Detection of Specularities in Color Images Using Local Operators

by

Paul N.E. Pellicano

BSc, University of British Columbia, 1981

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY

December 1985

Approval

Name: Paul N.E. Pellicano

Degree: Master of Science

Title of Thesis: Detection of Specularities in Color Images Using Local Operators

Chairman: Arthur L. Liestman

Brian V. Funt
Senior Supervisor

Thomas W.G. Calvert

Binay K. Bhattacharya

James P. Delgrande
External Examiner

4 December 1985
Date Approved

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

Detection of Specularities in Color Images

Using Local Operators

Author: _____
(signature)

Paul N.E. Pellicano
(name)

16 December 1985
(date)

Abstract

Areas of spectral reflectance, or *highlights*, can be analyzed for a wide range of information or clues that they give us about a scene. This thesis presents a local algorithm for analyzing a moderately unconstrained color image to determine the areas of spectral reflectance. The algorithm is based on the separability of the diffuse and spectral reflection components by differential methods.

The location of specular reflectances are marked by finding zero–crossings in concave down regions for two–dimensional arrays of intensities representing the color image. These zero–crossings correspond to the centers of the highlight regions. The highlight centers are then expanded to highlight regions by region growing in a direction orthogonal to the local orientation of the highlight. Thus, at the conclusion of the algorithm, the information known about each highlight includes location, size and direction.

Acknowledgements

I would like to acknowledge the support and assistance of my supervisor, Dr. Brian Funt and the helpful criticisms of Kim Adamson–Sharpe. Nedenia Holm digitized the images contained in this thesis and many others. This research was funded by Natural Sciences and Engineering Research Grant A4322.

Table of Contents

List of Figures

Chapter 1

Introduction

Computer Vision is often perceived as something that should be trivial. The reason for this perception is that we are ourselves so good at vision, we take the whole process of vision for granted. In fact, the interpretation of our three dimensional world, as portrayed in a two dimensional array of intensities, is anything but trivial. While humans bring a vast amount of 'intrinsic' information to bear on the problem of image analysis, the computer does not have the capacity, at the current time, to perform the same feat. Therefore, to permit any useful analysis of an image whatsoever, we tend to limit, or constrain our image world such that analysis becomes feasible with respect to the limited amount of knowledge we can impart to the computer.

A particularly useful and efficient task that we practice every day, however unwittingly, is that of distinguishing between objects made from different materials. An important prerequisite for such perception is the ability to discern the quality of an object's appearance. Various qualities of appearance are apparent in the world around us, such as texture, color, shine, luster, etc., all of which give us important clues as to an object's composition. This thesis will concern itself with the quality of surface gloss.

Glossiness, in general, is correlated with specular reflectance [Beck 72]. By

looking at figure 1–1 we can easily determine which objects are shiny by the presence of areas of spectral reflectance.

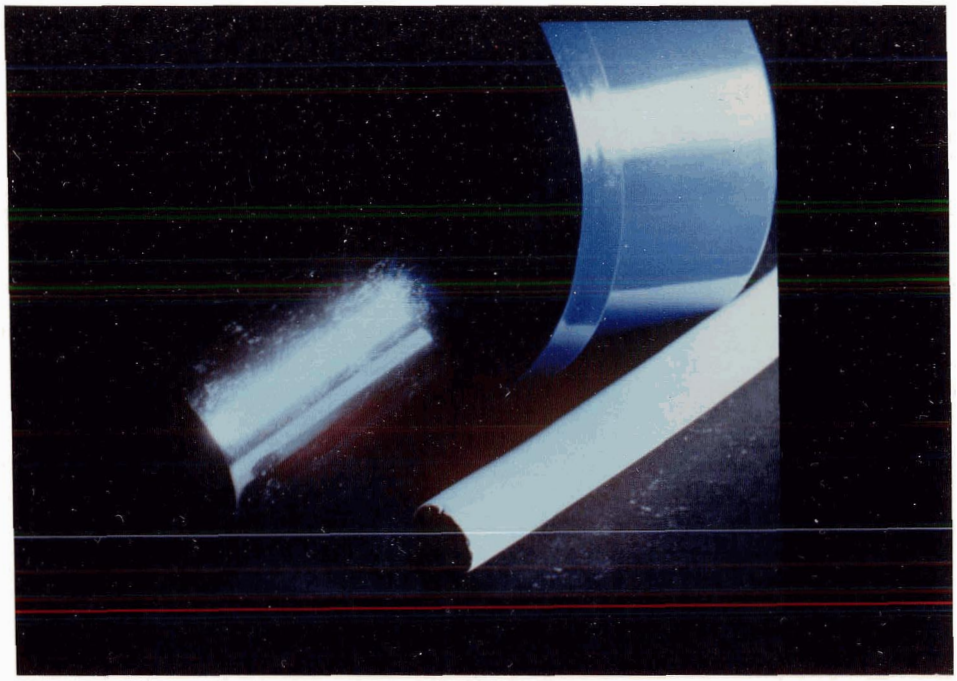


Figure 1-1: Objects with different reflection properties

Surface sheen, shine, gleam, etc., (see [Wyszecki 75] for a discussion of these terms) is a very important aspect in material discrimination. We regard metals as having a shiny appearance, whereas plastics, while they may have as smooth a finish, appear somewhat dull in comparison. Other surfaces may be altogether matte. These differences are caused by the presence, or absence, of local mirror–like or specular regions of reflected light, henceforth called highlights. If we can detect these highlights within an image, we can glean information that will help us to identify materials.

Some other consequences of finding highlights are that: (i) they would aid in constraining the size, color, and location of a light source; (ii) they would simplify

object recognition or matching by identifying the regions so that some of the effects of the illumination could be 'factored out'; and (iii) they also would enable constraints to be placed on object size and location [Thrift 82]. Perhaps a more basic or fundamental reason for wanting to locate highlights is that computer vision is concerned with modeling human vision, of which an inherent feature is the ability to locate highlights.

The thesis format is as follows: Chapter 2 contains a review of pertinent literature, Chapter 3 discusses the research equipment, Chapter 4 is comprised of the image analysis algorithms, Chapter 5 presents the results, Chapter 6 proposes some extensions and Chapter 7 is the conclusion.

Chapter 2

Review of Literature

For the purpose of discussion, this will be the illumination model referred to;

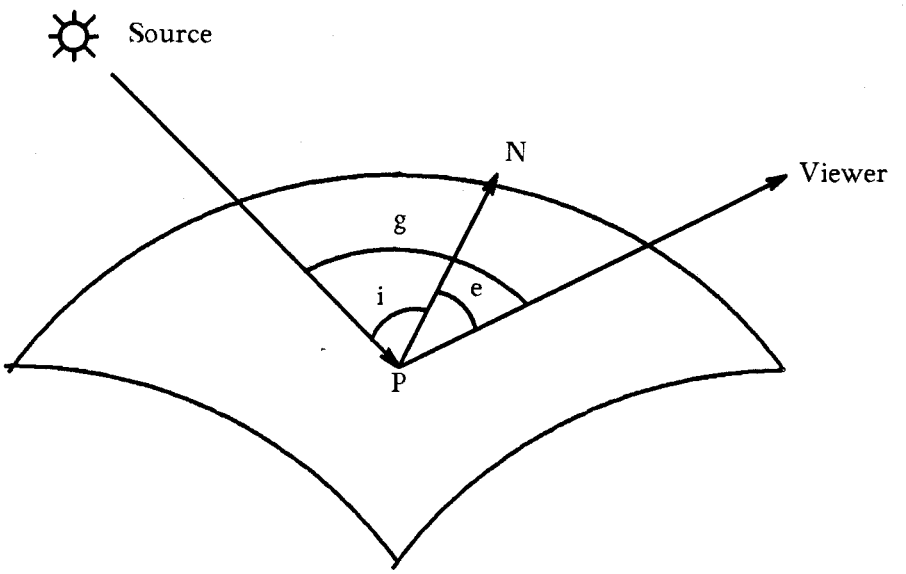


Figure 2-1: Illumination Model

i = angle between ray incident on the surface and the normal(N) to the surface at that point(P).

e = angle between reflected ray and the surface normal at point P.

g = angle between the incident and reflected ray, or the phase angle.

If we are talking about an ideal specular reflecting surface, then the angle of incidence equals the angle of reflectance $i = e$ and, by Fermat's principle of least

time [Sears 58], i,e, and N, all lie in the same plane. Therefore $i + e = g$. In reality most surfaces are not perfectly specular, and therefore they scatter light in a cone shaped region centered about the ideally reflected ray. It has been shown [Horn 75] that the cosine of the angle between the direction for perfect specular reflection and any other direction is $2cos(i)cos(e) - cos(g)$. In the ideal direction ($i = e$ *and* $i + e = g$) this equals one and falls to zero as the angle increases to a right angle.

A computer graphic simulation of the specular contribution of light scattered from a surface is accomplished by calculating $2cos(i)cos(e) - cos(g)^n$ for some positive integer n. The greater the value of n, the more compact the the highlight becomes. Phong [Phong 75] simplified Horn's term by using $cos^n(i - e)$, showing that the shape of the highlight curve is not critical to the perception of such phenomenon. The contribution of the highlight is only one component in the final calculation of the intensity value for a given computer generated surface. A complete system such as Hall and Greenberg's [Hall 83] reflectance model (see figure 2-2), using ray-tracing techniques, has produced very realistic images.

The important thing to note, from the simulation model, is that the components of the illumination are separable, and that the spectral component creates a very conspicuous curve that peaks with respect to the mirror direction. It is this 'peaked' region that constitutes a highlight.

Highlights of various sizes and shapes can be divided into two classes that are appropriate for analysis; those arising from curved surfaces, and those from planar surfaces.

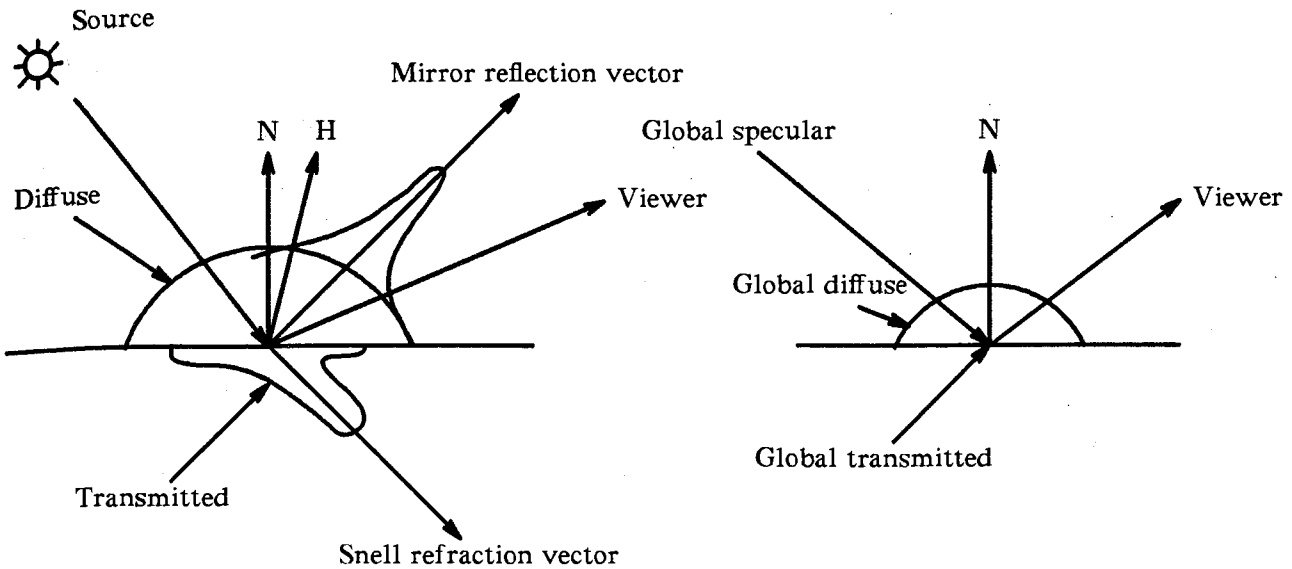


Figure 2-2: Hall illumination model

2.1. Highlights from curved surfaces

By assuming a point source and viewer located at infinity, Forbus [Forbus 77] generated a series of one dimensional profiles of intensities to determine the parameters relevant to the perception of highlights in achromatic images. Forbus found that the contrast between the specular and diffuse illumination components was not an important factor in perceiving highlights, but rather that the ratio of their respective widths is. Forbus also noted that the shape of the specular component curve $2cos(i)cos(e) - cos(g)$ did not matter, as long as it was not a step–change in intensity. He concluded that:

(i) If the diffuse component was from a curved matte surface, then as long as the specular component width was less than 75% of the total width of the reflecting surface, it appeared as a highlight (see figure 2–3). If the diffuse component's width is B and the specular component's width is A; $\frac{A}{A+B} < .75$.

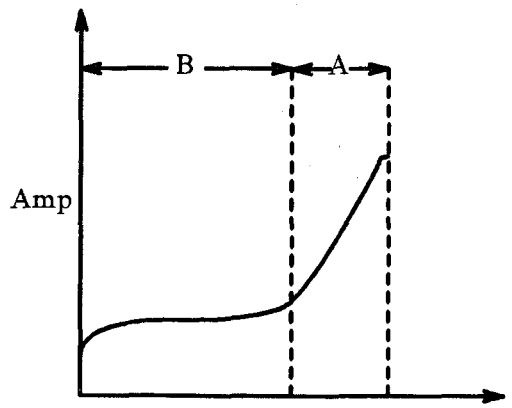


Figure 2-3: Highlight vs. Curved Matte Surround

(ii) If the diffuse component was constant, then as long as the specular component width was less than 20% of that component's width, it appeared as a highlight (see figure 2–4). If the diffuse component's width is B and the specular component's width is A; $\frac{A}{B} < .2$.

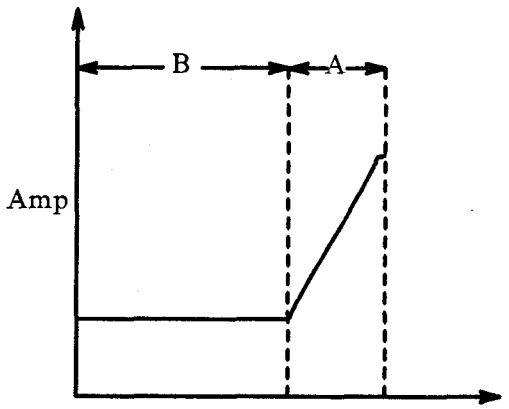


Figure 2-4: Highlight vs. Constant Surround

Forbus' conclusions are extremely useful for computer generation of scenes that contain highlights, but they do not give any insight into how highlights may be detected in a natural image. If an image could be segmented into objects the

analysis to determine Forbus' parameters would require the parsing of the object into highlight and surround regions. Forbus does say that part of the requirement for this subsequent parsing is that the local context must be right. This merely means that the second derivative of intensity must exist and be non zero.

2.2. Highlights from planar surfaces

If surfaces are perfect mirrors, then we can interpret the search for the highlights they cause as equivalent to a search for light sources within the image. These virtual sources will have all the properties of real sources and can be located by looking at two adjacent areas in the image and computing both their intensity and gradient ratios assuming they have the same orientation and illumination (see Appendix A). If the ratios are not equal, then one of the areas is perceived as a light source [Ullman 75].

The problem, as Ullman points out, is that real images tend to be noisy and the boundaries of areas tends to be quite indistinct (see figure 2–5).

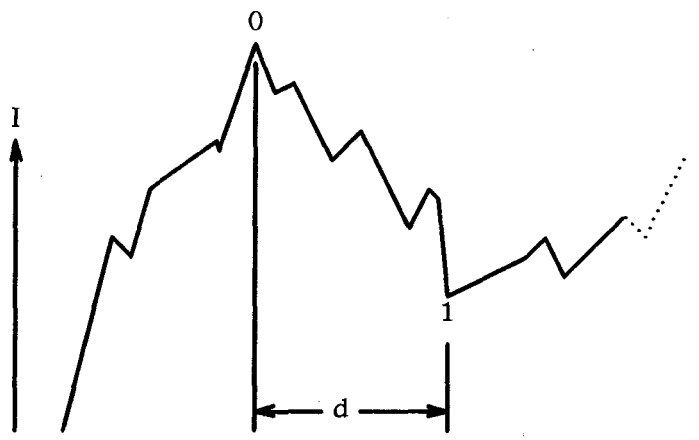


Figure 2-5: Realistic image intensity data

To alleviate some of the problems this causes, he derived a S–operator which finds the light source intensity. The S–operator (S) is defined as follows:

$(S)\ L = Amp_0 - Amp_1 * (S_0/S_1) + S_0 d$

L = source intensity(assumed to be in area 0)

d = distance from area 0 to area 1

S = gradient in subscripted area

Amp = intensity reaching the eye (image intensity) at subscripted area boundary

The use of the S–operator includes setting a threshold above which an area is labeled as a source. The S–operator is not without its faults, such as:

(i) insufficient image resolution which creates problems when trying to determine region boundaries;

(ii) singularities in image intensities that cause spurious results; and

(iii) light sources detectable at low background intensities become undetectable at higher ones.

Forbus [Forbus 77] extended Ullman's S–operator by relaxing the restriction that the orientation of the areas be the same. The problems Forbus encountered were:

(i) the variation in the intensity gradient due to the beam spread of the source, which alters the angle of incidence to the surface; and

(ii) the effects caused by the lack of spatial uniformity in actual light sources and the effects of mutual illumination.

Forbus' S–operator gives a better indication of how to locate highlights from achromatic, planar surfaces, but there still exists the problem of subdividing the image into adjacent areas for analysis. Even if the areas could be found, Forbus' S–operator requires knowing the exact orientation of at least one of the areas so that the relative orientation between areas may be calculated.

2.3. Highlights and Color

Ullman's S–operator, later modified by Forbus, was used to search for highlights in an achromatic image. This was deemed important by Forbus since he observed that people can recognize highlights in black and white photographs. The use of color as a tool for highlight analysis resulted from Shafer's [Shafer 84] quest to separate the effects of shading from those of highlights.

Shafer separated the reflection components of materials into two parameters — interface and body reflection. These parameters are basically equivalent to the specular and diffuse reflection terms that are more generally used. Interface reflection occurs at the surface of the object, while body reflection occurs at the internal pigment level. Therefore interface reflection will have the color of the illuminant and body reflection will have a color caused by the interaction of the light with the subsurface pigment layer. Materials that have both types of reflection are called inhomogeneous.[1] Shafer's work is only relevant for inhomogeneous materials.

---

[1] examples include plastics, most paints, ceramics, varnishes, etc.

If the assumption is made that the objects are uniformly colored then a plane may be fitted through the object's pixel values in color space. Shafer then bounded this plane by a parallelogram whose sides are the values of the interface and body reflection colors. This means that you can then express any pixel color value for the object as a linear combination of these two factors.

To separate the effects of highlights from that of shading, or interface from body reflection, the parallelogram could be parsed into two regions across which a recognizable color shift occurs. Difficulties in implementing Shafer's method would arise in defining the boundaries of the objects in the scene and in subdividing the resulting parallelograms. The latter of these two problems would cause the method to fail as a highlight detection scheme for inhomogeneous materials because the color shift apparent in real images could be extremely gradual, which means that the parallelogram would have contiguous color values from one side to the other. It would only be possible to determine where the color shift actually occurred with probabilistic analysis.

The previous discussions focussed on research that concerned the detection of highlights within an image. The following section will pertain to the uses of identified highlights.

2.4. Applications of Highlights

2.4.1. Object Shape, Size and Orientation

One use for the identified highlight is the determination of object properties such as shape, size and orientation. By looking at figure 1–1 we can easily see that this information is apparent in this scene. However, the extraction of such information from the scene depends on having the highlights predefined.

Thrift and Lee [Thrift 82] use highlights to constrain object size and location for cylinders and spheres assuming known highlight and point source location. They use the 'highlight point', that is the point where $i = e$ and $i + e = g$, to derive their constraints. The simplest case to imagine would be a specularly reflecting sphere with a single highlight pixel on it that corresponds to the highlight point. Knowing the source direction you can deduce the normal to the surface at that highlight point. Using the equations that define the distance to the highlight point you can derive constraints for the sphere's radius and center location.

Babu, Lee and Rosenfeld [Babu 85] studied methods of determining the orientation of specularly reflecting planar surfaces. By assuming all the properties of the light source are known, they used the contours of equal brightness to find the surface normal. Along these contours the normal and source vectors are constant, as are the parameters describing the surface reflection properties. This means that by using a reflectance model such as Horn's, it is possible to derive the unknown normal vector. While this is true for planar surfaces only, a highlight detection algorithm is of great use in defining the highlight region so that it can be analyzed.

2.4.2. Object Material Classification

As stated previously, one use of highlights is to constrain object size and location. An additional use for highlights is outlined by the discovery of Cook and Torrance [Cook 82], of the visual difference between metal and plastic.

> A typical plastic has a substrate that is transparent or white, with embedded pigment particles. Thus the light reflected directly from the surface is only slightly altered from the light source. Any color alterations are a result of the reflectance of the surface material. Light that penetrates into the material interacts with the pigments. Internal reflection thus gives rise to a colored, uniformly distributed diffuse reflection.
>
> ...reflection from a metal occurs essentially at the surface. Thus internal reflections are not present to contribute to a diffuse component, which can be important for a nonmetal. ...The specular reflectance component [of copper] has a copper color...showing that a correct treatment of the color of the specular component is needed to obtain a nonplastic appearance. [Cook 82]

Bearing this in mind, we could use this information to differentiate between objects made of plastic, and those made of metal. This analysis requires the use of color images rather than black and white.

Chapter 3

Research Equipment

An *International Imaging System* (IIS) Model 70F interactive digital processing system with a Vax 750 host computer was used for the displaying and processing of previously digitized color images. The IIS provides a mode whereby 48 different arithmetic and logic functions can be used to process 512*512*8 bit images in 33 milliseconds or one frame time. The ability to do these computationally intense operations very quickly is particularly important since the algorithms in this thesis require a series of many such operations.

The IIS provides a *Refresh Memory* (RM) in which images resident on the host computer's disk are stored for either processing or displaying or a combination of both. Preliminary image processing may occur in the *Input Function Memory* (IFM) as the data travels from the disk to the RM. This preliminary processing helps to increase the overall speed of the imaging system since some of the operations performed by the IFM would otherwise have to be done by the much slower host computer.

The IFM is a look–up table that is applied to the data on the way to the RM. It may be used or bypassed. Since the RM can store only 8 bits of information, the IFM may do things such as scale, take logarithms of, add constants to, etc. any data from the disk of up to 13 bits.

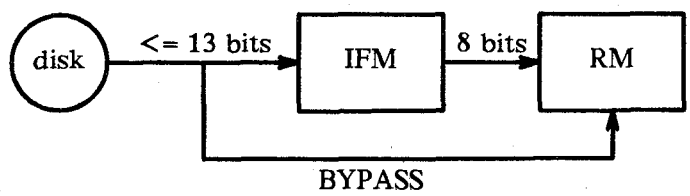


Figure 3-1: Disk data path

Whether the data from the disk passes through or bypasses the IFM it arrives at the RM to be stored. The current configuration has four RM boards, or channels, which are labeled as CHANNL1, CHANNL2, CHANNL3 and CHANNL4. The data may be stored in any or all of the channels. Once the data is stored in the channel(s) it can be displayed on the color monitor.

3.1. The IIS image display path

The IIS has a red, green and blue (RGB) output which feeds the RGB monitor. Thus the image that is located in the RM can be displayed in any one of these primary colors on the screen and these color outputs are called the *red, green or blue pipe.* A complete color image requires each tricolor–band component of the image to be sent to the screen through its respective pipe. What this means is that, to get a color picture displayed on the screen, the IIS requires three different disk images of the same scene, one each for the red, green and blue intensity components that make the RGB display image. Post RM image intensity pipeline data can be altered to change the colors that are actually displayed on the screen in the *pipeline color processor.* There is one pipeline color processor for each color pipe.

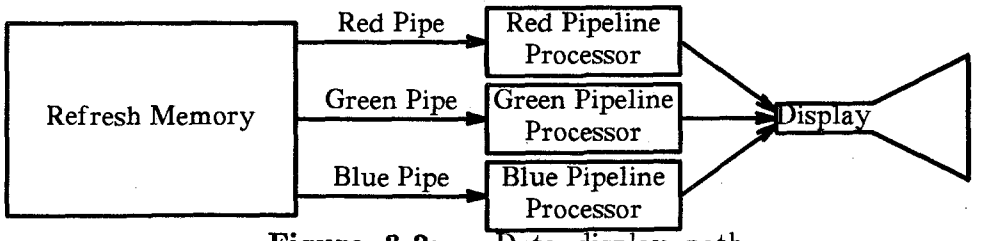


Figure 3-2: Data display path

A pipeline color processor has features such as image scrolling, hardware zoom, an adder array, a *Look-Up Table* (LUT) for each RM channel and a *Output Function Memory* (OFM). The 8 bit image intensity data stream from the RM channel is modified by using each intensity value as an index into the LUT array and replacing it with the value stored there. For example if there is a negative LUT for one channel (each image intensity value is replaced with its negative counterpart), and a positive LUT for a different channel and the data stream is sent through the same pipeline color processor then the adder array will do the two's complement sum of the LUT outputs. The result will be the difference of the two images. This enables arithmetic operations such as add, subtract, multiply and divide to be accomplished in one cycle time.

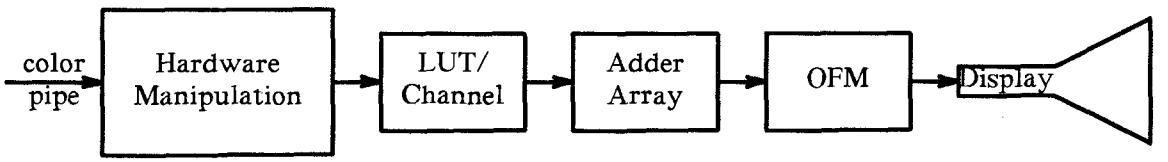


Figure 3-3: Pipeline color processor

Before the results of any such operations on the data can be displayed it may be necessary to scale the results. The IIS provides the OFM to perform this task.

The OFM is just a LUT whose output generates the final RGB values, 0 to 255, to be displayed. It should be noted that the original data in the RM channel is unmodified no matter what operations are performed to get the display image. To keep the display image requires storing it in a RM channel.

3.2. Image processing

Image processing often requires more modifications to the image intensity data stream than those that can be performed along the image display path. The data may need to be stored, remodified, compared, displayed, or stored, in a cycle that may have many iterations. To increase the speed of these more complex image processing operations the IIS provides a loop from output to input that omits the intermediate display and store steps. This is called the *feedback loop.* The feedback loop has an *arithmetic and logic unit* (ALU) that can perform 48 different operations on the data from the OFM. The ALU requires two inputs for the operations, one being the output from the OFM and the other being the 512*512*n bit (n = 8 or 16) data in the *accumulator.* The precision of the accumulator can be 8 or 16 bits depending on whether channel 1 or the channel 1 and channel 2 pair (channel 1 for low order bits and channel 2 for high order bits) are used respectively.

The application of image analysis procedures such as smoothing, differentials and edge enhancement are accomplished by convolving the original image by an odd sized mask that represents the task. A convolution is the sum of all the pixels under the mask, with each pixel being weighted in the sum by its overlying mask value. Each pixel in the original image is rewritten as the resultant of this

weighted sum when the image pixel is under the mask center. If we have a n–by–n image and a m–by–m mask, this means that

$$Pixel_{row,col} = \sum_{i=\lfloor -m/2 \rfloor}^{\lfloor m/2 \rfloor} \sum_{j=\lfloor -m/2 \rfloor}^{\lfloor m/2 \rfloor} weight_{i,j} \times pixel_{row-i,col-j}$$

The importance of using an image processing system like the IIS is realized upon the comparison with the host computer's capabilities. To perform a convolution on an entire image, with a 3*3 mask, takes 11 feedback cycles or 363 milliseconds, whereas with "a conventional computer, it would be necessary to perform 2,359,296 floating point multiplications and 262,144 floating point divisions." [Bryant 83]

Chapter 4

Image Analysis

4.1. Taking Pictures

The photographic process involves finding the subject matter, taking a light meter reading to determine the correct exposure and f–stop setting, and then exposing the film. It is irrelevant if some areas are recorded incorrectly on the film, as long as the picture looks correct. This merely means that we can process a picture distorted by filters, incorrect exposure, etc. and still describe what the picture contains ie. we can recognize a mountain and a tree no matter what color they happen to be.

The problem with taking pictures for a computer to interpret is that the computer does not know what 'looking right' means for a scene. An example could be a picture taken of a car's bumper on a sunny day. We know that a car's bumper should appear quite shiny and therefore those bright spots in the picture are specular reflections. The computer has no conception of what a car's bumper is and must formulate its conclusions solely on the intensity values present within the picture. If those bright spots in the picture were actually overexposed, their digitization would not produce curves that the computer could interpret as highlights, but more like step changes in intensity or edges.

Adjusting the picture–taking process so that the highlights are not overexposed may cause the rest of the bumper, or the car to be underexposed. The realization that highlights can be many times brighter than 'white' under the same lighting conditions gives some idea of the difficulties when taking pictures of scenes that contain highlights. This is particularly important since the number of different intensity values that the IIS can quantize is 256.

The IIS has a video digitizer whose input comes from a black and white surveillance camera. To get a RGB image of a scene with the video digitizer and a black and white camera requires each color component of the scene to be digitized separately through a color filter. The filters used are standard tri–color separation filters. There is a problem with using the surveillance camera with color filters as frame input since the transmittance of each of the filters is different and the camera tube records intensity values into the near infrared. Non–equivalent transmittances can be corrected by adjusting the f–stop on the lens, but the infrared problem proved to be unmanageable.

The contribution of the infrared radiation of objects to the digitized scene effectively adds a constant to each one of the color bands, which is like whitewashing the displayed RGB color image. Colors are pale and indistinct. An infrared absorbing filter can be used in conjunction with the tri–color separation filters to alleviate this problem, but the added absorption of the infrared filter to that of the darker blue and green filters causes the scene to be so dark that f–stop adjustment to equalize transmittances becomes ineffectual. If the scene is digitized in its dark state the colors are all distorted since the camera's gain and

pedestal hardware tries to correct for this situation. To try and bypass these problems color slide film was used in preference to the surveillance camera.

Color slide film was exposed at various f–stop settings with a constant shutter speed so that the best possible picture of the scene could be recorded. The best possible picture was so judged based on the minimization of the problems associated with highlight scenes. This picture was then digitized into its three color bands with a resolution of 25 microns using an Optronics Digitizer with its built–in RGB filters. An additional advantage to using color slide film is that the scene needs to be recorded only once; whereas, with black and white film or a black and white camera, the same scene must be immobilized for three pictures to prevent the aliasing of pixels.

4.2. Differential Operator

Luminance edge detection has become one of the primary building blocks used for image analysis. The luminance edge is caused by local discontinuities in image intensity values, and often demarcates the boundaries between objects. The analysis of 'edge' images is less complicated than analyzing the objects themselves because of the amount of information that must be processed. The principal methodology for extracting edge information has been through the use of differential operators.

First order spatial derivatives are defined as;

$$d_x = \partial I(x,y)/\partial x$$
$$d_y = \partial I(x.y)/\partial y$$

which gives rise to the following scenario (see figure 4–1) for an illumination edge.

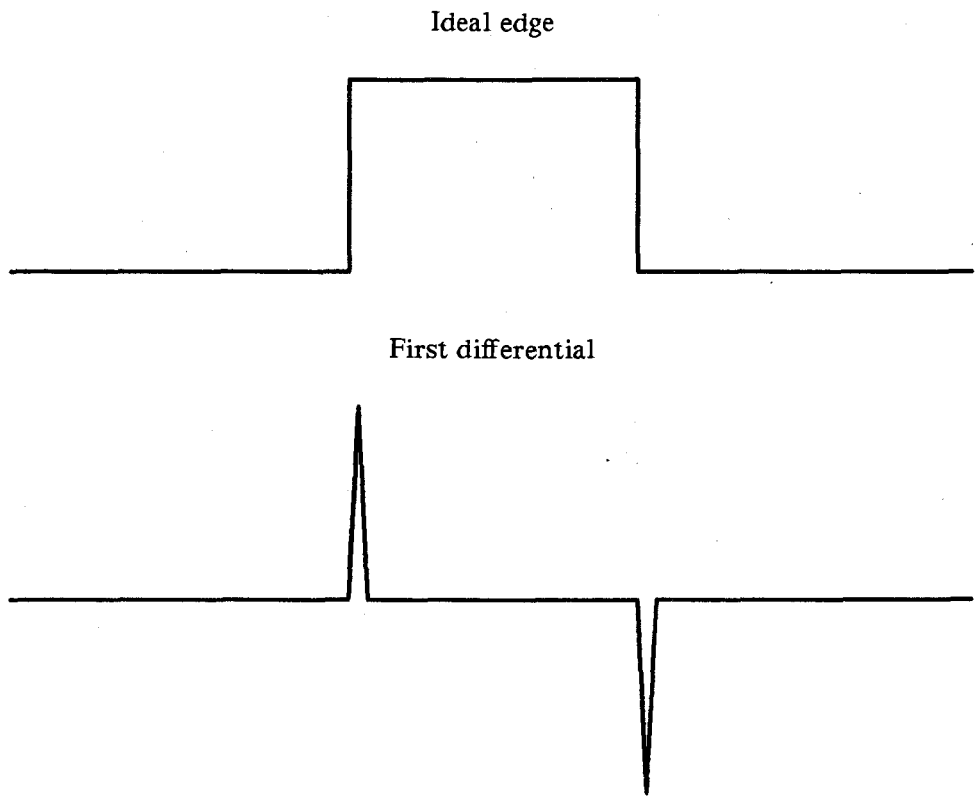


Figure 4-1: One dimensional directional derivatives

The problem with using first order differentials is that a threshold must be set, above which a pixel is considered an edge element. The setting of the threshold is not an elementary chore since it depends upon the noise and range of intensity values present within the image. If the threshold is set too low, discontinuities caused by noise could be interpreted as edges, whereas if the threshold is set too high, smaller valued edges could be eliminated as noise. Marr and Hildreth [Marr 75] rectified the problem of setting an arbitrary threshold by looking at the second order differential rather than the first order differential.

The second order differential has a zero–crossing at the point of the luminance

discontinuity. A zero–crossing is a term describing the positive to negative or negative to positive transition of neighboring pixels within the second differential image. The application of the second differential is direction dependent and thus must be applied enough times to coincide with all the possible orientations for edges. However, there is an operator, called the Laplacian, which is formed from the second order spatial derivatives

$$d_{xx} = \partial^2 I(x,y)/\partial x^2$$
$$d_{yy} = \partial^2 I(x,y)/\partial y^2$$

that is non–directional [Pratt 78], ie. it is a scalar. One important point about the solution to Laplace's equation, $d_{xx} + d_{yy} = 0$, is that the intensity cannot have a relative maximum or minimum inside the region defined by the solution unless the intensity is constant [Powers 79]. This proves to be an important factor in Marr and Hildreth's edge detection algorithm.

The formation of highlights takes on a different interpretation than edges. For example we have seen that highlights are of a roughly triangular shape (see figure 4–2), so rather than having a zero–crossing in the second derivative, used by Marr and Hildreth to identify edges, we want to locate zero–crossings in the first derivative.These points correspond to $\partial I/\partial x=0$ and $\partial I/\partial y=0$. Since these points can be either a maximum or a minimum, care needs to be taken that a peak is truly found and not a trough as might be caused by a shadow edge. Therefore a second operator must be used in conjunction with the previous operator to eliminate this possibility. The following sections describe these two operators.

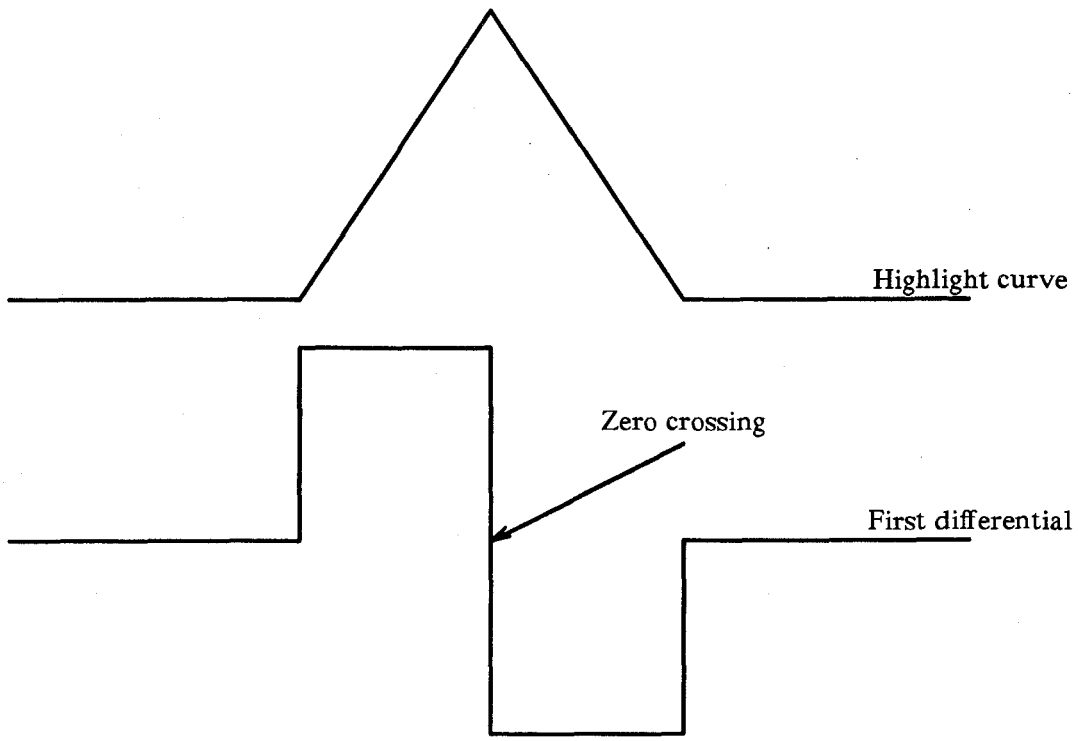


Figure 4-2: Highlight curve and its derivative

4.3. First Differential Operator

The existence of noise and other types of intensity singularities have proved to be a source of frustration when trying to analyze images. The use of a smoothing operator has been employed to try and eliminate these image irregularities. As Marr points out

> The reason why one chooses the Gaussian for this purpose, rather than blurring with a cylindrical pill–box function (for instance), is that the Gaussian distribution has the desirable characteristic of being smooth and localized in both the spatial and frequency domains and, in a strict sense, being the unique distribution that is simultaneously optimally localized in both domains. And the reason, in turn, why this should be a desirable property of our blurring function is that if the blurring is as smooth as possible, both spatially and in the frequency domain, it is least likely to

introduce any changes that were not present in the original image. [Marr 82]

The idea, therefore, is to use a Gaussian operator on the image intensity values to eliminate said singularities.

The form of the smoothing operator is a two dimensional mask of values, calculated to simulate a Gaussian, which can then be convolved with the image. The spatial constant σ, which denotes the shape of the Gaussian, determines the size of the image irregularities that will be removed. Since σ defines the shape of the Gaussian, it is important that the size of the Gaussian mask be variable with σ because they are dependent terms (see Appendix B).

Once the image is smoothed, the procedure is then to differentiate the image intensity values by convolving it with another mask that represents the first differential. The simplest mask for doing this is based upon the definition of the derivative.

$$d_x = \frac{I(x+h)-I(x)}{h} \quad d_y = \frac{I(y+h)-I(y)}{h}$$

Considering discrete valued image coordinates, the minimal value for h is 1 and the resultant x and y directional derivatives are:

$$d_x = I(x+1)-I(x) \quad d_y = I(y+1)-I(y)$$

By looking at the corresponding convolution masks in figure 4-3 it is obvious that they are not symmetric and the range of pixels that contribute to the calculated differential is quite small. Since the convolution will write the result of the application of the mask to the pixel (x,y) the differential will be biased in the

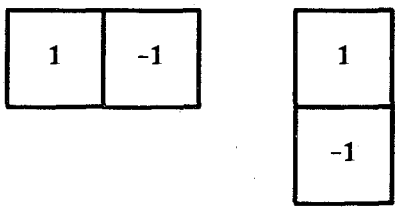


Figure 4-3: x & y directional derivative masks

direction of (x+1,y) and (x,y+1) for the x and y directional derivatives respectively. That only two pixels contribute to the convolution also causes a problem since the error in the average over those two pixels can be quite high. The two problems are linked because an increase in the number of pixels contributing to the convolution decreases both the error in the average over those pixels and the non-symmetrical nature of odd-ordered derivatives (odd-ordered derivatives have an even number of terms while even-ordered derivatives have an odd number of terms). These difficulties can be corrected by a mathematical identity. That is, the derivative of the image that results from convolving the original image with the Gaussian is equivalent to convolving the derivative of the Gaussian with the original image.

If $G(x,y)$ *is the Gaussian*
and $\otimes$ *the convolution operator*

$$d_x(G(x,y) \otimes I(x,y)) = d_x G(x,y) \otimes I(x,y)$$

and likewise for the y direction.

This enables flexibility in the range over which the first differential applies because the range now becomes a function of σ. We can make the mask as large as we like by choosing an appropriate σ which negates the effect of the unbalanced symmetry.

While we may choose to apply our differential operator using a single value for σ, that would be as difficult as setting a threshold for the determination of edges. Highlights with a wide range of sizes can only be located by using multiple values for σ and then ORing all the results together. To ensure that we find zero–crossings not caused by singularities due to the choice of σ, the algorithm incorporates a phase which finds the zero crossings for two values of σ and then ANDs those images together. Taking two values of σ relatively close together ensures that when we AND the results we do not eliminate 'true' highlights due to the scale of our operator. We can then OR a few such σ pairings to cover the gamut of highlight sizes.

However, we must remember that the differential masks are directional (non–isotropic) and by using a mask larger than 2–by–1, this means other directions, besides just the x and y axes, become relevant. The actual form of the n^{th} order generalized derivative is given by:

$$f^n(x,y) = \sum_{k=0}^{n} \binom{n}{k} \frac{\partial^n I(x,y)}{\partial x^{n-k} \partial y^k} cos^{n-k}\theta \, sin^k\theta \quad [Prewitt70]$$

and in particular the first order directional differential of the Gaussian, G(x,y), will be:

$$\frac{\partial G(x,y)}{\partial x} cos\theta + \frac{\partial G(x,y)}{\partial y} sin\theta$$

The consequence of having a non–isotropic differential operator is that it must be applied to the image at various angles of θ. If, for example, a highlight is vertically oriented in the image we would expect a vertical line of zero–crossings to result from the application of our operator. If the differential operator is

computed with θ equaling 0 degrees (horizontally sensitive) the outcome of its convolution with the image would be zero values rather than zero crossings. Therefore the first differential Gaussian operator was convolved with the image for various angles of θ.

By looking at the possible orientations of highlights in an image we can determine which angles of θ should be used. Of course we could use the angles that correspond to the N, NE, E, ... NW compass point neighbors of a pixel, but to reduce the number of computations necessary to find the zero–crossings we would like to minimize the number of angles. As mentioned previously, highlights oriented perpendicular to that of the differential operator do not give a zero–crossing. However, as long as the differential operator's angular orientation is within forty–five degrees of the highlight's orientation, zero–crossings will result along the line of the highlight (of course this is assuming that the highlight is of sufficient size to be recognized as such). But if the differential operator is applied at every forty–five degrees, we are back to looking at each pixel's neighbors for all the points of the compass.

The solution to this dilemma comes from the realization that if the orientations of two differential operators are 180° apart, then the only difference between their resultant zero–crossings is whether the zero–crossing transition goes from positive to negative or negative to positive. The key factor is that the zero–crossings are identical. Therefore we need only apply the differential operator four times with θ being 0°, 45°, 90° and 135° (see figure 4–4 for zero–crossings vs. θ).Since it was determined that an angle of forty–five degrees between the differential operator's

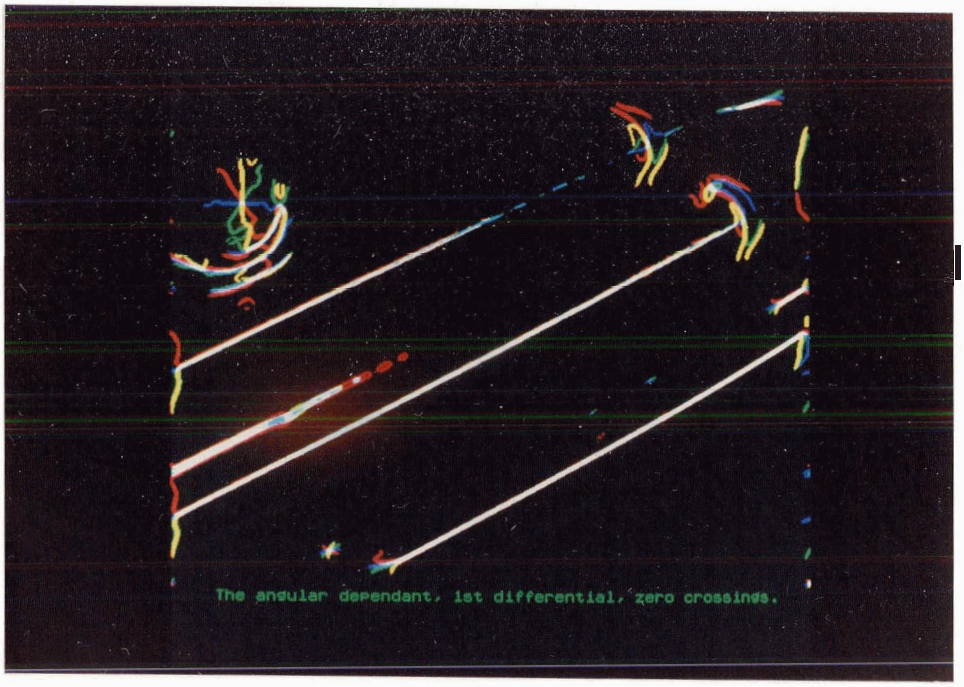


Figure 4-4: Zero–crossings vs. angle

and highlight's orientation was the upper limit for zero–crossing detection, it might be assumed that differential masks should only be convolved with the image at two angles, say 0° and 90°.

The differential mask with 90° orientation would find all the zero–crossings for highlights between 45°–135° and 255°–315°, similarly, the other mask would find all the remaining zero–crossings. While this works for most object's highlights, the exception is spherical or planer objects that create circularly symmetric highlights when the source is also circularly symmetric. We would expect that the result of the convolutions with the image to give a symmetric blob located at the center of the highlight. Instead, due to the symmetrical shape of the highlight, a differential mask oriented at any angle θ, will find a line of zero–crossings across the highlight oriented at that angle θ. This is because a cross–section of a circularly

symmetric highlight in a direction perpendicular to that of the differential mask's looks like a one–dimensional highlight curve. Therefore that pixel is interpreted as a zero–crossing element. To alleviate the confusion this might cause when trying to interpret the resulting zero–crossing diagram, it is necessary to regress to using four angles of θ. The convolution of the differential mask with the image need only be done twice however (for two orthogonal angles) since the other angular orientations can de derived from those results. Once we have the zero–crossings for the four angles of θ the algorithm must then combine the results in some manner.

The solution is to binarize (set zero–crossings to one and everything else to zero) the four zero–crossing images and then logically AND all the six possible image pairs together. The six images resulting from the previous ANDing operations are then logically ORed together to form the final zero–crossing diagram for the original image (see figure 4–5). A precursor to any of the aforementioned logical operations is the necessity to blur or widen the zero–crossing chains present in the zero–crossing image. This is because the ANDing of two images in which the zero–crossing chains are one pixel wide could mistakenly give a zero where there should be a one. This occurs because pixels may be diagonally connected in two directions and not overlap, even though they represent the same highlight (see figure 4–6). The blurring operation is accomplished by convolving the zero–crossing image with a 3–by–3 mask so that the zero–crossing chains become three pixels wide, thus leaving no holes between diagonally connected pixels.

Now that we have the final, fattened, zero–crossing image, we must parse it into

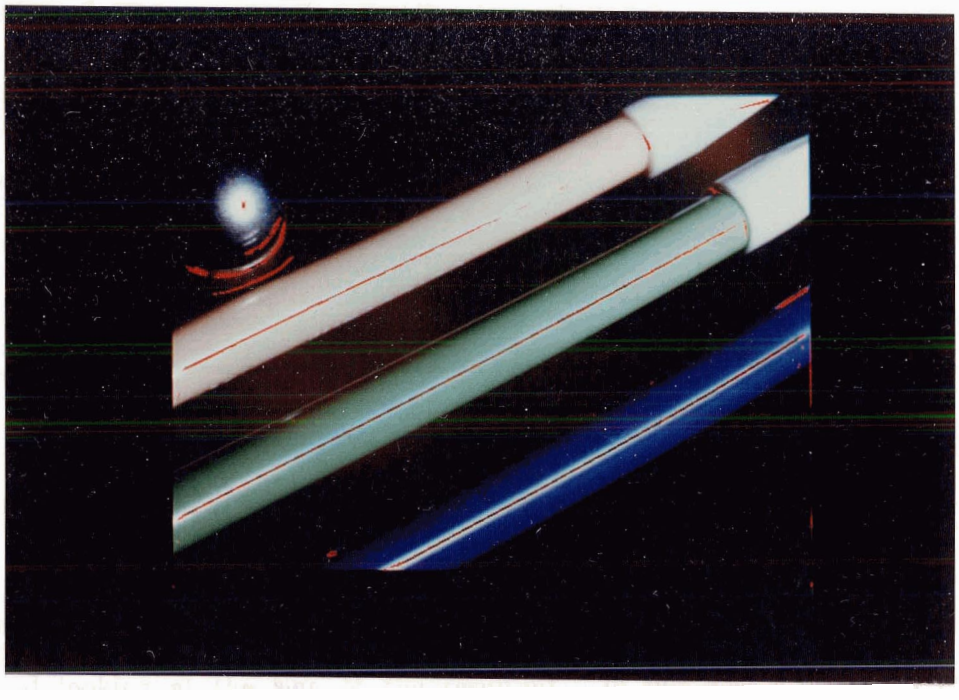


Figure 4-5: Final zero–crossing image

*				#
	*		#	
		*	#	
		#	*	
	#			*

Figure 4-6: Non–overlapping highlight chains

zero–crossings caused by peaks and troughs in intensity. Upon completion of this task it will be necessary to undo the effects of blurring the zero–crossing chains since we only want a single pixel wide chain representing the center of the located highlight. The next section describes the algorithm for determining concavity.

4.4. Concavity Operator

The zero–crossing image we have thus far contains more information than we can make use of. First of all the blurred zero–crossings are too wide and secondly the unwanted zero–crossings caused by regions of darkness in the original image are still incorporated in the zero–crossing image. The undesirable zero–crossings can be removed by looking at the concavity of the region in which the zero–crossing pixels lie. If it is concave down then the zero–crossing is caused by a highlight; otherwise, it is assumed to be caused by a valley.

Concavity is determined by taking the second derivative with respect to the curve and looking at the sign of the resultant. In our case this means convolving the original image with a second differential mask. The mask could be formed in the same manner as the first differential directional mask, that is the differentiation is with respect to the Gaussian, but with the exception of using the Laplacian rather than the directional derivative so that the convolution need only be done once. The result of the convolution is an image composed of positive and negative regions demarcating concave up and concave down areas respectively. The unfortunate part about this scheme is the need to be accurate in the determination of these regions. The integer arithmetic performed by the IIS array processor during a convolution causes inaccuracies in the resultant concavity image that cannot be overcome. This is illustrated in Appendix C.

To alleviate the inaccuracies of the IIS, an operator was designed to do the work of the second differential, ie. find the concavity in the neighborhood of a pixel. This operator is more effective than the Laplacian of the Gaussian since it

uses larger mask values overall, and does so in a manner that facilitates a better understanding of the resultant image.

What we are really doing when we convolve the second differential mask with an image is determining whether or not a particular pixel has a greater value than all of its neighbors. If it does then it must be locally concave down at that point. Due to inherent noise in the images, rather than choosing a single central pixel to compare to its neighbors, it is more desirable to take an average of a bounding area around that center pixel. For the same reason an average should be taken over the surrounding neighborhood to be used for comparison to the center area. The mask is named neighborhood–average for obvious reasons.

The neighborhood–average mask is grown by a square tessellation out from the mask center, which is zero–valued. If the center of the mask is (0,0) then the square perimeter at a distance i from (0,0) will have the number of pixels $= 2^3 \times i$. So if the central portion of the mask has width *centersize*, the total number of pixels in the center area will be $\Sigma_{i=1}^{\lfloor centersize/2 \rfloor} 2^3 \times i$. Likewise, the surround will have $\Sigma_{j=1}^{wingsize} 2^3 \times [\lfloor centersize/2 \rfloor + spacesize + j]$, where *wingsize* is the width of the square perimeter surround and *spacesize* is the width of space between the end of the central area and the start of the surround (see figure 4–7).

The concavity is determined by the sign of the value in the image produced by the convolution of the neighborhood–average mask with the original image. Since the surround area comprises more pixels than the center and the values for the surround mask are calculated such that their sum with that of the center is zero,

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1								-1
-1								-1
-1			4	4	4			-1
-1			4		4			-1
-1			4	4	4			-1
-1								-1
-1								-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Figure 4-7: neighborhood-average mask.
centersize=3,wingsize=1,spacesize=2

it was desirable to maximize the values present in the center area. The effect is to offset the error caused by small mask values as discussed in Appendix C.

As was stated previously, we can determine the concavity by the sign of the resultant image value. However, there is more information in this convolution image than just the concavity. Underlying the sign of the concavity is the gradient of the highlight in the concave down regions. If we were to make assumptions about consistency in source lighting and object reflectivity, we could use this information to give a measure of object orientation. This is because the maximum value of the gradient will occur when the view and source vectors are 45° from the surface normal and lie in the same plane as the surface normal (see Chapter 2). However, no such assumptions were made for this thesis, and so we cannot use the gradient information in such a manner. What we can do is choose a number for the gradient that will enable the calculation of what values are expected from the convolution and then exclude all resultant pixels from the convolution image that have smaller values. This assumes consistency in image

surfaces over the range of the convolution mask and small image gradients due to source location and size.

If the neighborhood–average mask center (0,0) overlays a highlight peak pixel with intensity VAL and we call the gradient SLOPE (assumed constant for the calculation), the expression for the resultant image value IV after the convolution is given by:

$$spacesize = s$$
$$wingsize = w$$
$$centersize = c$$
$$masksize = centersize + 2\times(spacesize + wingize) = m$$

$$IV = 2^3 \sum_{i=1}^{\lfloor c/2 \rfloor} i(VAL - SLOPEi) - \frac{c^2-1}{4w(m-w)} 2^3 \sum_{j=1}^{w} [\lfloor c/2 \rfloor + s + j](VAL - SLOPE[\,])$$

therefore

$$IV = 2^3 SLOPE[\frac{c^2-1}{4(m-w)} \sum_{j=1}^{w} [\lfloor c/2 \rfloor + s + j]^2 - \sum_{i=1}^{\lfloor c/2 \rfloor} i^2]$$

The interesting part about this equation is that it expresses the relative difference between the center and surround based on the neighborhood–average mask parameters and the image gradient and does not depend on the absolute magnitude of the intensities in the image. This equation is particularly useful since we can ignore small gradients that could not possibly be interesting as concave down sections by setting up a LUT that sets pixels whose value is less than IV to zero. Obviously the gradient has to be one or more over the area considered, and the value chosen for the gradient was two.

The choice of two was due to the consideration of the IIS hardware as well as

the desire to minimize the number of excluded pixels. Maximum OFM values are generated when the neighborhood–mask values are scaled upwards by a factor of 4, which also helps reduce the error caused by small mask values. The neighborhood–average mask used had centersize = 3, spacesize and wingsize = 2. Thus the convolution of the neighborhood–average mask with the image will give values between −14 and 1020 ($\frac{-4}{72} \times 255$ *and* 255×4). This means we need two channels for the accumulator (channel 1 and 2) for fifteen bits of integer precision and a sign bit. Using two channels requires checking the sign bit in channel 2 and then picking up the relevant data pixels from both channels. To save a lot of work it is desirable to circumvent the necessity of examining two channels. This was done by only looking at the positive values present in channel 2. Since the most significant bit of channel 2's eight bits is the sign bit, this corresponds to values between 1 and 127. But the values in channel 2 actually correspond to numbers between 2^8 *and* 2^{15}. Therefore a pixel value of 1 in channel 2 actually means 256. The slope multiplicity factor from the previously derived IV equation is 142 for the neighborhood–average mask being used and therefore the SLOPE value is equal to $\lfloor 256/142 \rfloor$ or two.

Like the first directional differential mask, the choice of an operator with constant size and shape will not be optimal for all of the scales inherent in the image. The sphere in figure 4–8 does not have the center portion colored as being concave down because of the large size of the highlight as compared to the neighborhood–average mask. We would like to find the correct concavity for highlights that have a smaller slope than those previously determined while still having a value of 1 in channel 2 as being our delimiter. The way to do this is called hierarchical discrete correlation or HDC [Burt 83a].

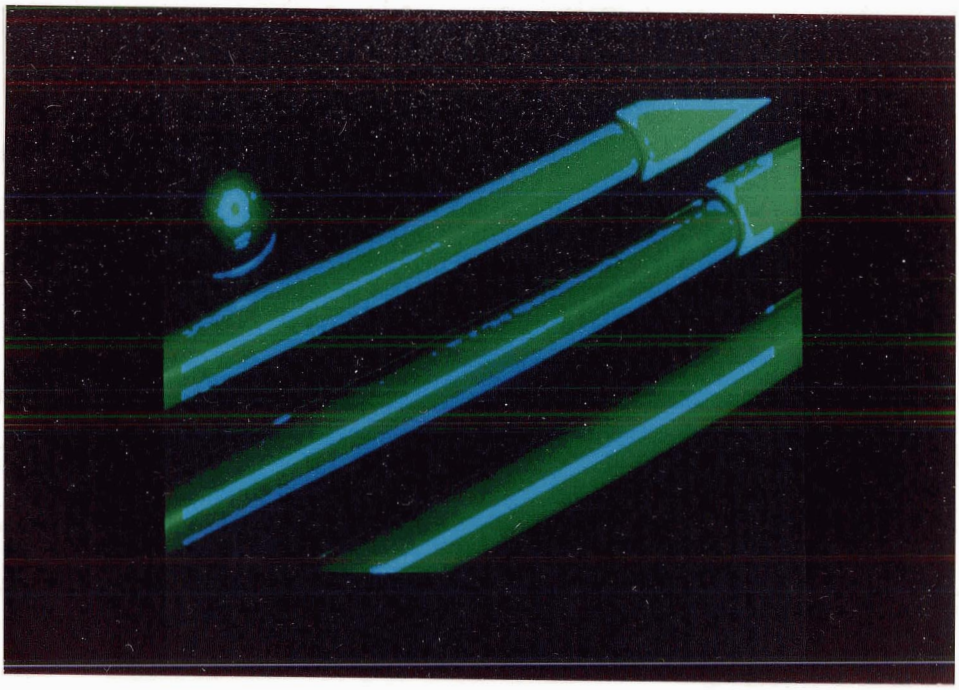


Figure 4-8: Concave down sections after one application of the neighborhood–average mask

The basic idea is that the samples that contribute to the computation are not contiguous image pixels. "Rather, they are separated by a distance d_1, and this distance is doubled with each iteration. As a consequence the HDC process generates a sequence of low–pass filtered images in which the bandlimit of each image is one octave lower than that of its predecessor." [Burt 83a] HDC uses the fact that contiguous encoded information in the image is redundant and any operators being used on subsequent images need not examine each pixel. In fact this methodology has been applied to storing images in a compact manner by reducing this redundancy [Burt 83b] without losing any relevant information.

The HDC was used as a recursive process with d_1 being equal to one and two (for images containing very large highlights larger values of d_1 would have to be

used). This means that a value of two for d_1 finds the regions for which the slope is equal to one or greater since our convolution mask with d_1 equal to one found regions whose slopes were two or greater (one octave difference between the two cases). The two images were then ORed together to form one image with the complete concavity information for the different sized masks.

The image of the concave down sections found by the previous algorithm was then ANDed with the image containing the zero–crossing chains. The result is an image containing highlight chains for the original monochromatic image. There are still two other monochromatic images left from the original RGB images to process. Color is used to corroborate the data [Kanade 81] so that we have a 'true' highlight identification system. The basis for this assertion is that the three color bands must work in conjunction to produce a highlight. It may be expected that a color edge will have a possibility of being interpreted as a highlight in a monochrome image if it is of a ramp shape, but it clearly should not be labeled as a highlight, since it is only a stimulus due to a color change in one of the three color bands. It is therefore necessary to use color to eliminate this confusion. This is of course assuming a white light source.

For example, if the monochrome ramp edge was caused by such a curve in the red band, we would see that the other two bands remained passive across the edge (see figure 4–9). If we had a 'true' highlight you would see different curves (see figure 4–10). After all the RGB images have been processed, their resulting highlight zero–crossing images are ANDed together to give the final zero crossing diagram for the RGB original image (see figure 4–11). The last image processing step is to thin the highlight chains.

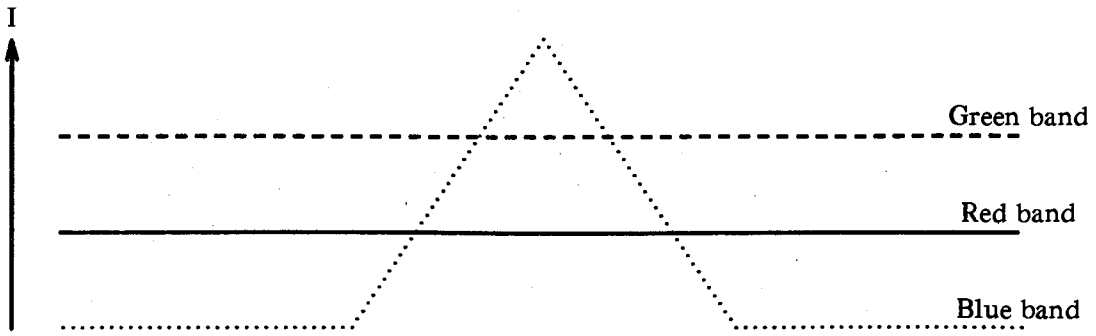
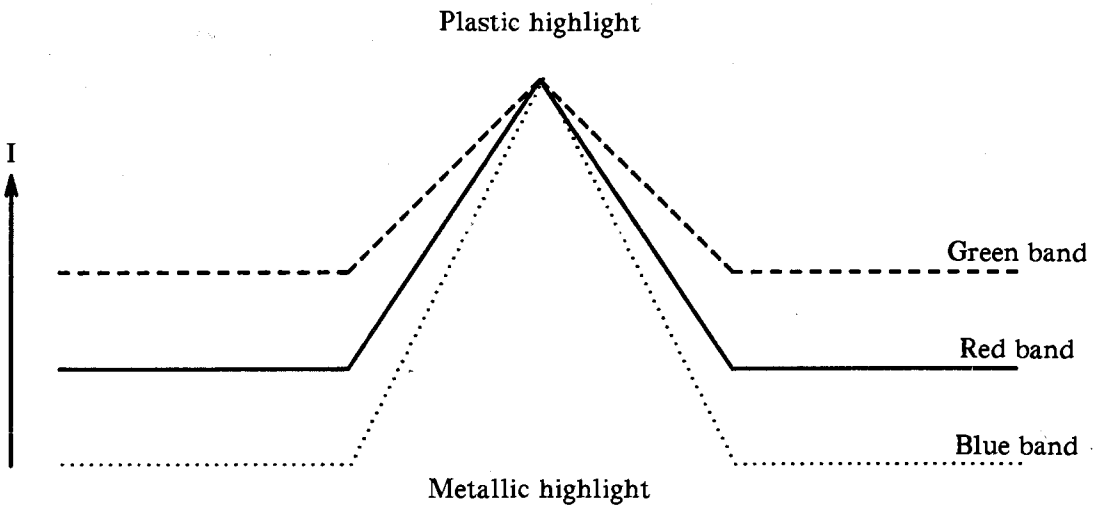


Figure 4-9: Three band intensities



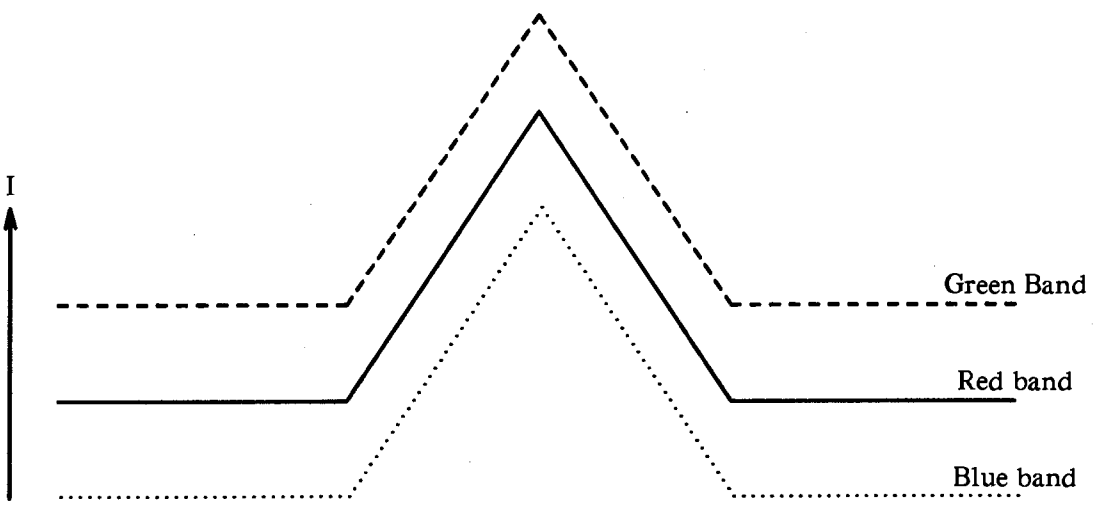


Figure 4-10: Metal & plastic highlights

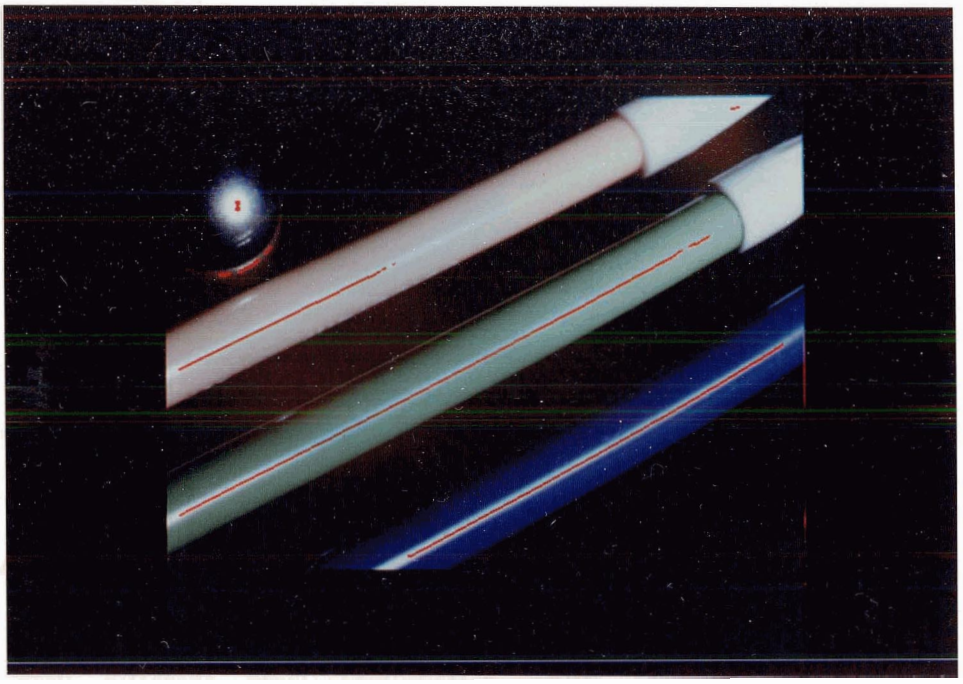


Figure 4-11: Final, fattened, zero–crossing image

4.5. Skeletonization

We now have an image that contains zero–crossings that correspond to highlights. A quandary arises from the desired reduction of the zero–crossing chain's thickness to a one pixel wide chain. This slimming can be achieved by skeletonizing each chain. Davies and Plummer [Davies 81] devised a quasi–parallel algorithm, with six stages, which accomplished this task with a fair degree of efficiency. A modification of their approach was designed to produce a fast skeleton of an image's contents using convolutions by the IIS.

The primary idea of Davies' and Plummer's algorithm is their pixel identification and removal schema. They remove a North point if the window in figure 4–12 appears, with additional constraints supplied by the previous two stages in the

*	0	*
*	N	*
*	1	*

* = Don't care

Figure 4-12: Point removal window

algorithm. The modification to the point removal phases of Davies and Plummer requires no previously computed constraints, and reduces the whole algorithm to one operation (with the omission of their Clean and Purge stages used for noise spur removal).

The new algorithm consists of setting up a 3-by-3 mask with the constraint that every different pixel combination under the mask will give a unique numerical answer. This can be accomplished by using a power-of-two mask as depicted in figure 4-13.

2	4	8
1	0	16
128	64	32

Figure 4-13: Power-of-two mask

Since the center pixel, North in this case, will be removed (set to zero) based on the result of the convolution of this mask with the zero-crossing image, it is irrelevant if that pixel is a one or a zero to begin with. The removal of the center pixel occurs if it is found to be exterior to the desired skeleton, which can be seen in figure 4-14.

The convolution of the power mask with the zero-crossing image will give values between 0 and 255 if the zero-crossing image is binary valued. To delete a pixel

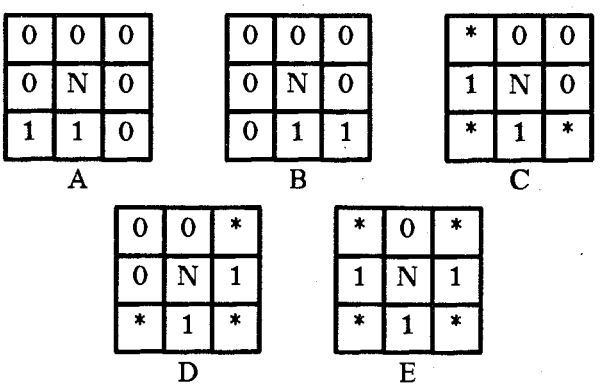


Figure 4-14: Pixel removal diagrams

from the zero–crossing image merely requires checking to see if the corresponding pixel value in the convolution image matches that of one of the values for diagrams A through E. There are thirty-four such values. But this is just for removal of a North pixel. Subsequent South, West and East pixel deletions occur by rotating the convolution mask by the required amount and then doing the same convolution and pixel value examination as was done for the North orientation.

The algorithm continues in the N, S, W, E ... convolution pattern until completion. The algorithm terminates when it is determined that there are not any pixels in the resultant convolution image that have one of the values from diagrams A through E. We now have an image comprised of unit–width pixel chains that represent the center of the highlights in the original image (see figure 4–15). Since the determination of these chains was exclusively by convolutions, we do not know where these chains are, ie. they are not stored in data structures. The next section describes the extraction of the highlight chains from the image and the color analysis.

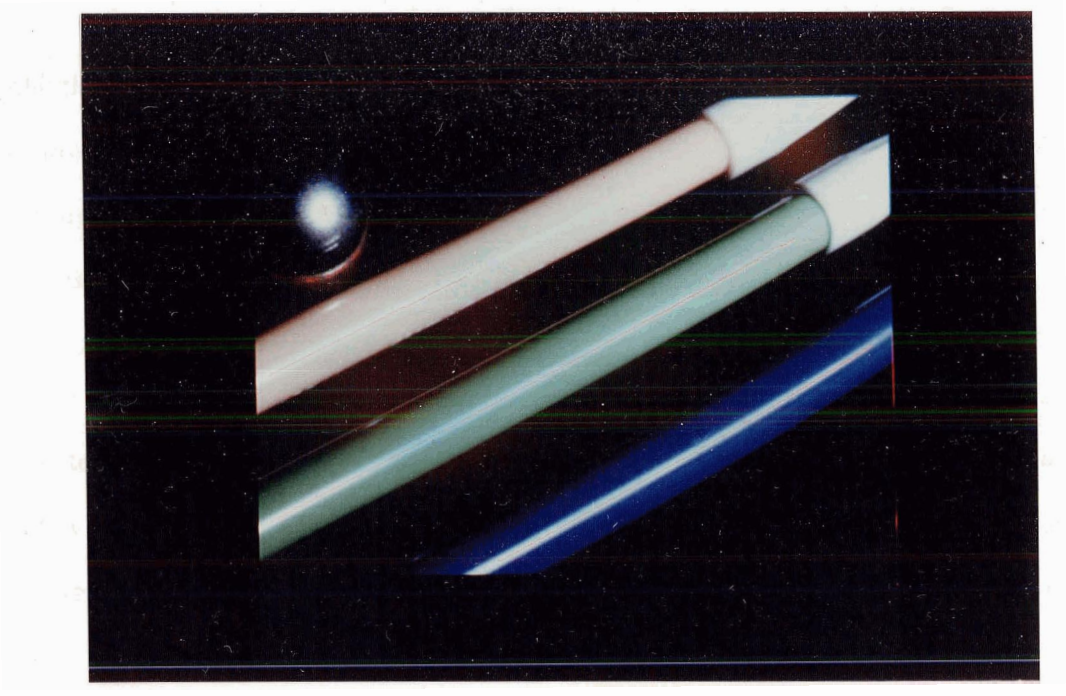


Figure 4-15: Highlight center chain skeleton in red.

4.6. Highlight chain extraction

Having completed all the image processing, we are left with an image that contains the highlight center chains, but we do not know where they are. The values in this image were binarized (chain pixels equal one and all others are zero) so that the algorithm could scan for highlight pixels.

The scanning for highlight pixels proceeds by first horizontally searching the image row by row until one such pixel is found. Once a highlight pixel is located the pixel coordinate is put on a stack and that pixel's value set to zero. The eight neighbors of the pixel are then examined to see if they are highlight pixels and, if so, they are in turn put on the stack. The process continues until the chain is completely removed from the image and stored in a data structure.

Horizontal scanning then continues from where it stopped until finding another highlight pixel or reaching the bottom of the image. Upon reaching the bottom, the process of identifying the locations of highlight chains is completed. The data structure containing all these chains is then passed to a procedure for color analysis.

Since we used the RGB color bands to pinpoint highlights by assuming the characteristics of plastic and metal highlight curves are known, we can thus identify the material that creates the highlight. The plastic's highlight, assuming a white source, is of equal size and intensity in all three color bands, whereas the metal's maintains the original color bias of the material. What this means is that the ratios R/(R+G+B), G/(R+G+B), and B/(R+G+B) for plastic should be equal while for metal they should be constant, but not equal, along the highlight chain. The procedure for color analysis computes the averages of these ratios and decides whether the material is homogeneous, inhomogeneous, or unknown, by the value of that average and its standard deviation.

Since some highlights can be quite small there will only be a few pixels contributing to the color ratio calculation. Homogeneity can be more accurately determined by increasing the number of pixels that contribute to the color ratios. This can be accomplished by examining highlight pixels other than just the central ones. The color shift towards white for colored inhomogeneous material, or color constancy for homogeneous material, can be used as corroborating evidence for the color ratio average when determining material type. The pixels that need to be inspected are those surrounding the highlight center that are part of the highlight

and orthogonal to the local direction of the highlight chain. Orthogonality is important since it determines the direction that will have the correct geometry for the color shift.

In figure 4–16, for example, a color shift does not occur along the major axes of the colored plastic pens.

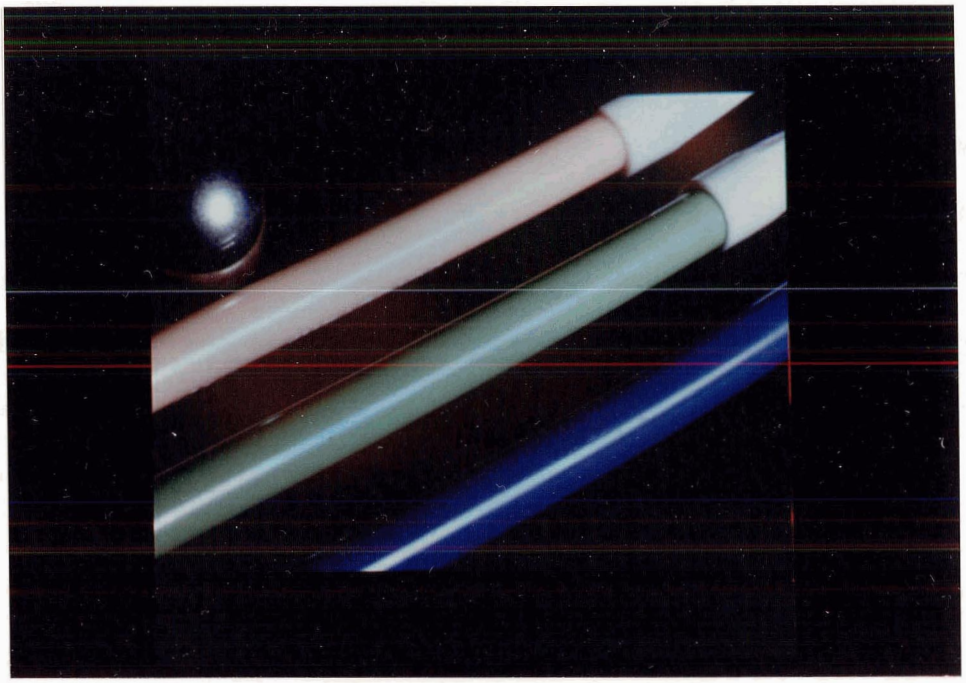


Figure 4-16: Color shift orthogonal to highlight direction

This is because along these axes the surface normals all lie in a plane and have the same direction. The only difference between neighboring pixel intensity values is due to the change in the incident and emittent angles with respect to the surface normal. If the source is far enough away, these values will be constant. Orthogonal to these axes the object pixel points will not have a constant value, rather the object curvature plays a part in determining the contribution of the specular reflection to the total intensity. Since the curvature is changing quite

rapidly along the orthogonal axes the surface normal at each point is directed in a different direction than that of its neighbors. This means that the color shift will be quite pronounced in this direction.

The next section describes the method for finding the local direction and the region surrounding the highlight center.

4.7. Highlight Growing

The highlight pixels that surround the previously located highlight center chain can be found by growing outwards from that central chain in a locally orthogonal direction. To ascertain which direction is orthogonal, the local direction of the highlight center chain is found over a 3–by–3 mask. The growing proceeds while the results of convolving a growing one–dimensional orthogonal mask, with the original image, are increasing.

The previous skeletonizing routine resulted in the highlight center chain being only one pixel wide. This means that by convolving the power–of–two mask in figure 4–13 with the binary valued highlight chain image (1 for highlight pixels and 0 otherwise) and examining the resulting numeric value, we can determine the local direction. Using the eight–connected neighbor model there are four basic directions; North–South(NS), East–West(EW), Southwest–Northeast(SWNE) and Northwest–Southeast(NWSE). For example, if the the convolution result is 17 or 68, the highlight center chain pixel is tagged with a EW or NS flag respectively. Similarly other values demarcate various compass directions. Direction flagging can be accomplished very quickly by setting up a different LUT value for each of the

directions and feeding back the results of the convolution through the LUT to a storage channel. An example of SWNE direction flagging can seen in figure 4–17.

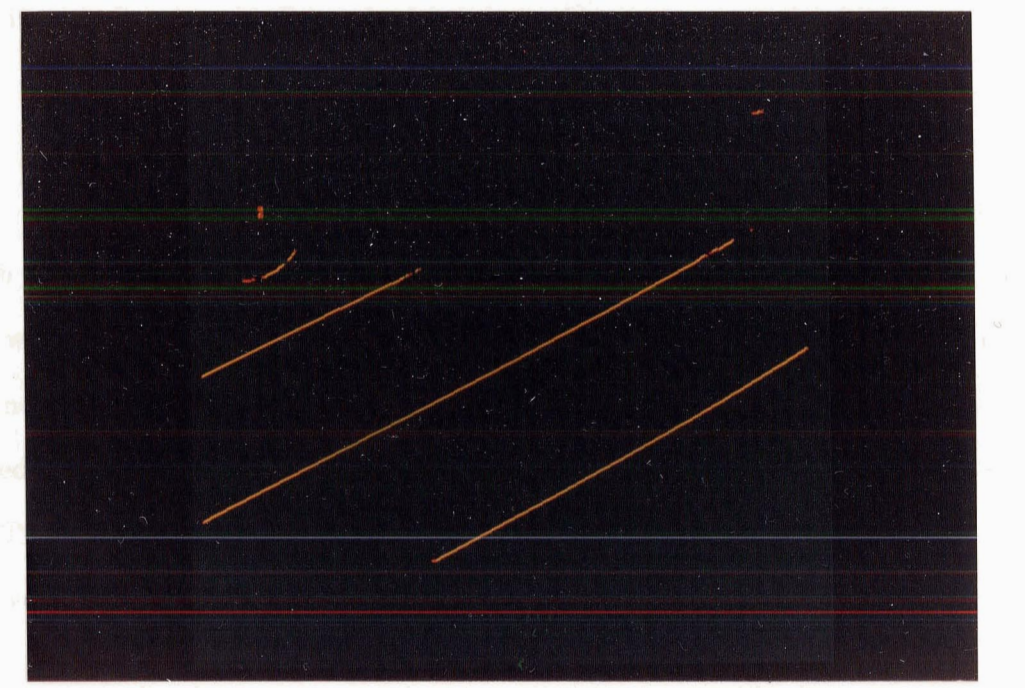


Figure 4-17: SWNE direction flagged highlights.
Flagged highlights are yellow, others are red.

Choosing the orthogonal direction can be impossible for some of the patterns that can arise in a 3–by–3 area, so these pixels receive special treatment.

These pixels are labeled 'blob' pixels since they are without a definite direction. To enable processing they are marked with a flag that states every direction is orthogonal, and thus they are processed for each of the four directions. Once all the center–chain pixels have been tagged for direction, the orthogonal direction highlight growing can proceed.

Orthogonal to the center–chain pixels, which identifies the peak of the highlight, the highlight intensity values decrease until reaching the value of the diffuse

intensity for that surface. Therefore a one dimensional orthogonal mask is set up to calculate the difference between a highlight center pixel and points on either side of it (see figure 4-18 for a horizontal mask).

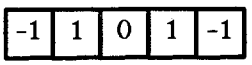


Figure 4-18: Starting orthogonal mask

At each iteration of the algorithm the positive values of the orthogonal mask remain where they are with respect to the highlight center pixel, but the distance to the negative values increases by one pixel. This means that at each stage this calculated value should be increasing if the negative values are overlaying highlight pixels. This is because these pixels should be 'downhill' from the last pixels and thus have smaller values. The algorithm keeps iterating for each highlight pixel point as long as the calculated value is increasing. When the algorithm terminates, the current value for the masksize determines the diameter of the highlight in the orthogonal direction. Since blob pixels are processed for each direction, the current value of the mask size is stored so that it can be compared with the subsequent mask size values. The final mask size chosen for blob pixels will be the minimum of the directional mask sizes. The blob pixel will then be marked with one of the four directions that corresponds to the minimum sized mask.

Of course highlights with small slopes in the orthogonal direction (slopes less than one) would cause the algorithm to terminate prematurely, therefore an average is taken over a few pixels for each negative side lobe, to minimize this possibility. Taking an average over a few pixels also helps to reduce the effects of noise in the image. Since highlights are not usually perfectly symmetric, the

orthogonal direction is divided into two halves, one half on each side of the highlight. The 'new' orthogonal direction convolution proceeds on one half at a time, with the orthogonal mask having only one negative side lobe rather than two (see figure 4–19).

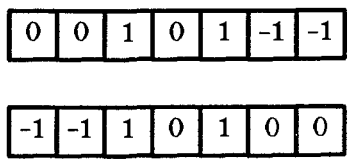


Figure 4-19: Redefined horizontal, orthogonal masks

The highlight diameter will be the minimum of the two orthogonal masks used for each highlight pixel.

We now have each highlight–chain pixel tagged with a highlight diameter and a direction (see figure 4–20 for a scene with the highlight diameters used to generate disks around the highlight center pixels). We can use this information to calculate the color ratios across the highlight. An additional use for this information could be the marking of the highlight pixels for subsequent removal from the image. This would make some image processing or identification tasks easier without these confusing illumination effects.

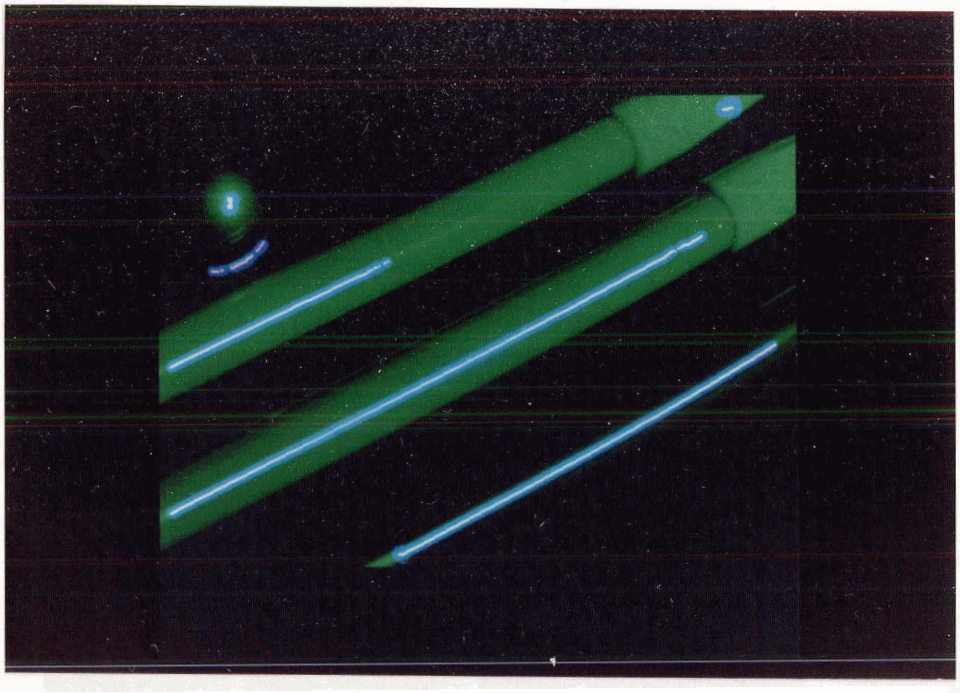


Figure 4-20: Highlight diameters used to generate disks

Chapter 5

Results

This section illustrates the results of applying the highlight detection algorithm to various scenes.

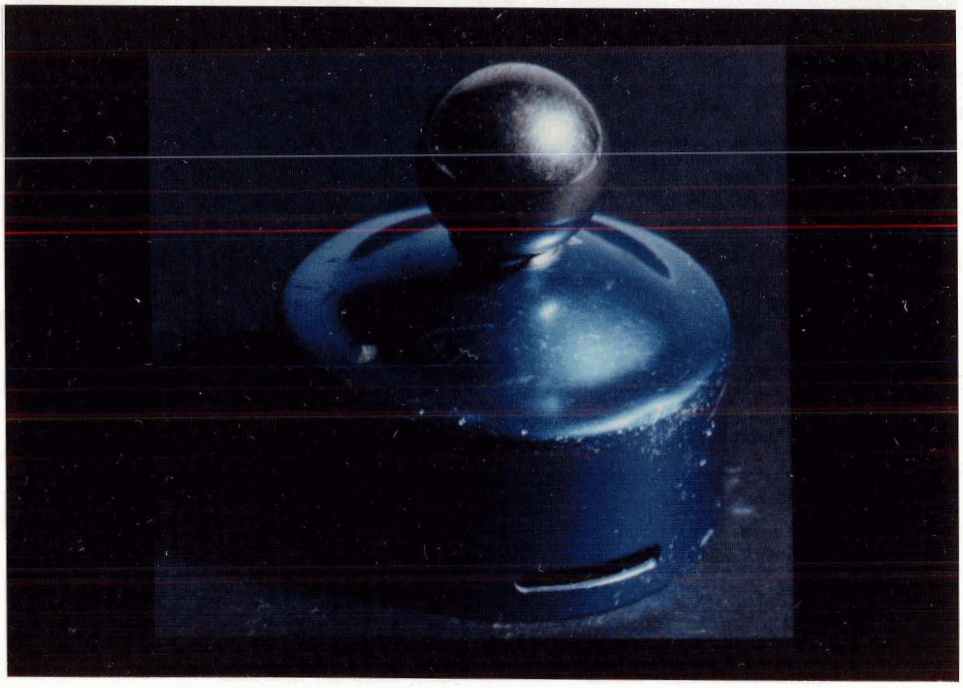


Figure 5-1: The original image.
Metallic fishing reel top and a ball bearing.

Figure 5–1 is a scene containing two metallic objects. There is specular reflection from both objects, as well as the black crepe paper that constitutes the background. There is an abundance of noise in both the background and objects themselves, which, in conjunction with the surface markings, makes the task of highlight detection quite difficult.

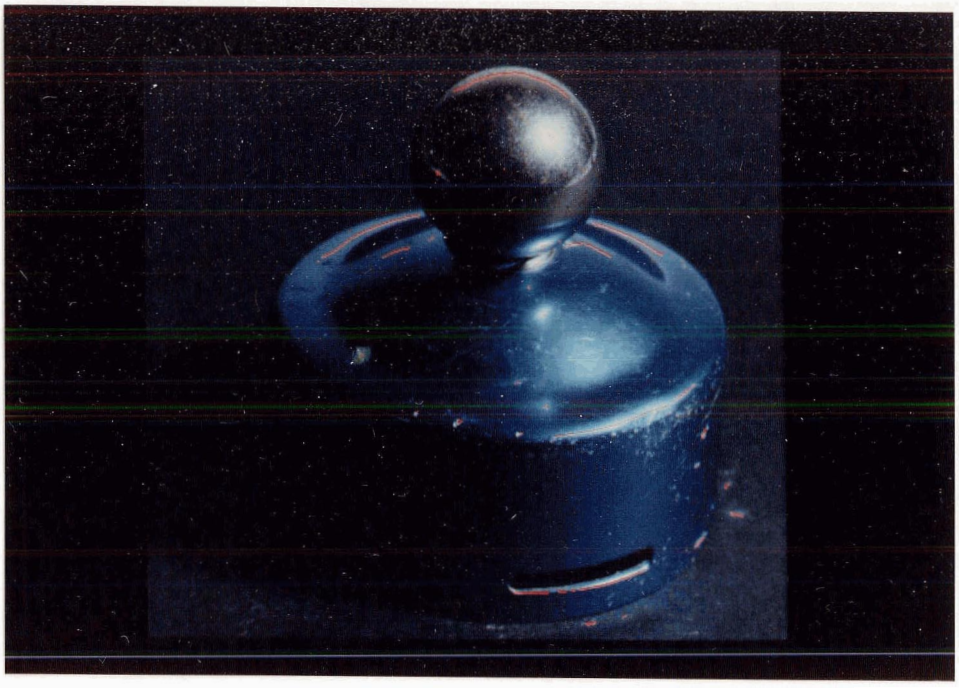


Figure 5-2: The highlight image.
Metallic fishing reel top and a ball bearing.

Figure 5--2 shows the highlights found from applying the highlight detection algorithm to the original image. The detected highlights are shown in red, superimposed over the original image. The very large highlight in the central portion of the fishing reel was not detected because it has such a small gradient across it. The algorithm detects all the highlights regardless of the noise.

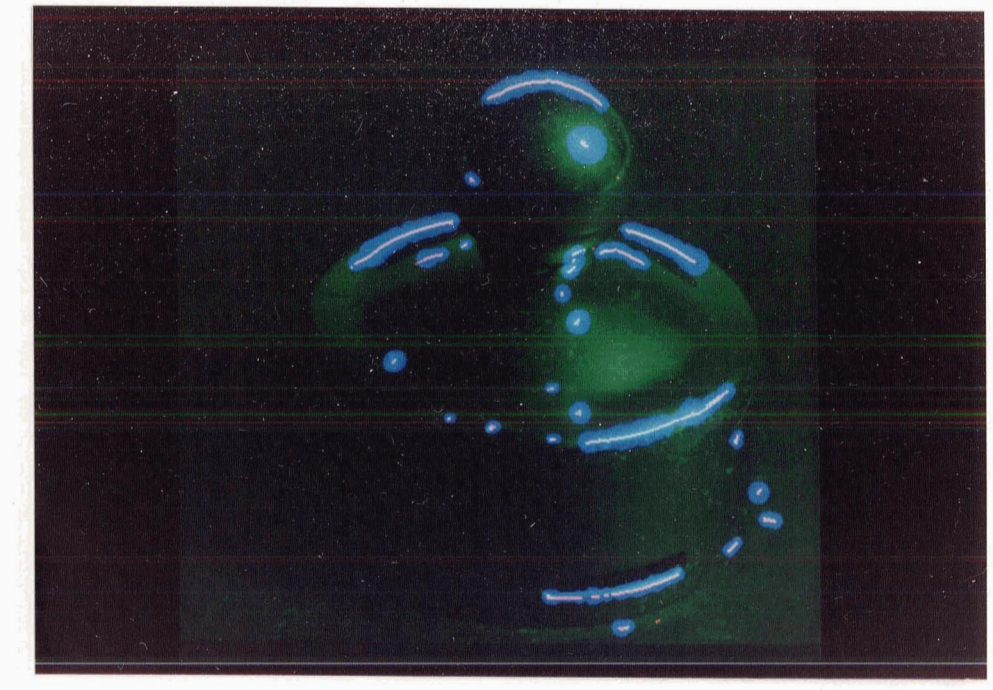


Figure 5-3: The highlight region image.
Metallic fishing reel top and a ball bearing.

Figure 5–3 shows the results of using the diameter information, about each highlight–center–chain pixel to generate a solid disk about that pixel. The detected highlights are shown in red, superimposed over the original image in green. The diameter of the highlights are shown by blue disks with their centers along the highlight center chain. This figure gives an indication as to the extent of the highlights. However, symmetric disks do not accurately portray the directional information that the central highlight pixels had been tagged with during the orthogonal direction processing. A more accurate covering method would involve joining the points at the extremes of the diameters in a closed, smoothed curve. Note that highlights are not necessarily convex.

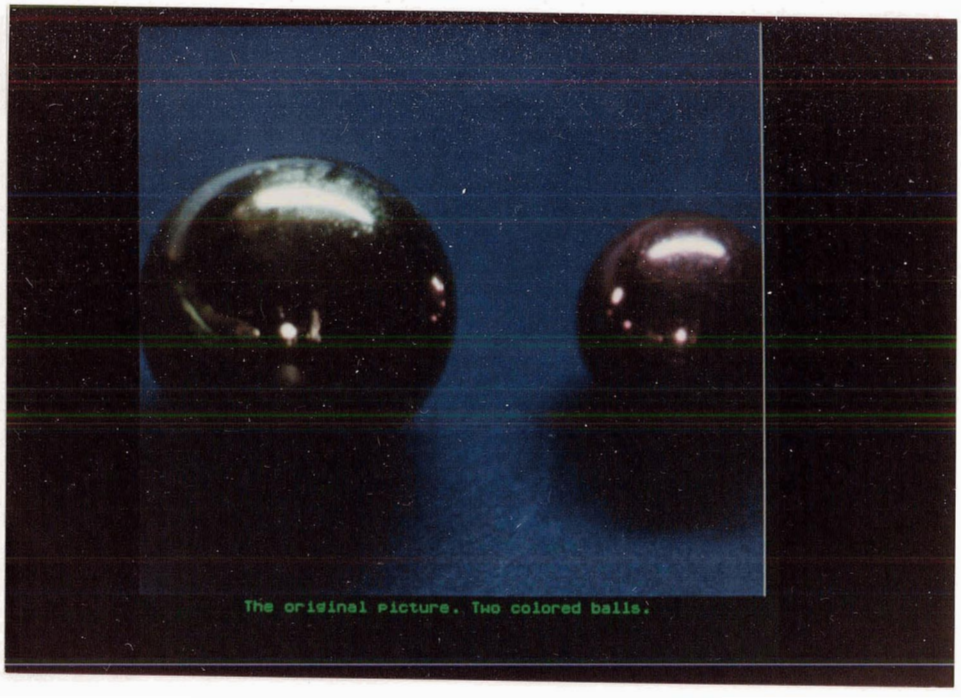


Figure 5-4: The original image.
Two christmas ornaments.

Figure 5–4 contains two spherical, metallic, christmas ornaments. One sphere is gold and the other is red. The light sources were a florescent bar light and an incandescent bulb. The bar light source creates large, curved highlights, while the bulb creates circularly symmetric ones. Note the 'cross–specular' reflection occurring where one ornament creates a highlight on the other by reflecting the light source. These types of reflectances would be extremely confusing if highlights were to be used to determine source location. The color of cross–specular reflection makes the determination of source color as perplexing as that of location.

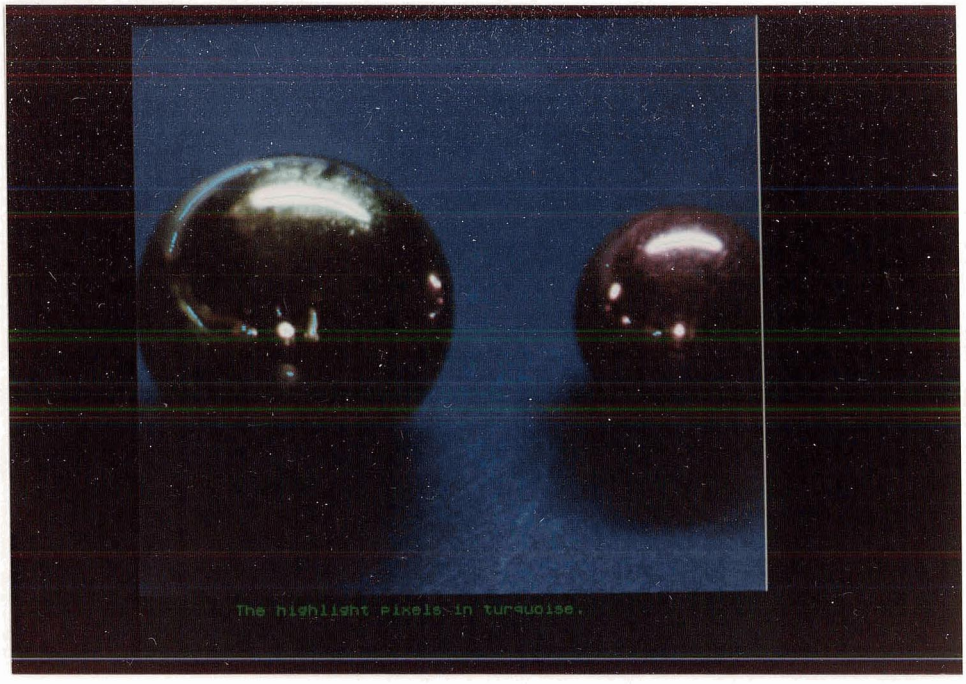


Figure 5-5: The highlight image.
Two christmas ornaments.

Figure 5–5 shows the highlights detected by the algorithm. The detected highlights are shown in turquoise. Figure 5–6 shows the results of highlight region growing. The detected highlights are shown in red, superimposed over the original image in green. The highlight diameters are shown by blue disks with their centers along the highlight center chain.

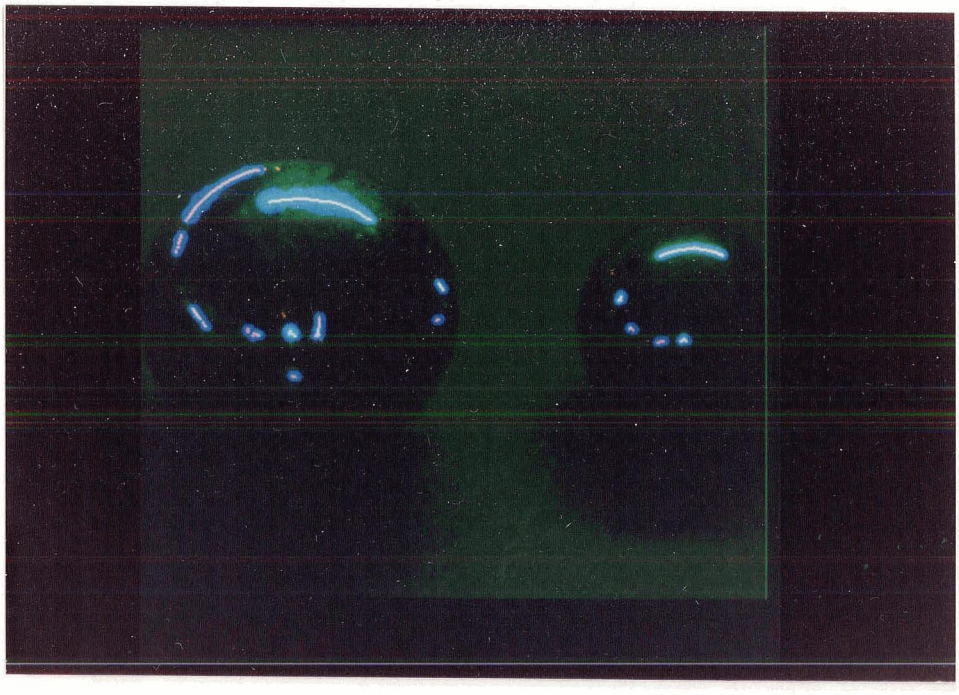


Figure 5-6: The highlight region image. Two christmas ornaments.

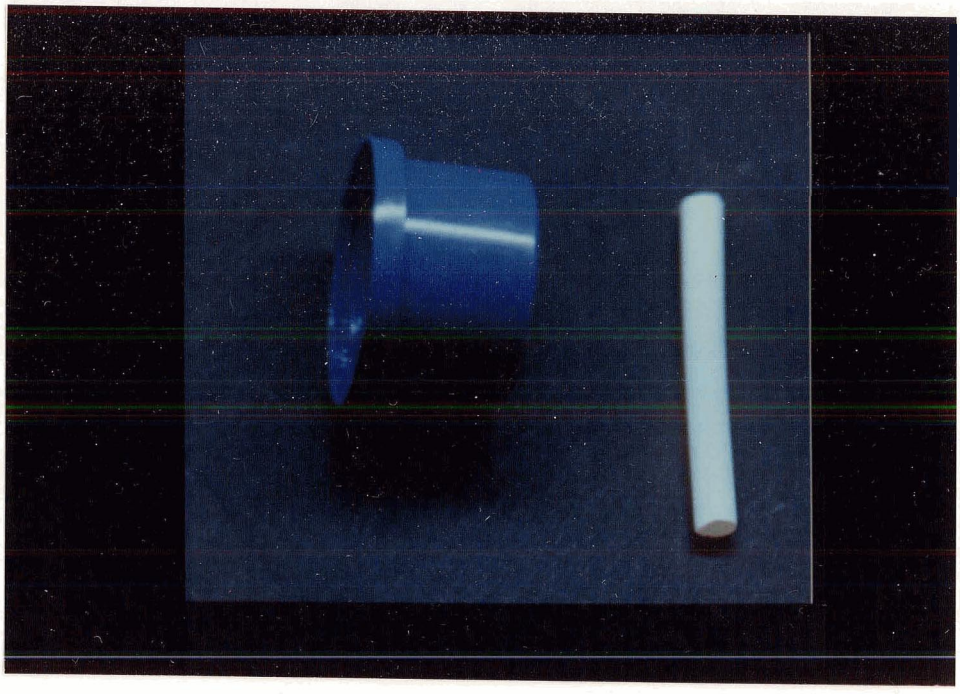


Figure 5-7: The original image.
A blue plastic aerosol cap and a piece of chalk.

Figure 5–7 is a scene containing one specularly reflecting surface and one completely matte reflecting surface. The light source in this case was a florescent bar.

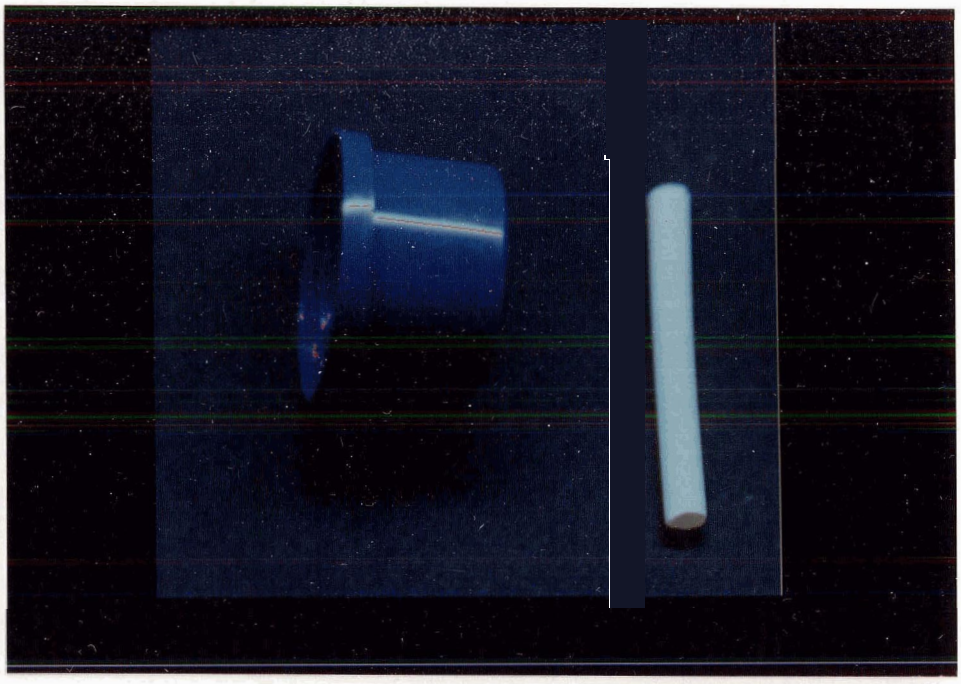


Figure 5-8: The highlight image.
A blue plastic aerosol cap and a piece of chalk.

Figure 5–8 shows the highlights found from applying the highlight detection algorithm to the original image. The detected highlights are shown in red, superimposed over the original image. The algorithm does not find a highlight for the matte piece of chalk. Figure 5–9 shows the results of using the diameter information, for each highlight center chain pixel, to generate a solid disk about that pixel. The detected highlights are shown in red, superimposed over the original image in green. The highlight diameters are shown by blue disks with their centers along the highlight center chain.

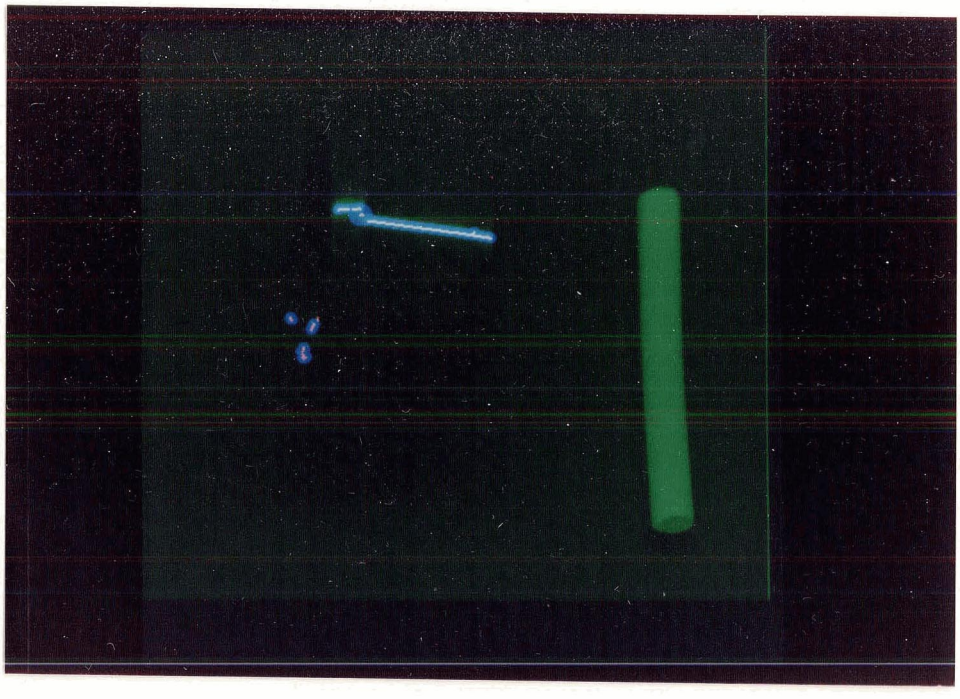


Figure 5-9: The highlight region image.
A blue plastic aerosol cap and a piece of chalk.

Chapter 6
Extensions

All the theory presented thus far depends upon the light being a point source. Investigation could be done into changing the theory to include extended sources. This may aid in solving one of the problems that Forbus found when trying to apply the S-operator, that is, the effects on intensity across a surface (plane) due to absorption and beam spread. It may turn out that this problem cannot be circumvented in the quest for a highlight detection algorithm, since point sources are basically theoretical, and everyday images deal with somewhat extended sources anyway.

Although we may be able to extend the theory to incorporate extended sources, how do we know if the highlight is caused by an extended source, or not? It may be possible, by using the size and shape of the highlight, to derive this information. It also seems that this information, while constraining the source properties, also has to constrain the surface properties as well. For example if we have a bar light source specularly reflected by a sphere, there seems no way to avoid the fact that it would appear as a curved bar (see figure 6–1). This curved highlight, present in our intensity data, therefore tells us something about both the light source and surface. Observations have been made that "a localized highlight is indicative of an elliptic surface, while a linearly extended one is

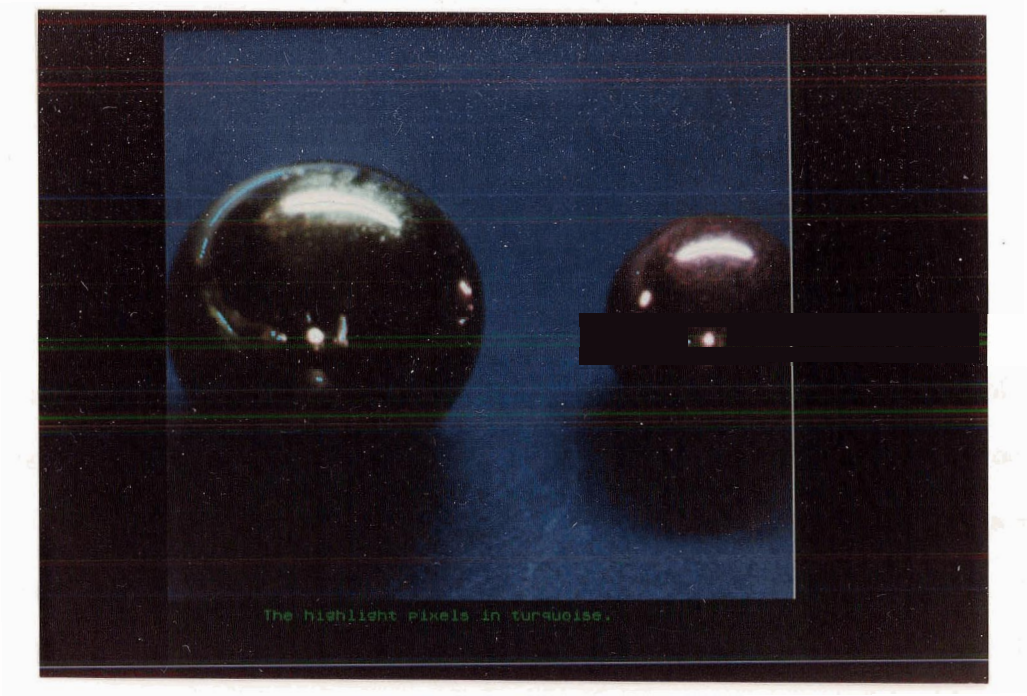


Figure 6-1: Highlights from a bar light source

indicative of a cylindrical surface" [Barrow 81], although this is generally only true for point light sources. There might be some confusion of course, ie. a curved bar light source reflected from a planar surface, but if we can make some general assumptions about the types of light sources involved, highlights still appear particularly useful.

It was hoped at the outset of this thesis to be able to ascertain material types by examining the colors of the detected highlights. However, the inability to produce colored highlights for colored metallic surfaces underlines the need for better color processing techniques. The capability is there for material type discrimination, but the current digitization process loses too much information to make this possible.

It seems appropriate to conclude that there is a wide array of applications for highlights.

Chapter 7
Conclusion

An algorithm is developed in this thesis which locally processes scenes of various objects to determine areas of spectral reflectance or highlights. The algorithm is based on the separability of the spectral from the diffuse reflectances by differential methods. Once the highlight center chains are detected by the algorithm, they are expanded to highlight regions by region growing in a direction orthogonal to the local orientation of the highlight. At the conclusion of the algorithm, the information known about each highlight includes location, size and direction. Thus the algorithm provides information that a Computer Vision system must make use of when analyzing, or understanding, a scene. In addition, the information this algorithm provides can be used as a preprocessor to image processing algorithms that rely on predetermined areas of spectral reflectance. It can also be used to identify areas of spectral reflectance so that they can be removed from the scene. This eliminates illumination peculiarities which might confuse later/other algorithms that do pattern matching.

The fact that the algorithm is successful using moderately unconstrained images is important since it decreases the gap between the world that the computer can now understand and the extremely complicated one in which we live.

Appendix A
Ullman's Source Detector

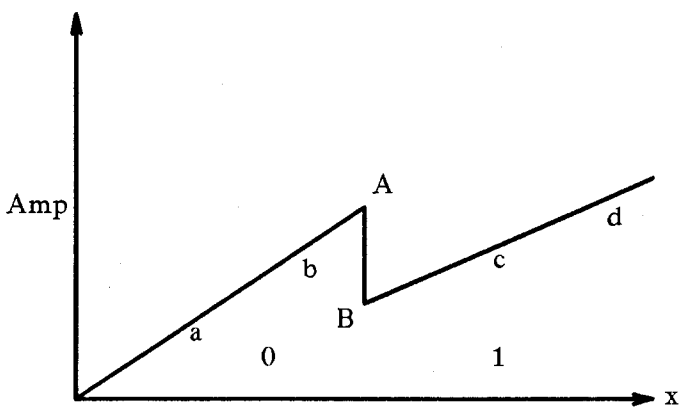


Figure A-1: One Dimensional Intensity Map

Let I_p be the incident light intensity at a point p.
r_p be the reflectance of the surface at a point p.
A_p be the resultant amplitude measurement at a point p.
S_s be the slope of the surface s.
→ mean 'which can be redefined as'

$Amp_a = I_a * r_a$
$Amp_b = I_b * r_b$

but $r_a = r_b \rightarrow r_0$ since a surface has constant reflectance.

Let $\Delta A_0 = Amp_b - Amp_a$
$\Delta I_0 = I_b - I_a$
$\Delta I_1 = I_d - I_c$

but $\Delta I_0 = \Delta I_1$ since this is the illumination gradient with respect to the x direction and the points on each surface have an equivalent Δx.

therefore $\frac{\Delta Amp_0}{\Delta Amp_1} = \frac{r_0}{r_1}$

so $\frac{S_0}{S_1} = \frac{r_0}{r_1}$

we also have that $I_A = I_B$ since they are adjacent points

so $Amp_A = \frac{r_0}{r_1} \times Amp_B \rightarrow \frac{S_0}{S_1} = \frac{Amp_A}{Amp_B}$

Appendix B

Sigma vs. Masksize for Gaussian Operators

By looking at the two dimensional Gaussian, in alignment with one of the axes, we can get a perspective of how large a mask needs to be to accurately reflect the choice of σ. The two dimensional Gaussian is given by:

$$G(x,y) = e^{-(x^2 + y^2)/2\sigma^2}$$

and its x axis alignment by:

$$G(x,0) = e^{-x^2/2\sigma^2}$$

For positive x greater than 3.5σ the value of the second expression becomes negligible and thus the masksize needs to be at least 7σ when you also consider negative values of x.

The first directional derivative of the x axis aligned Gaussian becomes:

$$G'(x,0,\theta) = \frac{-x\cos\theta}{\sigma^2} e^{-x^2/2\sigma^2}$$

and by letting θ equal zero we can determine the first differential masksize. Like the previously determined masksize, choosing a positive x value of 3.5σ results in a negligible value for the first directional derivative expression as long as σ is greater than 3. A positive x value of 4.5σ gives negligible values for σ greater than 1.

Therefore we can see that the masksize is dependent on the choice of σ and is different for each differential degree.

Appendix C

Convolution inaccuracies of the IIS

The Laplacian of the Gaussian is given by:

$$\nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}$$

but lets assume that the y component is zero, and look at the profile of the resultant in the x direction.

$$\nabla^2 G(x,0) = \frac{e^{-x^2/2\sigma^2}}{\sigma^2}[x^2/\sigma^2 - 2]$$

By looking at the profiles of the two curves, $\sigma = 2$ and 4, shown in figure C–1, we can see the relative heights of the positive wings and negative center sections. If the maximum negative value for these curves was one, then it is easy to see that the positive values would be far smaller. The reason why this is important when considering the choice of masks to determine concavity is that the sum over the whole two–dimensional mask should be zero when applied to an area in the image with constant intensity. Unfortunately, this will not be true when this mask is used in conjunction with an integer array processor.

For example, if the maximum positive value was 0.25 for this mask, then the result of overlaying a pixel with intensity less than three would be zero, whereas the maximum negative value would give –1. Therefore we can see how the problem arises. Every time a mask value is convolved with an image intensity

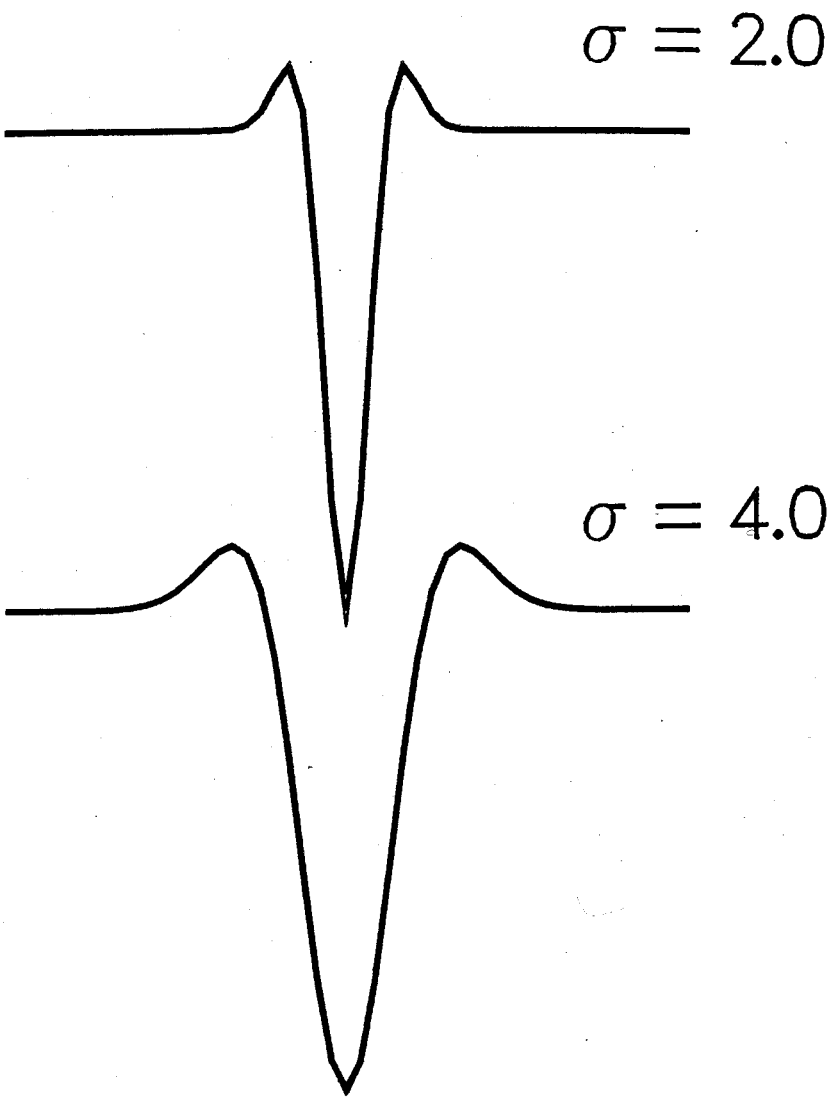


Figure C-1: Profiles of the Laplacian of the Gaussian

value it is converted to an integer and stored. This could be considered a very gross round–off error. It is not that the positive and negative convolution mask values are treated any differently, but only that their relative magnitudes cause the results to be biased towards the larger negative values. This example is a very simplistic one since the majority of the positive values are considerably smaller than 0.25 which means that even bigger image intensity values give a zero result during the convolution.

References

[Babu 85] Babu, M.D.R., Lee, Chia-Hoang, Rosenfeld, A.
Determining Plane Orientation from Specular Reflectance.
Pattern Recognition (1), May, 1985.

[Barrow 81] Barrow, H.G., Tenenbaum, J.M.
Artificial Intelligence 17.
North Holland, 1981.

[Beck 72] Beck, J.
Surface Color Perception.
Cornell University Press, 1972.

[Bryant 83] Bryant, E.C.
Computer Interpretation of Computer Tomography Scans of Sawlogs.
Master's thesis, Simon Fraser University, August, 1983.

[Burt 83a] Burt, P.J.
Computer Graphics and Image Processing 21.
Academic Press, 1983.

[Burt 83b] Burt, P.J.
The Laplacian Pyramid as a Compact Image Code.
IEE Transactions on Communications 3(4), April, 1983.

[Cook 82] Cook, R., Torrance, K.
A Reflectance Model for Computer Graphics.
ACM Transactions on Graphics 1(1):7-24, January, 1982.

[Davies 81] Davies, E.R., Plummer, A.P.N.
Thinning Algorithms: A Critique and a New Methodology.
Pattern Recognition 14(1-6), , 1981.

[Forbus 77] Forbus, K.
Light Source Effects.
MIT AI Lab Memo (422), May, 1977.

[Hall 83] Hall, R., Greenberg, D.
A Testbed for Realistic Image Synthesis.
IEEE CG & A , November, 1983.

[Horn 75] Horn, B.K.P.
Image Intensity Understanding.
MIT AI Lab Memo , August, 1975.

[Kanade 81] Kanade, T.
Artificial Intelligence 17.
North Holland, 1981.

[Marr 75] Marr, D., Hildreth, E.
Theory of Edge Detection.
MIT AI Lab Memo (518), April, 1975.

[Marr 82] Marr, D.
Vision.
W.H. Freeman and Company, 1982.

[Phong 75] Phong, B.T.
Illumination for Computer Generated Images.
Commun. ACM , June, 1975.

[Powers 79] Powers D.L.
Boundary Value Problems.
Academic Press, 1979.

[Pratt 78] Pratt W.K.
Digital Image Processing.
John Wiley and Sons, 1978.

[Prewitt 70] Prewitt, J.M.S.
Object Enhancement and Extraction.
Picture Processing and Psychopictorics.
Academic Press, 1970.

[Sears 58] Sears, F.W.
Optics.
Addison–Wesley Publishing Company, 1958.

[Shafer 84] Shafer, S.A.
Using Color to Separate Reflection Components.
Carnegie-Mellon Technical Report (136), April, 1984.

[Thrift 82] Thrift, P., Chia-Hoang Lee.
Using Highlights to Constrain Object Size and Location.
IEEE Publication , 1982.

[Ullman 75] Ullman, S.
On Visual Detection of Light Sources.
MIT AI Lab Memo (333), May, 1975.

[Wyszecki 75] Wyszecki, J.
Color in Business, Science and Industry.
John Wiley and Sons, 1975.