture at a known distance. Having determined k off-line, we can then start to recover on-line some part of the depth map from an arbitrary scene.

2.1 Calibration phase

As shown in Figure 2, we put a white sheet of paper in front of the camera at a suitable distance. The white sheet of paper is painted black on one half side. In proper lighting conditions, the ideal projected image is

$$E_0(x,y) = h + \delta, \text{ if } x \geq x_0;$$
$$E_0(x,y) = h, \text{ if } x < x_0$$

Because of blurring, the projected image becomes $E(x,y)$, which is $G(r,\sigma_0) \otimes E_0(x,y)$. Differentiating with respect to x, we obtain

$$\frac{dE(x,y)}{dx} = \delta \frac{1}{\sqrt{2\pi}\sigma_{0x}} e^{-\frac{(x-x_0)^2}{2\sigma_{0x}^2}} \tag{3}$$

Let $I(x,y)$ be the final sensed image right before the uniform quantization. Generally, there is a nonuniform mapping between $E(x,y)$ and $I(x,y)$. Then, $\frac{dE}{dx} = \frac{dE}{dI} \cdot \frac{dI}{dx}$. Substituting this into (3), taking the natural log of both sides, and simplifying the equation, we obtain

$$a(x-x_0)^2 + b = c$$

where $a = -\frac{1}{2\sigma_{0x}^2}; b = \ln \frac{\delta}{\sqrt{2\pi}\sigma_{0x}} - \ln \frac{dE}{dI}; c = \ln \mid \frac{dI}{dx} \mid$.

Because we can only obtain digitally sampled images, $x = \epsilon_x l_x$, where ϵ_x is the horizontal distance between sensing elements on the camera and l_x is an integer. The sharp edge is therefore located approximately at the point, $x = x_0 = \epsilon_x l_{x0}$, where $\frac{dI(x,y)}{dx}$ is maximized. If we assume c follows a normal distribution(this might be due to all kinds of noises in sensors and other electronic devices), then $\sum_{i=1}^{n} \left[a\epsilon_x^2(l_{xi} - l_{x0})^2 + b - c_i\right]^2$ should be minimized[5]. In other words, we can use the least squares

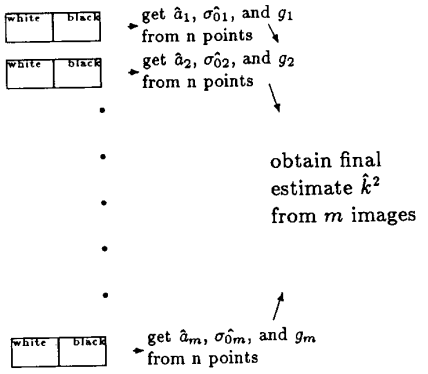

Figure 3: Review of the calibration phase.

method to get the estimator $\hat{a}$:

$$\hat{a} = \frac{\sum_{i=1}^{n}[(l_{xi}-l_{x0})^2 - E[(l_x-l_{x0})^2]]c_i}{\epsilon_x^2 \sum_{i=1}^{n}[(l_{xi}-l_{x0})^2 - E[(l_x-l_{x0})^2]]^2};$$

$$\hat{\sigma_0} = (-2\hat{a})^{-\frac{1}{2}}$$

where $E[(l_x - l_{x0})^2]$ is the average value of $(l_x - l_{x0})^2$, that is, $\frac{\sum_{i=1}^{n}(l_{xi}-l_{x0})^2}{n}$. Note that we do not have to know the exact value of the height δ of the intensity discontinuity and $\frac{dE}{dI}$. That is, we do not have to care about amplitude nonlinearity or companding of the imaging systems.

If the sharp edge were along y direction, then

$$\hat{a} = \frac{\sum_{i=1}^{n}[(l_{yi}-l_{y0})^2 - E[(l_y-l_{y0})^2]]c_i}{\epsilon_y^2 \sum_{i=1}^{n}[(l_{yi}-l_{y0})^2 - E[(l_y-l_{y0})^2]]^2};$$

$$\hat{\sigma_0} = (-2\hat{a})^{-\frac{1}{2}}$$

where $E[(l_y - l_{y0})^2] = \frac{\sum_{i=1}^{n}(l_{yi}-l_{y0})^2}{n}$; $c_i = \ln \mid \frac{dI}{dy} \mid_{y=y_i=\epsilon_y l_{yi}}$; and ϵ_y the vertical distance between sensing elements on the camera.

Referring to Figure 3, we can determine k^2 by a second least squares method as follows: From equations (1) and (2), we obtain

$$\hat{a} = \frac{1}{2\hat{\sigma_0}^2} = k^2 \cdot \frac{4f^2D^2}{-2(D(v_0-F)-Fv_0)^2} = k^2 \cdot g$$

where $g = \frac{2f^2D^2}{-(D(v_0-F)-Fv_0)^2}$.

Again, let us use the least squares method to get a linear estimate for this function. Samples of g_i are obtained by adjusting the distance between lens and the image plane, v_0.

$$\hat{k^2} = \frac{\sum_{i=1}^{m}(g_i - E[g])\hat{a_i}}{\sum_{i=1}^{m}(g_i - E[g])^2}$$

The more experiments we conduct, the larger m is, and the smaller the confidence interval for $\hat{k^2}$ would be.

2.2 Depth-recovery phase

We now show that if we have two images of exactly the same scene, but with a small difference in defocus, obtained by adjusting the distance between the lens and the image plane a little bit, we can recover the scene depth uniquely.

Let $E(x,y)$ be a small projected patch centered at (x_0, y_0). If the imaging system is completely ideal, i.e., without any of the defocus or nonideal behaviors described above, $E_0(x,y)$ would be obtained. That is, $E(x,y) = G(r,\sigma_0) \otimes E_0(x,y)$. Clearly, we would like to devise an algorithm with the following two properties. First, it factors out the contribution of the scene to the blurring images and measures directly the defocus, i.e., σ_0. Second, it is easily implemented and has small expectation of errors. Pentland chose to use the Fourier transform in his algorithm. We propose the following differential algorithm in the spatial domain.

In order to reduce the noise, we have to smooth the input images first. Let us use a 2-D Gaussian window $G(r,\sigma_s)$ to smooth the input images. This is justified because the Gaussian function minimizes the product of spread in the space and in the frequency domain and is easily implemented. Because different small patches in the image result from different local portions of the scene which might be in different depths, the σ_0 at different image patches over the input images might be different. If we use a larger patch size, we assume that the local portion of the scene with a constant depth is larger. Notice that the σ_0 is assumed to be constant over the image patch whereas the σ_s of the smoothing filter is a fixed constant used all over the whole input images. Therefore, the smoothed input image patch becomes

$$E_s(x,y) = G(r,\sigma) \otimes E_0(x,y) \tag{4}$$

where $\sigma^2 = {\sigma_0}^2 + {\sigma_s}^2$. Another way to smooth the input images is to average N input images taken when all camera parameters and the scene are fixed. That is, $E_s(x,y) = G(r,\sigma) \otimes \{\frac{1}{N}\sum_{i=1}^{N} E_0(x,y)|_{t=t_i}\}$.

In typical imaging systems, the F is fixed whereas the v_0 and f can be changed by turning the respective rings on the camera. Let us also fix f in the system setup. In other words, v_0 is the only changeable parameter in this phase. Taking derivatives with respect to v_0 on both sides of (4), we obtain

$$\frac{dE_s}{dI_s}\frac{dI_s}{dv_0} = (\sigma\frac{d\sigma}{dv_0})\nabla^2 G(r,\sigma) \otimes E_0 \tag{5}$$

Taking ∇^2 on both sides of (4), we obtain $\frac{dE_s}{dI_s}\nabla^2 I_s + \frac{\partial I}{\partial x}\frac{\partial}{\partial x}(\frac{\partial E_s}{\partial I_s}) + \frac{\partial I}{\partial y}\frac{\partial}{\partial y}(\frac{\partial E_s}{\partial I_s}) = \nabla^2 G(r,\sigma) \otimes E_0$. $\frac{\partial E_s}{\partial I_s}$ is almost spatially constant, so $\frac{\partial}{\partial x}(\frac{\partial E_s}{\partial I_s})$ and $\frac{\partial}{\partial y}(\frac{\partial E_s}{\partial I_s})$ are approximately zero. Therefore,

$$\frac{dE_s}{dI_s}\nabla^2 I_s = \nabla^2 G(r,\sigma) \otimes E_0 \tag{6}$$

If $\nabla^2 I_s(x_0,y_0)$ is not zero, dividing (5) by (6) at (x_0,y_0), we obtain $\frac{\frac{dI_s(x_0,y_0)}{dv_0}}{\nabla^2 I_s(x_0,y_0)} = \sigma\frac{d\sigma}{dv_0}$. Let $t(x_0,y_0)$ be $\frac{\frac{dI_s(x_0,y_0)}{dv_0}}{\nabla^2 I_s(x_0,y_0)}$ if $\nabla^2 I_s(x_0,y_0)$ is not zero, then

$$t(x_0,y_0) = \sqrt{{\sigma_s}^2 + {\sigma_0}^2}\frac{d}{dv_0}\sqrt{{\sigma_s}^2 + {\sigma_0}^2} = \sigma_0\frac{d\sigma_0}{dv_0} \tag{7}$$

So we do not need to know the standard deviation, σ_s, of the smoothing Gaussian function. It is cancelled out in (7). As pointed out in [8], the Laplacian operations demonstrate a significant vulnerability to noise in the sampled intensity values. This is another reason why we smooth the input images, especially with the Gaussian smoothing. Thus,

$$t(x_0, y_0) = \frac{\frac{dI_s(x_0,y_0)}{dv_0}}{\nabla^2 I_s(x_0, y_0)} = \sigma_0 \frac{d\sigma_0}{dv_0} \tag{8}$$

This is the key equation for recovering the scene depth. Suppose we have two input smoothed projected images, say $E_{s1}(x, y)$ and $E_{s2}(x, y)$, obtained from the same scene but with a small different amount of defocus, i.e., associated with different v_0s, say v_{01} and v_{02}. For corresponding sensed image patches of E_{s1} and E_{s2}, say $I_{s1}(x, y)$ and $I_{s2}(x, y)$, centered at the position (x_0, y_0), the equation (8) can be approximated by

$$t(x_0, y_0) \approx \frac{\frac{I_{s1}(x_0,y_0) - I_{s2}(x_0,y_0)}{v_{01} - v_{02}}}{0.5 \cdot (\nabla^2 I_{s1}(x_0, y_0) + \nabla^2 I_{s2}(x_0, y_0))} \tag{9}$$

The larger $\nabla^2 I_{s1}(x_0, y_0)$ and $\nabla^2 I_{s2}(x_0, y_0)$ are, the more textured the image would be, and the more reliable the recovered depth map would be. From (1) and (2), we obtain

$$\sigma_0 \frac{d\sigma_0}{dv_0} = \frac{(v_0 - F)D^2 + F(F - 2v_0)D + F^2 v_0}{4f^2k^2D^2} \tag{10}$$

Substituting (8) in (10) and simplifying, we can obtain a quadratic equation, $(v_0 - F - 4f^2k^2t)D^2 + F(F - 2v_0)D + F^2v_0 = 0$, where $v_0 \approx 0.5 \cdot (v_{01} + v_{02})$. The roots to this equation are

$$D = \frac{F(2v_0 - F) \pm F\sqrt{F^2 + 16v_0f^2k^2t}}{2(v_0 - F - 4f^2k^2t)} \tag{11}$$

These two roots are positive if t is within a finite interval $(-M_l, M_u)$. One of them is close to F, the focal length. The other root(the larger one) is the scene depth associated with the pixel (x_0, y_0). Considering the larger root, if $0 < t < M_u$, $D > \frac{Fv_0 + F(v_0 - F) + F^2}{2(v_0 - F)} = u_0$; if $-M_l < t < 0$, $D < \frac{Fv_0 + F(v_0 - F) + F^2}{2(v_0 - F)} = u_0$.

This can be intuitively justified as follows. Suppose t is positive, then σ_0 increases as v_0 increases, by the equation (8). That is, the image patch gets more blurred(away from the perfect focus) as v_0 increases. However, u_0 decreases as v_0 increases. Therefore, D must be larger than u_0. The reasoning is similar for the case when t is negative.

If the camera is allowed to move, i.e., D is also a variable parameter in this phase. Then we can fix v_0 and take derivatives with respect to D on both sides of (4). Similarly, let $h(x_0, y_0)$ be $\frac{\frac{dI_s(x_0,y_0)}{dD}}{\nabla^2 I_s(x_0,y_0)}$ assuming $\nabla^2 I_s(x_0, y_0)$ is not zero, then

$$h(x_0, y_0) \approx \frac{\frac{I_{s1}(x_0,y_0) - I_{s2}(x_0,y_0)}{D_1 - D_2}}{0.5 \cdot (\nabla^2 I_{s1}(x_0, y_0) + \nabla^2 I_{s2}(x_0, y_0))} \tag{12}$$

$$h(x_0, y_0) = \sigma_0 \frac{d\sigma_0}{dD} = \frac{v_0F(v_0 - F)D - v_0^2F^2}{4f^2k^2D^3} \tag{13}$$

If $h = 0$, then $D = D_0 = \frac{v_0F}{v_0 - F}$, where D_0 is the distance between the lens and the point of perfect focus. If $h \neq 0$, by simplifying (13), we can obtain a cubic equation, $D^3 + \alpha D + \beta = 0$, where $\alpha = -\frac{v_0F(v_0 - F)}{4hf^2k^2}$ and $\beta = \frac{v_0^2F^2}{4hf^2k^2}$. Let H be $\frac{\beta^2}{4} + \frac{\alpha^3}{27}$, then $H = v_0F(\frac{v_0F}{4hf^2k^2})^2[\frac{v_0F}{4} - \frac{(v_0 - F)^3}{108hf^2k^2}]$.

If $h > M_h = \frac{(v_0 - F)^3}{27f^2k^2v_0F}$ or $h < 0$, then $H > 0$. There will be one real root and two conjugate complex roots. The only real root $D = (-\frac{\beta}{2} + \sqrt{H})^{\frac{1}{3}} + (-\frac{\beta}{2} - \sqrt{H})^{\frac{1}{3}} < 0$. If $M_h \geq h > 0$, then $0 \geq H$. There will be one negative root and two positive roots. The positive roots are

$$D_1 = 2((\frac{\beta}{2})^2 - H)^{\frac{1}{6}} \cos[\frac{1}{3}(\pi - \tan^{-1}\frac{2\sqrt{-H}}{\beta})]$$

$$D_2 = ((\frac{\beta}{2})^2 - H)^{\frac{1}{6}} (\sqrt{3}\sin[\frac{1}{3}(\pi - \tan^{-1}\frac{2\sqrt{-H}}{\beta})] - \cos[\frac{1}{3}(\pi - \tan^{-1}\frac{2\sqrt{-H}}{\beta})]). \tag{14}$$

In particular, if $h = M_h$, then $H = 0$ and $D_1 = D_2 = (\frac{\beta}{2})^{\frac{1}{3}} = 1.5D_0$. Therefore, we can recover the depth of the scene patch only when the patch depth is farther away from the point of focus. Also, except at the point $D = 1.5D_0$, the depth recovery is not unique.

3 Implementation

The experiments were conducted in the Harvard Robotics Laboratory. We obtained images from a Panasonic solid state CCD camera, Model WV-CD50 with 50 mm focal length, adjustable v. The camera is actually the left eye of the the Harvard head[2]. The diameter of the lens is 2.3 cm, so $f = \frac{5.0}{2.3} \approx 2.17$.

3.1 Sensitivity Analysis

Before implementation, it is desirable to conduct a sensitivity analysis to have an idea how measurement errors and noise will affect experimental results. This analysis will guide the experimental setup and will provide crucial explanations about experimental results. It is also useful when this algorithm is fused with other depth-recovery algorithms.

Sensitivity analysis of $2k\sigma_0$, for the calibration phase: From equations (1) and (2), $2k\sigma_0 = \pm\frac{Dv_0 - DF - Fv_0}{Df}$. Differentiating, if we assume D, f, F and v_0 are independent variables, we get $\delta(2k\sigma_0) = C_1\delta D + C_2\delta f + C_3\delta F + C_4\delta v_0$. For the experimental setup, F and f are fixed and D and v_0 are controllable variables. We plot out values of C_1, C_2, C_3, and C_4 with respect to D and v_0 in [7]. One good experimental setup would be $D = 290$ cm and $v \approx 5.18$.

Sensitivity analysis of D, for the depth-recovery phase: As the graphs in [7] show, the first algorithm where v_0 gets changed is sensitive to the noise at depths around or larger than the depth of exact focus and the second algorithm where D gets changed is not as accurate as the first one. The recovered depth of both algorithms is sensitive to k.

3.2 Asymmetric Laplacian template

The scanning area of the WV-CD50 camera is 8.8 mm ×6.6 mm. Because the digital images we get in the laboratory are of 510 × 492, $\epsilon_x = \frac{0.88}{510} = 00.0017$ cm and $\epsilon_y = \frac{0.66}{492} = 0.0013$ cm. Due to the difference between the resolutions along the horizontal and vertical directions, the Laplacian template is not symmetric.

Following the notation in subsection 2.1, $\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = \frac{1}{\epsilon_x^2}\frac{\partial^2 I}{\partial l_x^2} + \frac{1}{\epsilon_y^2}\frac{\partial^2 I}{\partial l_y^2}$. Define $\rho = \frac{\epsilon_x^2}{\epsilon_y^2}$, then $\nabla^2 I = \frac{1}{\epsilon_x^2}[\frac{\partial^2 I}{\partial l_x^2} + \rho\frac{\partial^2 I}{\partial l_y^2}]$. The Laplacian template would be

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & \rho & 0 \\ 0 & -2\rho & 0 \\ 0 & \rho & 0 \end{bmatrix} = \begin{bmatrix} 0 & \rho & 0 \\ 1 & -2-2\rho & 1 \\ 0 & \rho & 0 \end{bmatrix}$$

multiplied by $\frac{1}{\epsilon_x^2}$.

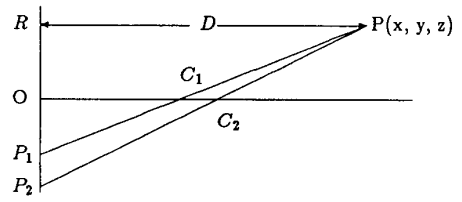

Figure 4: Pixel-alignment of two images with different v_0

3.3 Pixel-alignment of two images

To have an accurate measurement on $v_{01} - v_{02}$ in calculating $t(x_0, y_0)$ in (9), $v_{01} - v_{02}$ must be sufficiently large. However, changing v_0 expands or shrinks the input images. That is, image patches $I_1(x_0, y_0)$ and $I_2(x_0, y_0)$ might not correspond to the same local scene. This geometric image distortion can be restored by the following simple scheme. Because the change in v_0 is small, moving the image plane and fixing the pinhole is almost equivalent to fixing the image plane and moving the pinhole. As shown in Figure 4, C_1 and C_2 are the moving pinholes corresponding to v_{01} and v_{02} respectively. If O is the intersection of the line $\overline{C_1C_2}$ and the image planes, then $\overline{OC_1} = v_{01}$ and $\overline{OC_2} = v_{02}$. For a point P in the world space at a distance D from the image planes, let R be the orthographic projection of P in the image planes. P_1 and P_2 are the perspective projection of P in the image planes.

Assume that $\overline{OP_1} = x_1$, $\overline{OP_2} = x_2$, and $\overline{OR} = L$. By the similar triangles, we obtain $\frac{L+x_1}{D} = \frac{x_1}{v_{01}}$ and $\frac{L+x_2}{D} = \frac{x_2}{v_{02}}$. Simplifying, $\frac{x_1}{x_2} = (\frac{D-v_{02}}{D-v_{01}})(\frac{v_{01}}{v_{02}})$. If $v_{01} \ll D$ and $v_{02} \ll D$, we obtain $\frac{x_1}{x_2} \approx \frac{v_{01}}{v_{02}}$. Therefore, the ratio $\frac{v_{01}}{v_{02}}$ can be used to shrink the larger image to align corresponding pixels.

If we change D instead of v_0 in the depth-recovery phase, by a similar geometric reasoning, $\frac{L+x_1}{D+\delta D} = \frac{x_1}{v_0}$ and $\frac{L+x_2}{D} = \frac{x_2}{v_0}$, where δD is the depth change. Simplifying, $\frac{x_1-x_2}{x_1} = -\frac{\delta D}{D-v_0}$. If $D \gg v_0$ and $D \gg \delta D$, then $x_1 \approx x_2$. There is basically no need to conduct the pixel alignment.

However, $\frac{\delta(k\sigma_0)}{k\sigma_0} = \frac{Fv_0}{D(v_0-F)-Fv_0}\frac{\delta D}{D}$ and $\delta(k\sigma_0) = \frac{Fv_0}{2fD}\frac{\delta D}{D}$. In the current experimental setup, $\frac{\delta(k\sigma_0)}{k\sigma_0} \approx \frac{26.0}{0.2D-26.0}\frac{\delta D}{D}$ and $\delta(k\sigma_0) \approx \frac{6.5}{D}\frac{\delta D}{D}$. Therefore, to obtain sufficient changes in the degree of defocus of the input images, i.e., sufficiently large $\frac{\delta(k\sigma_0)}{k\sigma_0}$ or $\delta(k\sigma_0)$, $\frac{\delta D}{D}$ has to be large enough. In other words, when we change the D, the input images are geometrically zoomed before we see different degrees of defocus on the input images. Unless we use specially designed cameras, e.g. with large F and v_0, small f, some other compensation procedures have to be done before we employ the second algorithm. Darrell and Wohn proposed one of the zoom compensation methods in [3].

3.4 Implementation results

The main assumption of the proposed algorithm is that the point spread function due to defocus is a 2-D Gaussian function. We did an experiment to check this[7] and found that it is a good approximation for WV-CD50 cameras. H. Lee also studied the point spread functions derived from the wave optics and the geometric optics recently[9].

In the calibration phase, the horizontal or vertical sharp edge is set up 290 cm away from the Harvard head. The Harvard head can be programmed to moved horizontally and vertically. This made the calibration of the system parameter k on the whole image plane easy. We actually did 15 experiments on a vertical edge and 15 on a horizontal one, $v_0 = 5.21, 5.17, 5.12$ (five each). As a result[7], k ranges from 0.95 to 1.35 spatially. The fact that k is close to 1 justifies the initial assumption made by Pentland in [10]. Also, the k values on the image grid are quite random, not following any special pattern, e.g. circular symmetry.

The second depth-recovery algorithm is not as accurate as the first one (from the theoretical sensitivity analysis) and is not able to provide a unique depth value. Besides that, it also needs zoom distortion compensation, so we have not done experiments to investigate it yet.

Figure 5 represents one set of images and the thresholded depth maps obtained by the first depth-recovery algorithm(changing v_0). The upper-left image, **im1**.1, is taken at $v_0 = 5.21$, the upper-right one, **im1**.2, at $v_0 = 5.18$. The scene consists of two objects at distances around 100 cm and 150 cm from the camera, respectively. The farther object is at the left of the images, whereas the nearer object is at the right of the images. Based on the camera we were using, this configuration was a good tradeoff between the calibration accuracy of the perfect focus and the sufficiently large different amount of defocus. Texture edges in the farther object(left part of the images) in **im1**.2 are sharper than in **im1**.1, because the depth of perfect focus is 127 cm in **im1**.1, and 148 cm in **im1**.2. One the other hand, texture edges in the nearer object(right part of the images) in **im1**.1 seem sharper than in **im1**.2, but they are not as obvious as those in the farther objects.

The bottom-left image, **dep1**.120, is thresholded from the smoothed depth map where new pixel values are averages of all pixel values, within a 5×5 neighborhood, but different from 136 cm. We obtain the depth map by assuming $k = 1.1$. The black points in the this image **dep1**.120 are the pixels with recovered depth values smaller than 120. Notice that we deliberately set 120 to extract nearer points. Similarly, the black points in the bottom-right image, **dep1**.145, are the pixels with recovered depth values greater than 145 which is larger than 136. Most of the black points in the bottom-left image are on the right and most of the black points in bottom-right are on the left. That is, the texture edges on the right parts of the images have nearer depth values than those on the left parts.

Referring to [7], experiments showed that the distribution of the recovered depth for the nearer object skewed to 136 cm and the depth recovery at the points away(nearer or farther) from the focus is less accurate. Although we only obtained rough depth estimates, the results are still impressive, given that the camera is very noisy and nonideal and the readings of the adjustable

f-number and v in the camera are very rough.

One might want to take more than two pictures sequentially at different v_0's, or even at different depths(by moving the camera) in order to obtain a better depth estimate. Combining information from the two depth-recovery algorithms in a clever way could also be used to disambiguate the depth recovery in the second algorithm.

4 Conclusion

Assuming that the point spread function of the defocused images is a Gaussian function, we have shown that the depth of the scene can be recovered uniquely from two different images obtained by controlled defocus. The proposed depth-recovery algorithm consists of two parts, i.e., the calibration phase and the depth-recovery phase. In the first phase, the imaging system parameter k is calibrated off-line. Once k is calibrated, we can recover any scene depth by adjusting the amount of defocus. The depth-recovery algorithm is local, therefore easily implementable on SIMD parallel machines. If we know more about the imaging system, e.g. its nonlinearities, the algorithm can be improved.

Psychophysical experiments show that the stereopsis does indeed provide us a better three dimensional perception than accommodation does. Feature-based stereo algorithms often use some assumptions, e.g. figural continuity and ordering constraints, to solve the correspondence problem. This kind of heuristic may fail in certain cases. The theoretical depth errors due to accommodation are comparable to those from stereopsis and lack correspondence problems[10]. Therefore, we can use the monocular cue from this algorithm and other depth-recovery algorithms to attack the correspondence problem in stereo. In addition, the algorithm can provide a global depth estimate that traditional focusing algorithms can use to shorten the focusing time at some specific point in the scene.

5 Acknowledgement

The first author would like to thank Professor Roger Brockett for his constant support and encouragement.

References

[1] M. Born and E. Wolf, *Principles of Optics*, p435-449, Pergamon Press, 1965.

[2] J. Clark and N. Ferrier, *Modal Control of an Attentive Vision System*, the Second ICCV, 1988.

[3] T. Darrell and K. Wohn, *Pyramid Based Depth from Focus*, the IEEE CVPR, 1988.

[4] P. Grossmann, *Depth from focus*, Pattern Recognition Letters, Vol. 5, p 63-69, January 1987.

[5] R. Hogg and A. Craig, *Introduction to Mathematical Statistics.*

[6] B. K. P. Horn, *Robot Vision*, 1986.

[7] T. Hwang, J. Clark, and A. Yuille, *A Depth Recovery Algorithm Using Defocus Information*, TR, no. 89-2, Harvard Robotics Laboratory, 1989.

[8] E. Krotkov, *Focusing*, International Journal of Computer Vision, vol. 1, 1987.

[9] H. Lee, unpublished manuscript.

[10] A. Pentland, *A New Sense for Depth of Field*, IEEE Transactions on PAMI, July 1987.

[11] M. Subbarao, *Parallel Depth Recovery by Changing Camera Parameters*, the Second ICCV, 1988.

[12] M. Subbarao and N. Gurumoorthy, *Depth Recovery from Blurred Edges*, the IEEE CVPR, 1988.

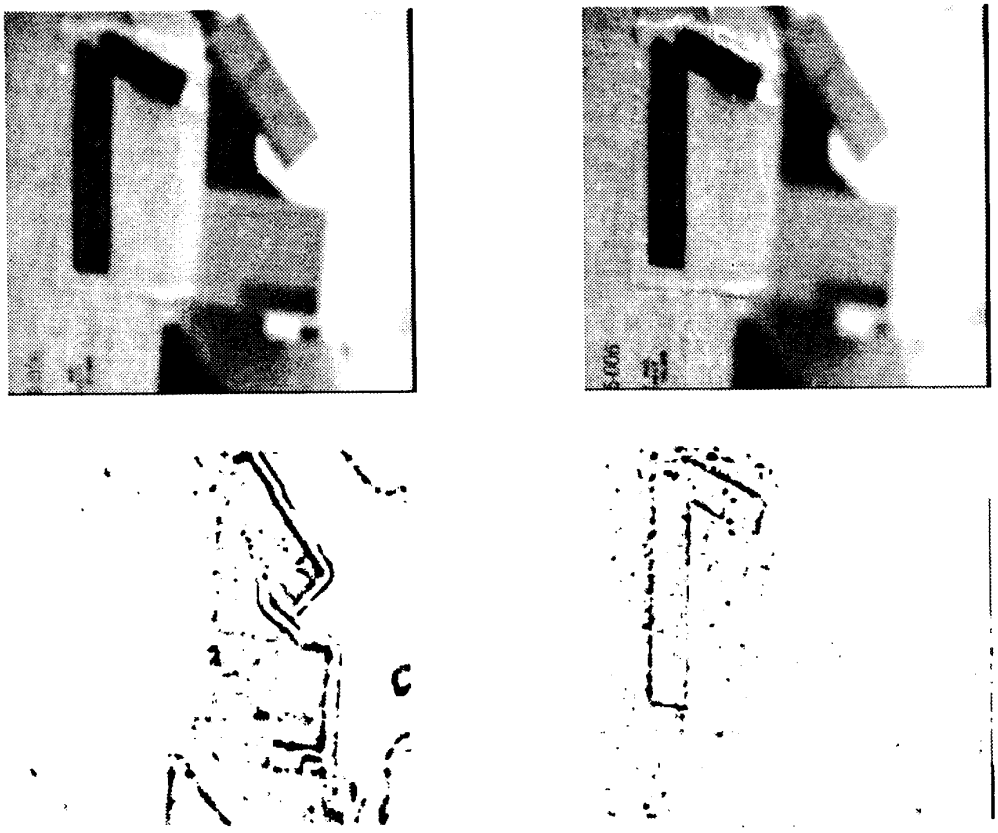

Figure 5: A two-object scene: the farther object is on the left, 150 cm away and the nearer object is on the right, 100 cm away. (a) top-left image: **im1.1**, taken at $v_0 = 5.21$; (b) top-right image: **im1.2**, taken at $v_0 = 5.18$; (c) bottom-left image: **dep1.120**, plotting all the points with recovered depth less than 120 cm; (d) bottom-right image: **dep1.145**, plotting all the points with recovered depth larger than 145 cm.