Sam Feig | Vladimir Zhdanov

H3 CSCI 4831/5722

1. (a) This proof will be shown for the $3 * 3$ kernel size but is the same for any $k > 1$. Consider the following separable 2D filter kernel g of size $2k + 1$ with $k = 1$.

$$
\begin{aligned}
g_1 &= \begin{bmatrix} w_4 & w_5 & w_6 \end{bmatrix} \\
g_2 &= \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \\
g &= g_2 g_1 \\
&= \begin{bmatrix} w_1 w_4 & w_1 w_5 & w_1 w_6 \\ w_2 w_4 & w_2 w_5 & w_2 w_6 \\ w_3 w_4 & w_3 w_5 & w_3 w_6 \end{bmatrix}
\end{aligned}
$$

Then, consider the following value for our "image" f, where the pixel of interest for this proof is x.

$$
f = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & x & a_5 \\ a_6 & a_7 & a_8 \end{bmatrix}
$$

Then, the convolution of f with g on the pixel x is:

$$
\begin{aligned}
h_{2D}(m, n) &= \sum_{k,l} g(k, l) f(m + k, n + l) \\
&= a_1 w_1 w_4 + a_2 w_1 w_5 + a_3 w_1 w_6 + a_4 w_2 w_4 + \\
&\quad x w_2 w_5 + a_5 w_2 w_6 + a_6 w_3 w_4 + a_7 w_3 w_5 + a_8 w_3 w_6
\end{aligned}
$$

Now, we must confirm that applying each of the 1D convolutions separately gives the same result. The convolution of f with g_1 is:

$$
\begin{aligned}
h_1 &= \sum_{l} g_1(l) f(m, n + l) \\
&= \begin{bmatrix} w_4 a_1 + w_5 a_2 + w_6 a_3 \\ w_4 a_4 + w_5 x + w_6 a_5 \\ w_4 a_6 + w_5 a_7 + w_6 a_8 \end{bmatrix}
\end{aligned}
$$

Then, the convolution of h_1 (the result from the previous step) with g_2 gives the following.

$$\begin{aligned} h_2 &= \sum_k g_2(k)h_1(m+k) \\ &= a_1w_1w_4 + a_2w_1w_5 + a_3w_1w_6 + a_4w_2w_4 + \\ &xw_2w_5 + a_5w_2w_6 + a_6w_3w_4 + a_7w_3w_5 + a_8w_3w_6 \end{aligned}$$

Thus, the result is the same in both cases, so convolving an image with a discrete, separable 2D filter kernel is equivalent to convolving with two 1D filter kernels.

(b) For each pixel, a 2D filter kernel would perform $(2k+1)^2$ operations, while each 1D filter kernel would perform $2k+1$ operations. So, the number of operations saved per pixel is:

$$\begin{aligned} (2k+1)^2 - 2*(2k+1) &= (4k^2+4k+1) - (4k+2) \\ &= 4k^2 - 1 \end{aligned}$$

Based on this, the total number of operations saved for an $N * N$ image is $\boxed{N^2(4k^2-1)}$.

2.

3. Dimensionality reduction is a process for reducing the number of dimensions (or features) to be used in analysis of a dataset. This is especially important in cases where raw data samples have a lot of redundancy and patterns. The goal is to choose the most important features that still give an accurate representation of the overall data.

   With respect to image processing, dimensionality reduction is extremely useful in image compression. With image compression, we want to extract the singular values which contain the most important information of the image, so that it is not lost when the image is compressed. This is done by preserving the largest singular values, while nullifying the rest, resulting in a more compressed image.

   A disadvantage of dimensionality reduction is that it is hard and ineffective to perform on complex datasets and images which don't have redundant data or patterns. With these types of images, the largest singular values are not as easily separable, and image compression may result in the loss of important information about the image.

4. Since a rigid body transformation is defined as a rotation followed by a translation, in

R^3 we can define g as the following for some $t_x, t_y, t_z, \alpha, \beta, \gamma \in \mathbb{R}$.

$$
\begin{aligned}
R &= R_z R_y R_x \\
&= \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ - & \sin\alpha & \cos\alpha \end{bmatrix} \\
&= \begin{bmatrix} \cos\beta\cos\gamma & \cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma \\ -\cos\beta\sin\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & \sin\alpha\cos\gamma - \cos\alpha\sin\beta\sin\gamma \\ \sin\beta & -\sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix} \\
g &= \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta\cos\gamma & \cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma & 0 \\ -\cos\beta\sin\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & \sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & 0 \\ \sin\beta & -\sin\alpha\cos\beta & \cos\alpha\cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos\beta\cos\gamma & \cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma & t_x \\ -\cos\beta\sin\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & \sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & t_y \\ \sin\beta & -\sin\alpha\cos\beta & \cos\alpha\cos\beta & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

(a) For some $v = [v_1, v_2, v_3]$, we must show that $\|gv\| = \|v\|$:

$$
gv = \begin{bmatrix} \cos\beta\cos\gamma & \cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma & t_x \\ -\cos\beta\sin\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & \sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & t_y \\ \sin\beta & -\sin\alpha\cos\beta & \cos\alpha\cos\beta & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 1 \end{bmatrix} =
$$

$$
\begin{bmatrix} v_1\cos\beta\cos\gamma + v_2(\cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma) + v_3(\sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma) + t_x \\ -v_1\cos\beta\sin\gamma + v_2(\cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma) + v_3(\sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma) + t_y \\ v_1\sin\beta - v_2\sin\alpha\cos\beta + v_3\cos\alpha\cos\beta + t_z \end{bmatrix}
$$

Since this vector has been transformed by $[t_x, t_y, t_z]$, this quantity must be subtracted out of the vector before taking the norm, as both the start and end of the vector were transformed equally.

$$
\begin{aligned}
\|v\| &= (v_1^2 + v_2^2 + v_3^2)^{1/2} \\
\|gv\| &= \Big(\big(v_1\cos\beta\cos\gamma + v_2(\cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma) + v_3(\sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma)\big)^2 \\
&\quad + \big(v_1\cos\beta\sin\gamma + v_2(\cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma) + v_3(\sin\alpha\cos\gamma - \cos\alpha\sin\beta\sin\gamma)\big)^2 \\
&\quad + (v_1\sin\beta - v_2\sin\alpha\cos\beta + v_3\cos\alpha\cos\beta)^2 \Big)^{1/2}
\end{aligned}
$$

There are a lot of terms in this norm, and all of them which do not contain a v_k^2 term cancel out and sum to zero. I couldn't get the math to fit cleanly in this document, so

I moved ahead to the final simplification steps.

$$
\begin{aligned}
&= \big(v_1^2(\cos^2\beta\cos^2\gamma + \cos^2\beta\sin^2\gamma + \sin^2\beta) \\
&+ v_2^2(\cos^2\alpha\sin^2\gamma + 2\cos\alpha\sin\alpha\sin\beta\cos\gamma\sin\gamma + \sin^2\alpha\sin^2\beta\cos^2\gamma + \cos^2\alpha\cos^2\beta \\
&- 2\cos\alpha\sin\alpha\sin\beta\cos\gamma\sin\gamma + \sin^2\alpha\sin^2\beta\sin^2\gamma + \sin^2\alpha\cos^2\beta) \\
&+ v_3^2(\sin^2\alpha\sin^2\gamma - 2\sin\alpha\cos\alpha\sin\beta\cos\gamma\sin\gamma + \cos^2\alpha\sin^2\beta\cos^2\gamma \\
&+ \sin^2\alpha\cos^2\gamma + 2\sin\alpha\cos\alpha\sin\beta\sin\gamma\cos\gamma - \cos^2\alpha\sin^2\beta\sin^2\gamma + \cos^2\alpha\cos^2\beta)\big)^{1/2} \\
&= (v_1^2 + v_2^2 + v_3^2)^{1/2}
\end{aligned}
$$

So, since $\|gv\| = \|v\|$, any rigid-body transformation $g : R^3 \to R^3$ preserves the norm of a vector.

(b) For some $u = [u_1, u_2, u_3]$ and $v = [v_1, v_2, v_3]$, we must show that $(g * u) \times (g * v) = g * (u \times v)$. In (a), we showed that $\|gv\| = \|v\|$ for any vector $v \in R^3$. So, by properties of cross products and vector norms:

$$
\begin{aligned}
\|(g * u) \times (g * v)\| &= \|gu\|\|gv\|\sin\theta \\
&= \|u\|\|v\|\sin\theta \\
\|g * (u \times v)\| &= \|g\|\|(u \times v)\| \\
&= \|g\|\|u\|\|v\|\sin\theta \\
&= \|u\|\|v\|\sin\theta \\
\to \|(g * u) \times (g * v)\| &= \|g * (u \times v)\|
\end{aligned}
$$

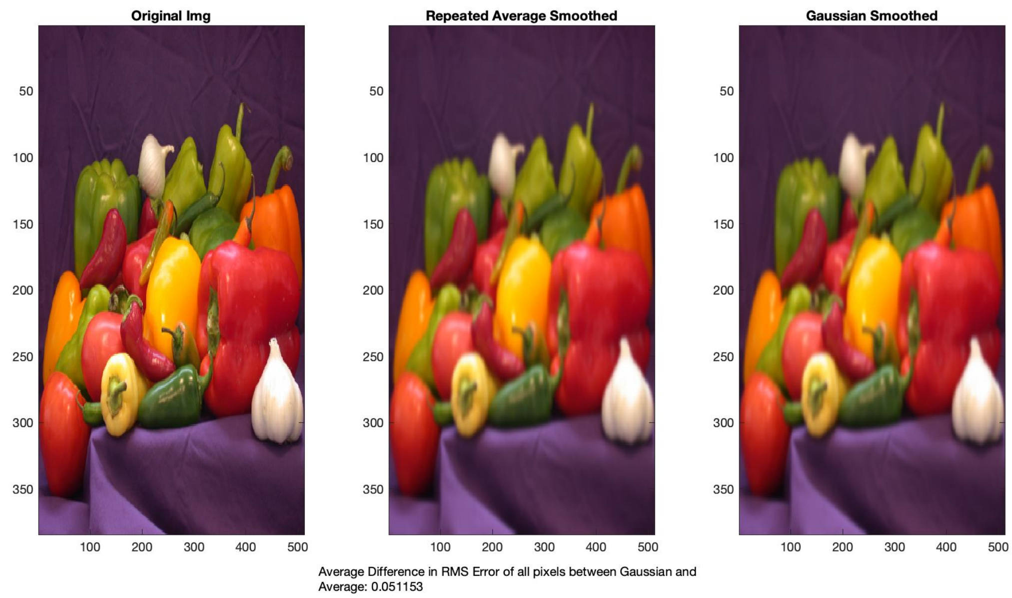
Since both vectors are transformed in the same way, the value for *theta* is consistent in both equations. Thus, this shows that the magnitude of both cross products are the same.

Now, we must confirm that the direction of both cross products is the same. For $(g * u) \times (g * v)$, the cross product will be some vector n_1 normal to both the vectors gu, gv. For $g * (u \times v)$, the cross product is some vector $g * n_2$, where n_2 is normal to both vectors u, v and is then transformed by the rigid body transformation g. Since the same rigid body transformation is applied to all the vectors in this case, we see that $n_1 = g * n_2$.

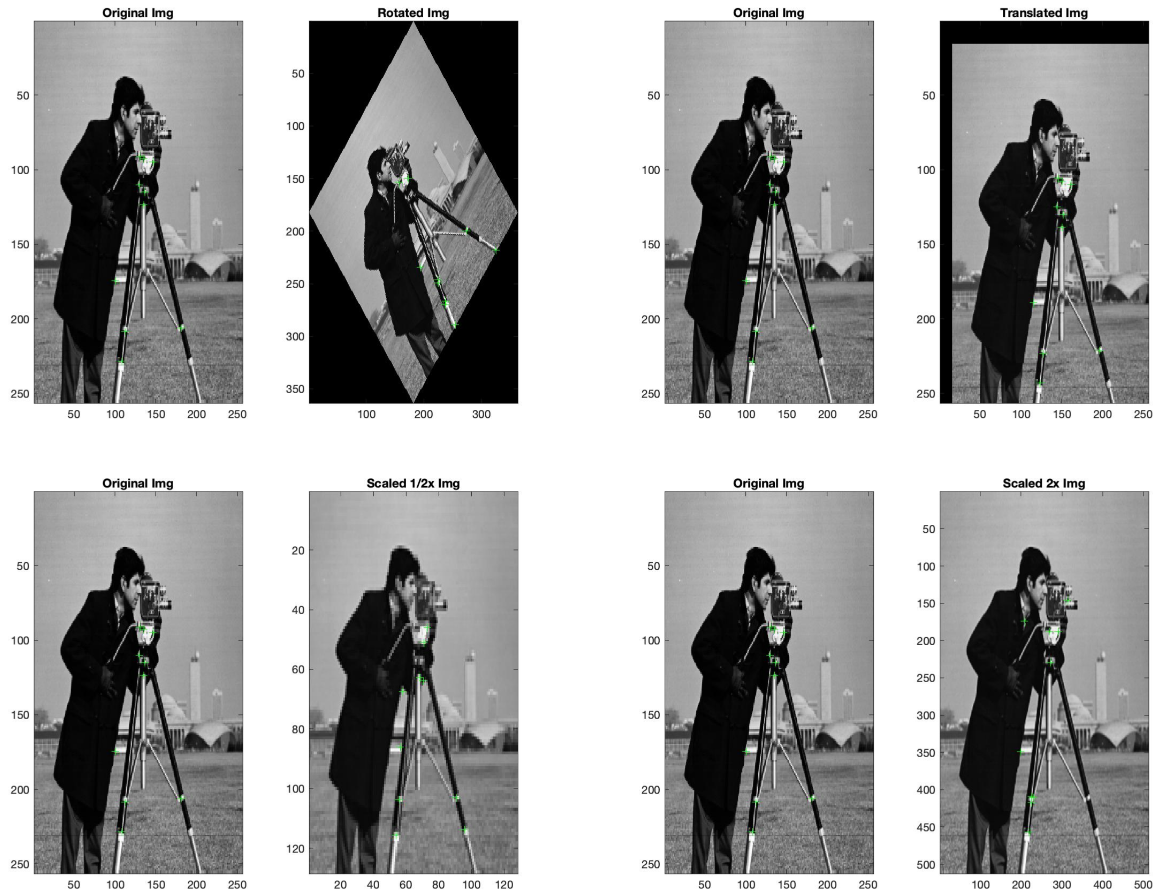
This shows that $(g * u) \times (g * v) = g * (u \times v)$, so any rigid-body transformation $g : R^3 \to R^3$ preserves the cross product of two vectors in R^3.

5. A way to verify that repeatedly applying an averaging filter will approximate Gaussian smoothing is to take an image and apply both side by side. First, apply a Gaussian smooth to the image, then take the original image and apply a averaging filter with a

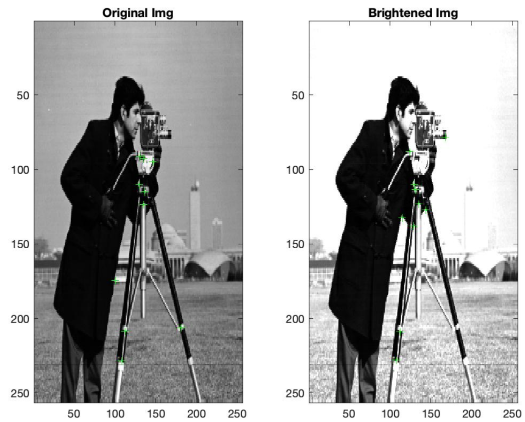
small kernel size, like 3x3. Then, repeat the application of the averaging filter until the output image looks approximately like the result of the Gaussian smoothing. Then, if we take the root mean square error of the pixels for both images compared to the original image and take the average of their difference, we can have an average RMS error for the difference between Gaussian smoothing and repeated averaging. Based on the plot below, we see that the error is very small, meaning the Gaussian smoothing with $\sigma = 2$ and using a 3x3 averaging filter 5 times result in nearly the same effect.

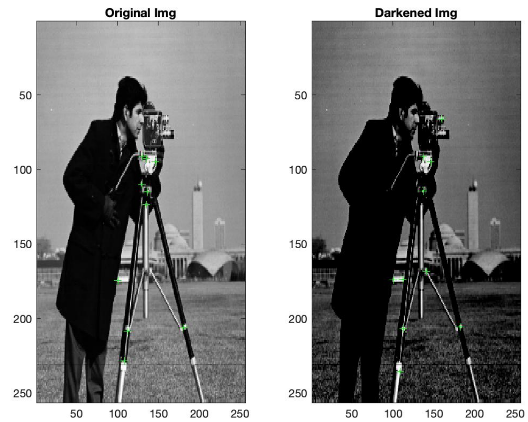


6. (a) The corner detection algorithm had no trouble with any of these transformations. Because all of these were only shifting pixels (either through rotation or translation) or were giving the image more resolution, it had no trouble detecting the same edges as the actual pixel values did not change. If the image got too small by scaling it down, I would imagine eventually it would cause issues for the Harris corner detection algorithm if it ran out of useful pixels to find corners with. Otherwise, these had no noticeable effect on the algorithm.

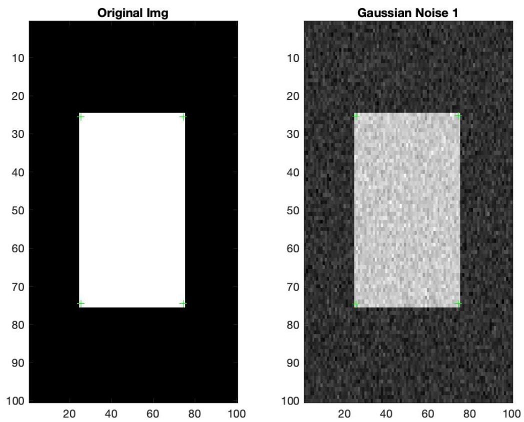


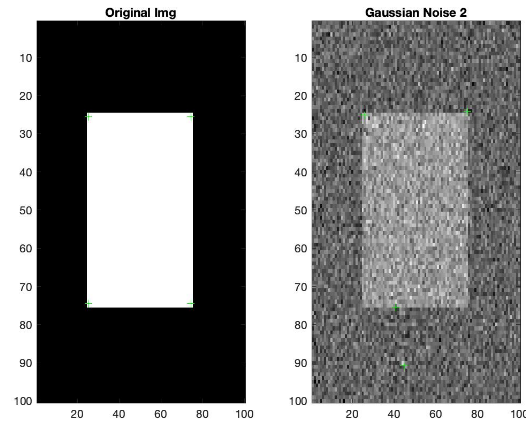
(b) Unlike in part A, these transformations of the images actually changed individual pixel value,s and therefore had a more noticeable effect on the results of the corner detection algorithm.

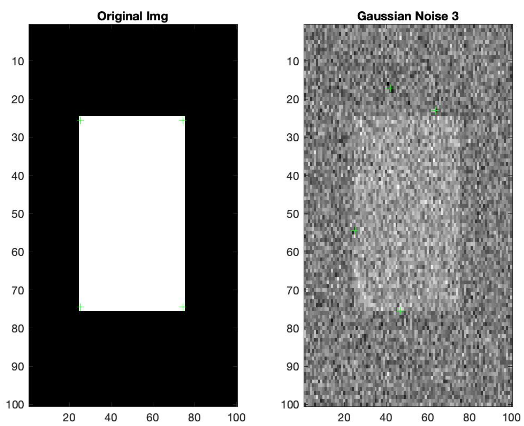


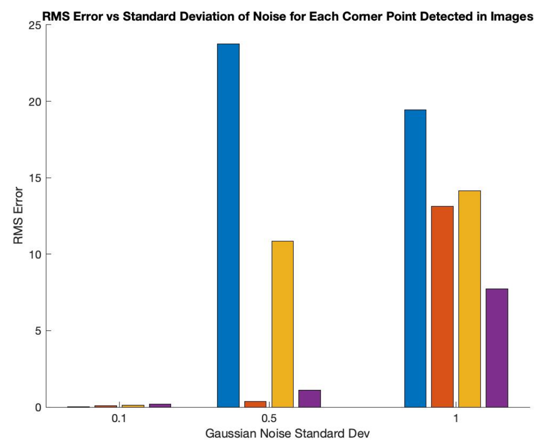


(c) Below is the output of adding increasing amounts of Gaussian noise to a white square on a black background. Gaussian noise 1 (least noise) has a standard deviation of 0.1. Gaussian noise 2 (middle) has a standard deviation of 0.5, and Gaussian noise 3 (most noise) has a standard deviation of 1. In the bar chart, each color bar represents one of the 4 corners we are detecting. Based on this plot, we see that the lowest level of noise barely affects the corner detection algorithm, as all 4 corners are found and have a very low RMS error. For the middle level of noise, the algorithm manages to find 2 corners with low error, but the other 2 are way off as seen by the spike in RMS error at a standard deviation of 0.5 up to above 10. The image with the most noise does not find any of the four corners, and all of the corner detection's have very high RMS errors.


Original Img
Gaussian Noise 1
10
20
30
40
50
60
70
80
90
100
20
40
60
80
100


Original Img
Gaussian Noise 2
10
20
30
40
50
60
70
80
90
100
20
40
60
80
100


Original Img
Gaussian Noise 3
10
20
30
40
50
60
70
80
90
100
20
40
60
80
100


RMS Error vs Standard Deviation of Noise for Each Corner Point Detected in Images
RMS Error
Gaussian Noise Standard Dev
0
5
10
15
20
25
0.1
0.5
1