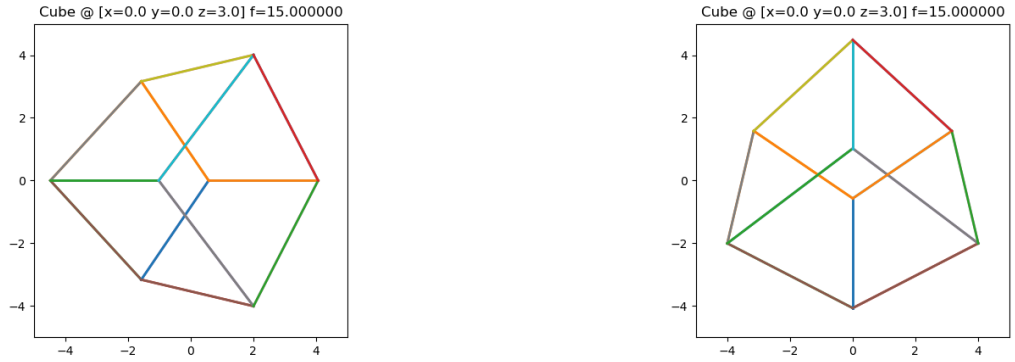
Computer Vision, CS583, Homework 1

Tony Stanell

Drexel University, Fall 2024

1 Camera Projection Matrix

1. **cube.gif** is included in the submission.



(a) rotX($\pi/4$), followed by rotY($\pi/4$)

(b) rotY($\pi/4$), followed by rotX($\pi/4$)

Figure 1: Comparison of rotation orders

2. Are 3D rotation matrices commutative? **No**, they are not commutative. This is demonstrated by the different outputs depending on the order in which the matrices are applied.

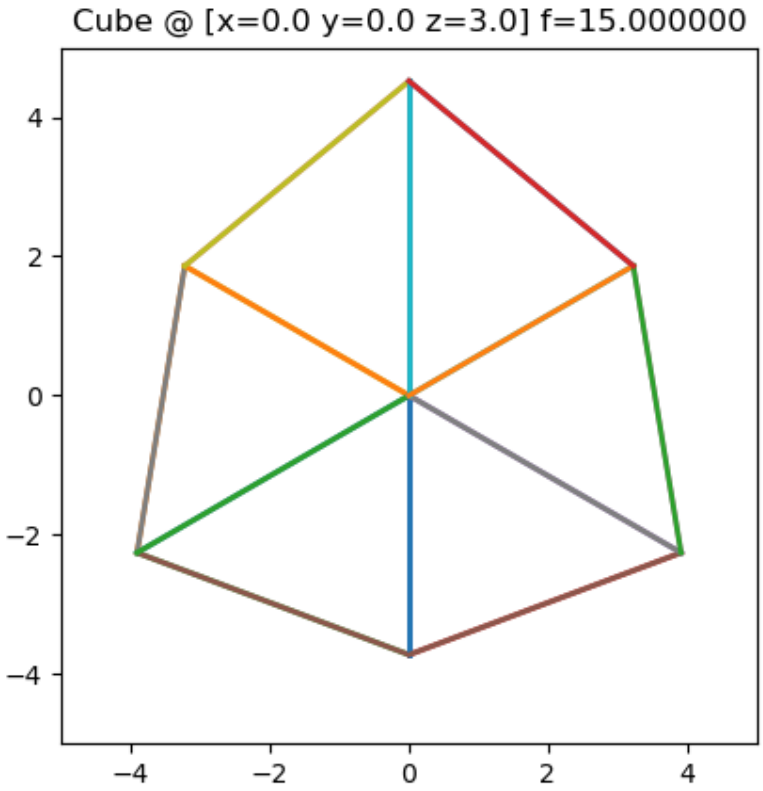


Figure 2: Question 3 answer: When the diagonal of the cube is projected to a single point, it is often referred to as Isometric Projection. The order of rotation and parameters are: rotate Y by $\pi/4$ and then rotate X by $\arctan(1/\sqrt{2})$. As discussed in [1], ".. imagine a cube with sides of length 2, and its center at the axis origin, which means all its faces intersect the axes at a distance of 1 from the origin. We can calculate the length of the line from its center to the middle of any edge as $\sqrt{2}$ using Pythagoras' theorem. By rotating around the vertical axis by 45 degrees we arrive at the opposite edge having length 1 (horizontal axis) and the adjacent edge having length $\sqrt{2}$." When we adjust by the arctan (angle of tangent), this is rotating around the X axis until it becomes equal with Z. My parameters for the cube's scale were taken from the given example file: $t[0, 0, 3]$ and $f = 15$ - this positions and adjusts the camera for a clean view of the overlapping diagonals.

3.

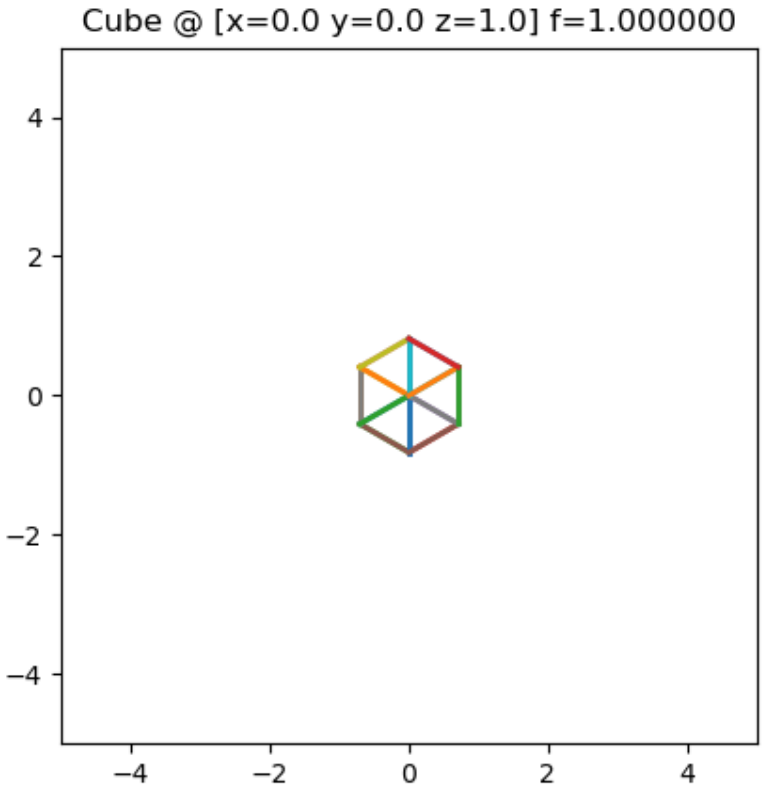
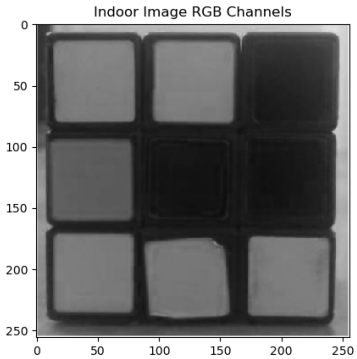


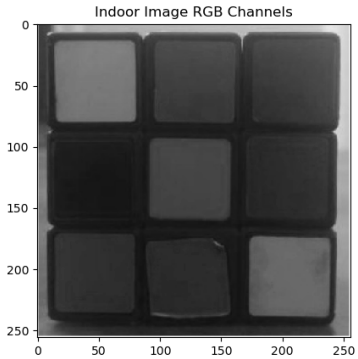
Figure 3: Question 4 answer: This is the orthographic projection of the same cube from question 3. The difference is that we are no longer considering z when rendering, thus making there no perception of depth. In doing so, parallel lines are maintained.

4.

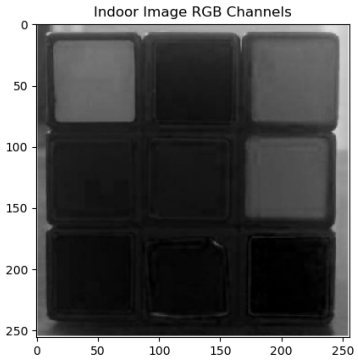
2 Color Spaces and Illuminance



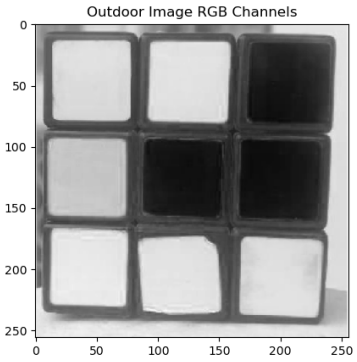
(a) Indoor, Red Channel



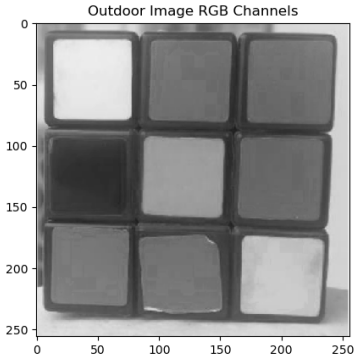
(b) Indoor, Green Channel



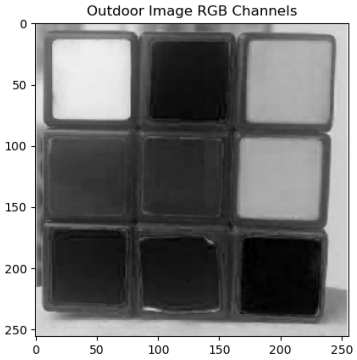
(a) Indoor, Blue Channel



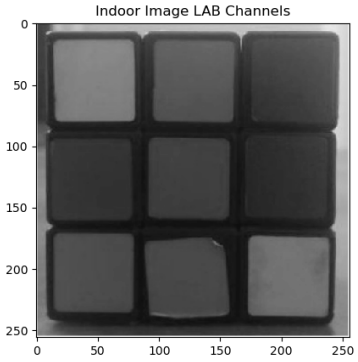
(b) Outdoor, Red Channel



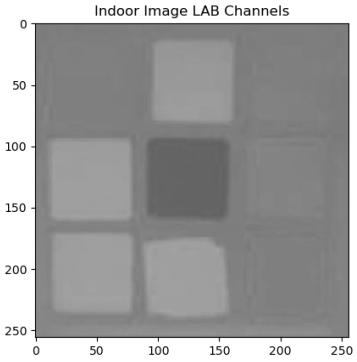
(a) Outdoor, Green Channel



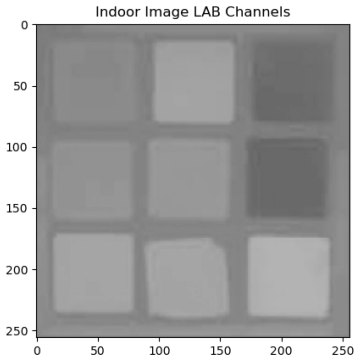
(b) Outdoor, Blue Channel



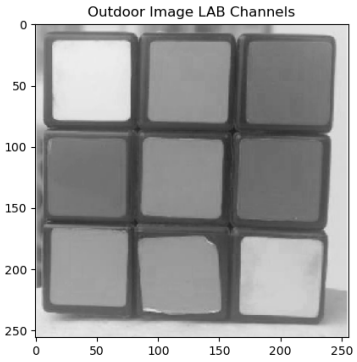
(a) Indoor, Luminance Channel



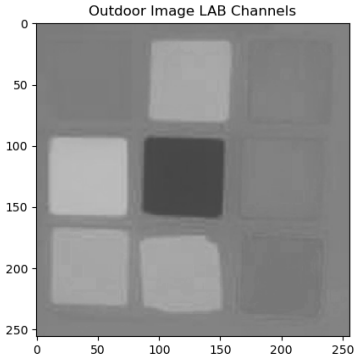
(b) Indoor, A Channel



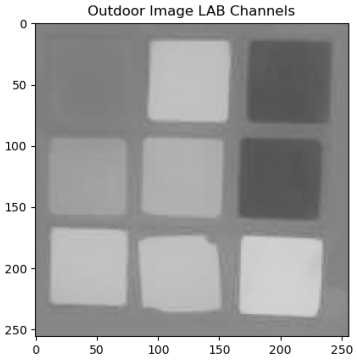
(a) Indoor, B Channel



(b) Outdoor, Luminance Channel



(a) Outdoor, A Channel



(b) Outdoor, B Channel

1.

2. **How do you know the illuminance change is better separated in LAB color space?**
   The LAB color space is superior to RGB channels with regards to illuminance change because of the dedicated channel for illuminance, "L". This dedicated channel separates the changes in light from the changes in color, resulting in a more responsive system to illuminance changes.

3 Image Filtering

All images have been normalized within the range of gray-scale values 0-255 to improve visual quality.

(a) (b) (c)

Figure 10: Image patches

1. (a)

   (b) The objects gained robustness to changes in illumination by converting to gray-scale and then being normalized across the mean and unit variance. This made lighting invariant and thus resulted in a good descriptor for applications where lighting can change significantly. However, the images don't make good descriptors for changes in the object's pose or scale as we are lacking any other significant information pertinent to those applications.

2. (a) Sequential convolutions of vertical and horizontal 1D Gaussian filters represent evenly decreasing values in both directions from the center point (maximum at center). When points share this symmetric distribution across two separate dimensions (horizontal being x, and vertical being y) within a given integral, this brings us to the nature of the 2D Gaussian filter, where points have a radially symmetric distribution.

   The relationship between the 2D and 1D Gaussian filters is directly tied to several mathematical properties of separability within the 2D function:

   $$G(x, y) = g(x) \cdot g(y)$$

   This property is present given the Euler number:

   $$e^{-\frac{x^2+y^2}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} \cdot e^{-\frac{y^2}{2\sigma^2}}$$

   as well as the constant term:

   $$\frac{1}{2\pi\sigma^2} = \left(\frac{1}{\sqrt{2\pi}\sigma}\right) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma}\right)$$

   The one-dimensional Gaussian function is given by:

   $$G(x) = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{x^2}{2\sigma^2}}$$

   where σ controls the spread of the distribution. This function produces a bell-shaped curve. After applying the property of separability as noted above, the two-dimensional Gaussian function is expressed as:

   $$G(x, y) = \frac{1}{2\pi\sigma^2}\, e^{-\frac{x^2+y^2}{2\sigma^2}}$$

   which produces a bell-shaped surface. We can see that the variance is represented by σ^2 in both 1D and 2D equations. The 1D variance is spread equally in one direction while the 2D variance is split between the horizontal and vertical axes. Mathematically, this relationship of the variances can be described as follows:

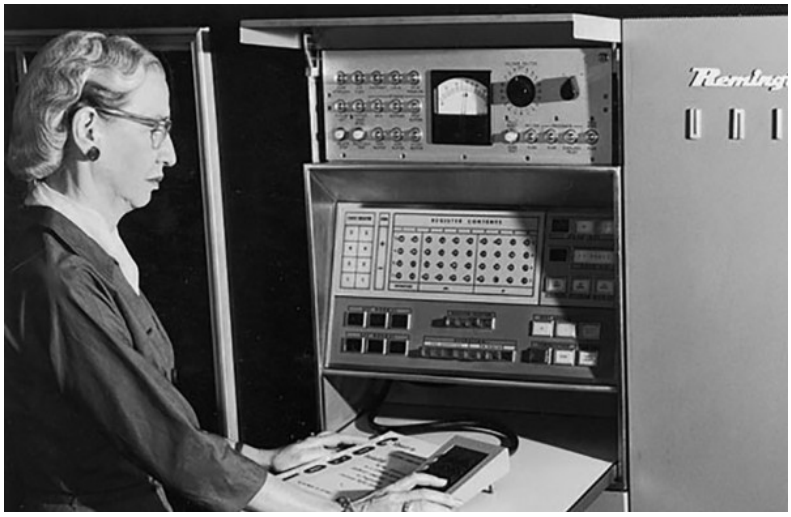
- **1D Gaussian spread or variance**:

$$\text{var}[axis] = \sigma^2$$

- **2D Gaussian spread or variance** is represented as a matrix:

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

Combining the property of separability and the relationship between the variances in both 1D and 2D Gaussian kernels, we can conclude that sequential convolutions of vertical and horizontal kernels result in the same output as the 2D kernel itself.



(a) Original image.

(b) Image with Gaussian blur applied.

Figure 11: Gaussian Filtering Effect

(b) Gaussian filtering applies an even smoothing, or blur, effect across the image.

3. (a) The derivatives for a given image I can be defined as below using central difference approximation:

$$I_x(x,y) = \frac{\partial I}{\partial x}(x,y) \approx \frac{1}{2}\left(I(x+1,y) - I(x-1,y)\right)$$
$$I_y(x,y) = \frac{\partial I}{\partial y}(x,y) \approx \frac{1}{2}\left(I(x,y+1) - I(x,y-1)\right)$$

We can represent the weights from the central difference approximation along the x axis as

$$\frac{1}{2}[1, 0, -1]$$

It is important to note that in my solution, the convolution function handles kernel flipping internally, eliminating the need to manually flip the Sobel kernels before computing the gradient.

The first position of 1 in the array is the weight of the pixel to the right of the center, the second position (center) is ignored, and the -1 adjusts the weight of the pixel to the left of the center. Combining this into the convolution function results in the equation across each row:

$$I_x(x-1,y) \times 1 + I_x(x,y) \times 0 + I_x(x+1,y) \times (-1) = I_x(x-1,y) - I_x(x+1,y)$$

which captures changes in the vertical gradient, interpreted as a horizontal edge. This same concept is then applied in the other direction, capturing changes in the horizontal gradient and thus finding a vertical edge. The horizontal and vertical kernels using the central difference approximation are applied to the image separately and then combined into a single image by calculating the gradient magnitude as:

$$G = \sqrt{G_x^2 + G_y^2}$$

(a) Image with edge detection.

(b) Image with edge detection and Gaussian blur applied.

Figure 12: Edge Detection Comparison

(b) The main difference between these two images is the noise within the image. Applying Gaussian blur via convolution reduces image noise and effectively smoothens the appearance of the image. This is most noticeable in areas with more lines. Further testing could include different levels of σ.

4. (a) Convolution kernels have been described as $I_x = I * k_x$ and $I_y = I * k_y$. Since convolution is a linear operation across the image matrix, we can use the associative property and say that given the existing application of the Gaussian kernel, we can add another pass with Sobel operators as below:

$$(I * k_{\text{gaussian}}) * G_s = I * (k_{\text{gaussian}} * G_s)$$

where G_s represents the Sobel operators. To show that using convolution again on the Gaussian-filtered image with these operators produces good approximate derivatives, we should examine the structure of the operators themselves.

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

The above kernel approximates the derivative by taking the difference between the left and right sides of the kernel (changes in the x-direction).

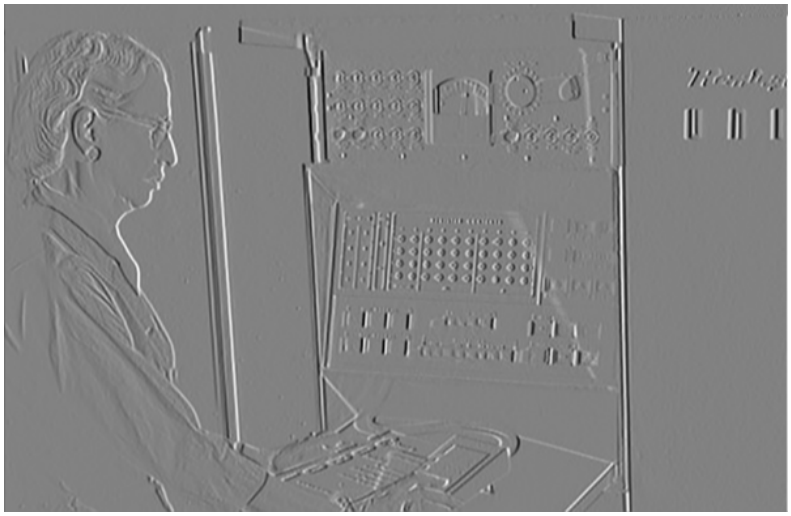
$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The above kernel approximates the derivative by taking the difference between the top and bottom sides of the kernel (changes in the y-direction).

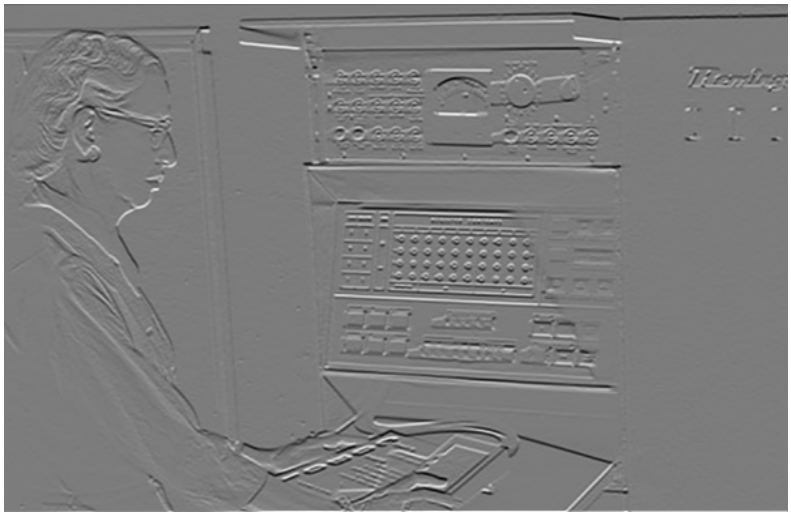
It is important to remember that these matrices consist of scalar values representing pixel weights. When processed via convolution:

- G_x detects changes in the vertical gradient, which can be seen as horizontal edges.
- G_y detects changes in the horizontal gradient, which can be seen as vertical edges.

These Sobel kernels provide a better approximation than standard edge-detecting kernels as they consider more neighboring pixels, leading to more accurate and robust edge detection.



(a) Gx



(b) Gy

(c) Gradient magnitude. $\sqrt{G_x^2 + G_y^2}$

Figure 13: Sobel Filter Outputs

(b)

(c) **(i)** The filter in question can be described as:

$$S(I, \alpha) = G_x \cos(\alpha) + G_y \sin(\alpha)$$

Here, we can conclude from previous statements that:

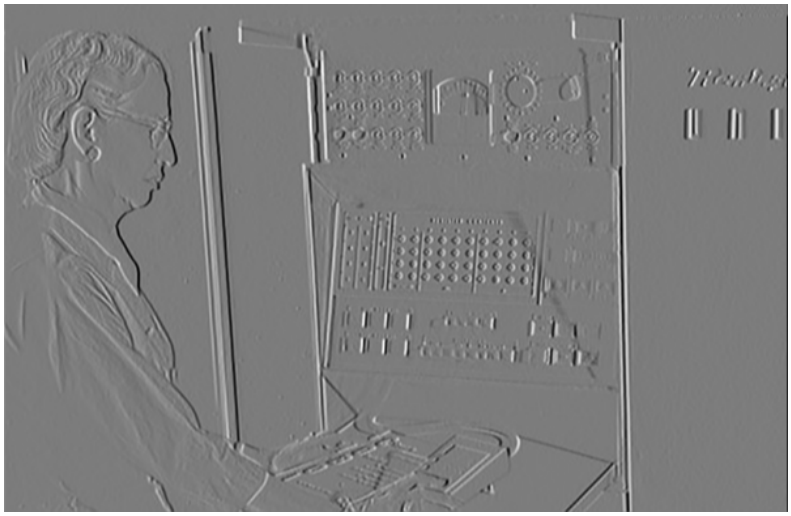
$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Passing G_x and G_y into the formula gives:

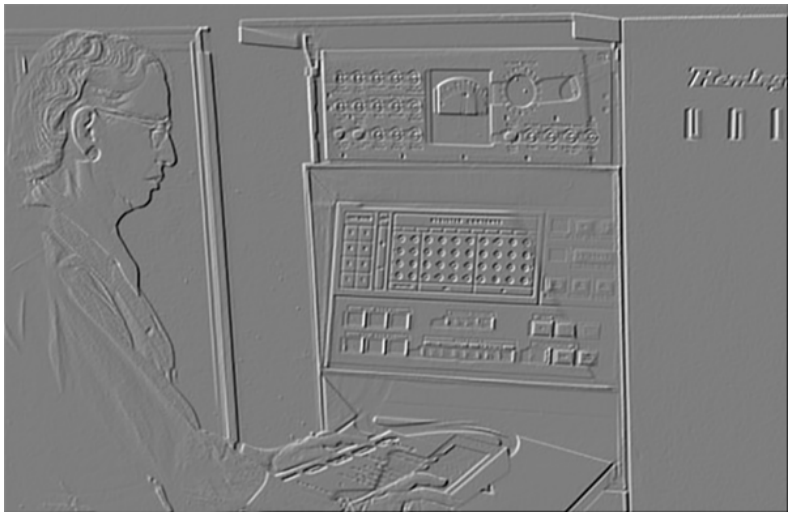
$$K(\alpha) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \cos(\alpha) + \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \sin(\alpha)$$

Combining the terms of $K(\alpha)$, we arrive at the final kernel:

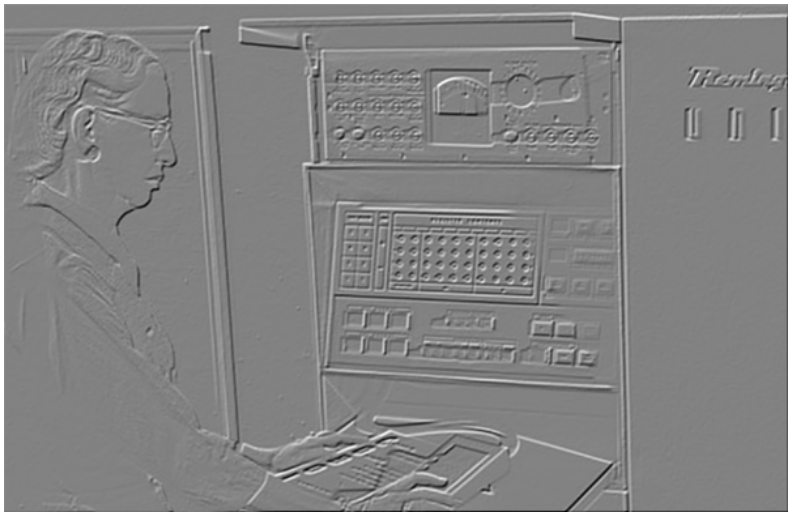
$$K(\alpha) = \begin{bmatrix} \cos(\alpha) + \sin(\alpha) & 2\sin(\alpha) & -\cos(\alpha) + \sin(\alpha) \\ 2\cos(\alpha) & 0 & -2\cos(\alpha) \\ \cos(\alpha) - \sin(\alpha) & -2\sin(\alpha) & -\cos(\alpha) - \sin(\alpha) \end{bmatrix}$$



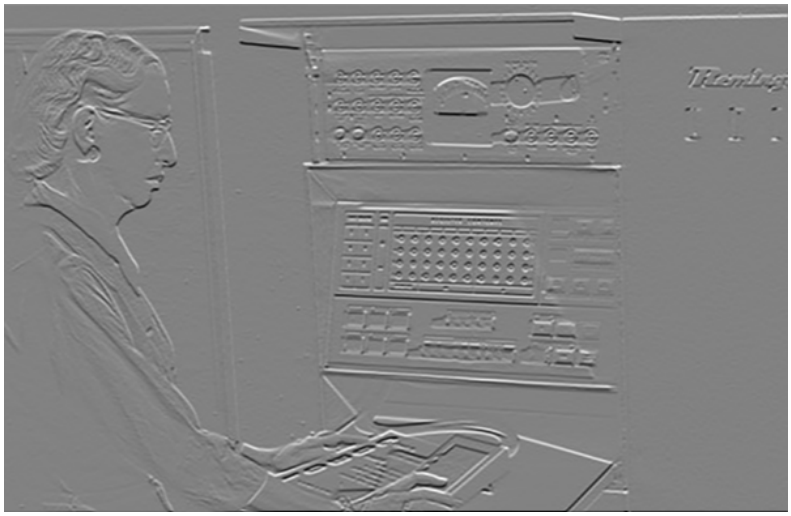
(a) Steerable $\alpha = 0$



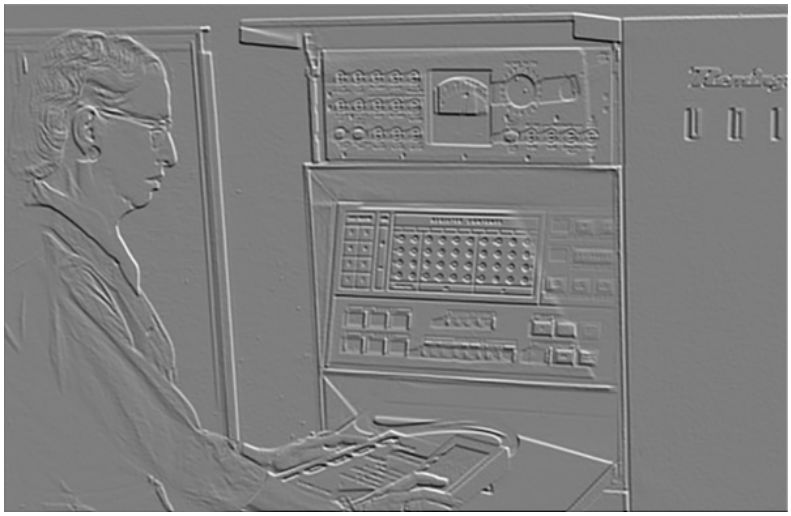
(b) Steerable $\alpha = \pi/6$



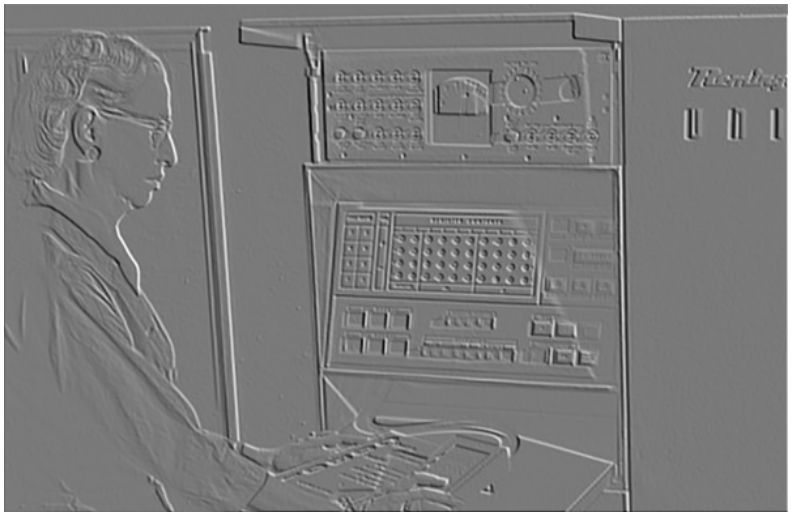
(c) Steerable $\alpha = \pi/3$



(a) Steerable $\alpha = \pi/2$



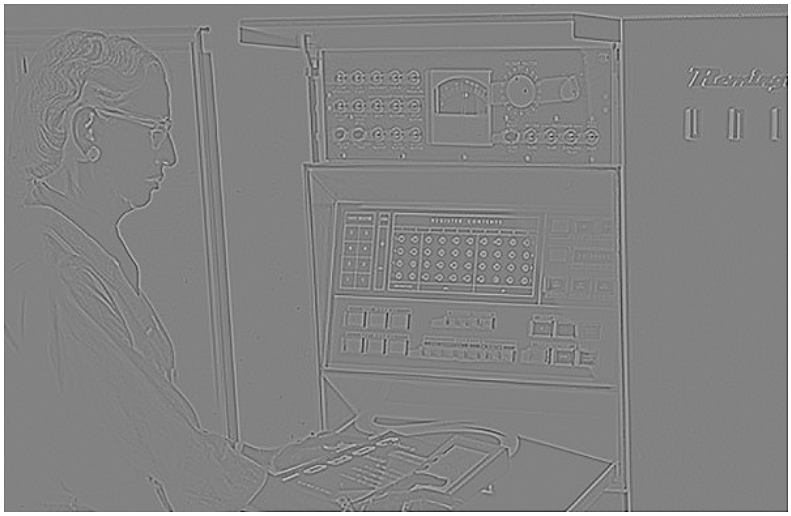
(b) Steerable $\alpha = 2\pi/3$



(c) Steerable $\alpha = 5\pi/6$

Figure 15: Steerable Filter Outputs at Various Angles

(d) **(iii)** This filter detects edges in various orientations of α within the given image. As α increases, the edges are pronounced in a clockwise rotation around the image, highlighting different edges under different angles.



(a) LoG filter 1

(b) LoG filter 2

Figure 16: Comparison of LoG Filters

5. (a) Visually, you can see that between the two LoG filters, there is a heavier weight on the lines in the second image, which emphasizes larger features. The difference here comes down to the size of the kernels and the values inside - the first filter has a 3x3 kernel with smaller values, and the second kernel is 9x9 containing much larger values. The larger values capture larger features due to the higher weights.

   These filters are both suitable for detecting edges. They may also be suitable for object detection, with the second filter being able to detect larger objects and the first filter detecting smaller details.

   (b) Instead of calculating the Laplacian of Gaussian (LoG) directly, we can approximate it using a simple Difference of Gaussians (DoG). This approximation works because the difference between two Gaussian functions with different standard deviations approximates the Laplacian (second derivative) of the Gaussian function. The second derivative from the DoG highlights areas in the image with sharp intensity changes.

References

[1] Wikipedia contributors. Isometric projection — Wikipedia, the free encyclopedia, 2023. [Online; accessed 10-November-2023].