A Note on the Separability of the Filters Used in ꟻLIP

Pontus Andersson

January 18, 2022

In this note, we explain how we separated the two-dimensional convolutions used in ꟻLIP into one-dimensional convolutions. This is done in two parts. First, we find the separated kernels and then we find the normalization factors that make the separated kernels output the same filtered image as the normalized two-dimensional kernel. We also mention some of the implementation details to make it easier for the user to understand the code. For details on the full algorithm, we refer to the ꟻLIP paper [Andersson et al., 2020]. In the following, bold face indicates two-dimensional arrays and functions. One-dimensional arrays and functions will be written without bold face.

If you have questions about or find problems in the text, please submit an issue about them on the ꟻLIP GitHub repository at `https://github.com/NVlabs/flip`.

1 Spatial Filters (Contrast Sensitivity Functions)

The so-called spatial filters used for ꟻLIP consist of primary space versions of the contrast sensitivity functions (CSFs). In particular, the CSFs are approximated using Gaussians. Single Gaussians are used for the achromatic and red-green channels, while the sum of two Gaussians are used for the blue-yellow channel. The two-dimensional Gaussian kernel functions are on the form

$$\mathbf{s}(x, y) = a\sqrt{\frac{\pi}{b}}e^{-\frac{\pi^2}{b}(x^2+y^2)} \tag{1}$$

and

$$\mathbf{d}(x, y) = a_1\sqrt{\frac{\pi}{b_1}}e^{-\frac{\pi^2}{b_1}(x^2+y^2)} + a_2\sqrt{\frac{\pi}{b_2}}e^{-\frac{\pi^2}{b_2}(x^2+y^2)}, \tag{2}$$

for the single Gaussians and sum of two Gaussians, respectively. Our kernels have radius r_c, which means that $x, y \in \{-r_c, -r_c + 1, \ldots, r_c\}$. The parameters for the three filters (achromatic, red-green, and blue-yellow), together with the filter radius, r_c, are given in the paper by Andersson et al. [Andersson et al., 2020]. Now, for a two-dimensional image, $\mathbf{I} \in \mathbb{R}^{W \times H}$, where W is the width and H

is the height of the image, the result of a convolution with a two-dimensional kernel, $\mathbf{g} \in \mathbb{R}^{w \times w}$, where $w = 2r + 1$ for some filter radius, r, can be written as

$$(\mathbf{I} * \mathbf{g})(x, y) = \sum_{i=-r}^{r} \sum_{j=-r}^{r} \mathbf{I}(x - i, y - j)\mathbf{g}(i, j), \tag{3}$$

where $(x, y) \in \{0, 1, \ldots, W - 1\} \times \{0, 1, \ldots, H - 1\}$. In the remainder of this note, we will often exclude the summation limits in our equations. We note that convolutions are commutative ($\mathbf{I} * \mathbf{g} = \mathbf{g} * \mathbf{I}$), associative ($\mathbf{I} * (\mathbf{g} * \mathbf{f}) = (\mathbf{I} * \mathbf{g}) * \mathbf{f}$), and distributive ($\mathbf{I} * (\mathbf{g} + \mathbf{f}) = \mathbf{I} * \mathbf{g} + \mathbf{I} * \mathbf{f}$). Convolution with a one-dimensional kernel is defined similarly, with the double sum replaced with a single one. The above properties hold for one-dimensional kernels as well.

We start by separating the convolution with a single Gaussian, $\mathbf{s}$, into two convolutions with one-dimensional kernels. Note that

$$\mathbf{s}(x, y) = a\sqrt{\frac{\pi}{b}} e^{-\frac{\pi^2}{b}(x^2+y^2)} = \sqrt{a\sqrt{\frac{\pi}{b}} e^{-\frac{\pi^2}{b}x^2}} \sqrt{a\sqrt{\frac{\pi}{b}} e^{-\frac{\pi^2}{b}y^2}} = s_1(x)s_2(y), \tag{4}$$

for any $x, y \in \mathbb{R}$. For the convolution (Equation 3), this gives

$$\begin{aligned}
(\mathbf{I} * \mathbf{s})(x, y) &= \sum_i \sum_j \mathbf{I}(x - i, y - j)\mathbf{s}(i, j) \\
&= \sum_i \sum_j \mathbf{I}(x - i, y - j)s_1(i)s_2(j) \\
&= \sum_i s_1(i) \sum_j \mathbf{I}(x - i, y - j)s_2(j) \\
&= ((\mathbf{I} * s_2) * s_1)(x, y) = ((\mathbf{I} * s_1) * s_2)(x, y),
\end{aligned} \tag{5}$$

where the last equality follows from the commutativity and associativity of the convolution operator. With that, we have now separated the convolution with a two-dimensional Gaussian to two convolutions with one-dimensional Gaussians. Note that the one-dimensional functions s_1 and s_2 are the same. This is exploited in our implementation.

Next, we consider the normalization of the kernels. The two-dimensional kernels, $\mathbf{s}$ and $\mathbf{d}$, are normalized by division by the sum of the kernel elements, i.e., $S(\mathbf{s}) = \sum_{i;j} \mathbf{s}(i, j)$ and $S(\mathbf{d}) = \sum_{i;j} \mathbf{d}(i, j)$ for $\mathbf{s}$ and $\mathbf{d}$, respectively. For the separated filters we create here, we need to figure out these normalization factors, preferably without computing the double sums above, but rather using just the one-dimensional kernels. For the single Gaussian, we note that $S(\mathbf{s}) = \sum_{i;j} \mathbf{s}(i, j) = \sum_{i;j} s_1(i)s_2(j) = \sum_i s_1(i) \sum_j s_2(j) = S(s_1)S(s_2)$, implying that if we divide the one-dimensional kernels by their respective sum, the result of the two convolutions with the normalized one-dimensional filters will be the same as the result of the convolution with the normalized two-dimensional filter. That

is, we have

$$\begin{aligned}\left(\mathbf{I} * \frac{\mathbf{s}}{S(\mathbf{s})}\right)(x,y) &= \left(\mathbf{I} * \frac{\mathbf{s}}{S(s_1)S(s_2)}\right)(x,y) \\ &= \left(\left(\mathbf{I} * \frac{s_1}{S(s_1)}\right) * \frac{s_2}{S(s_2)}\right)(x,y) \\ &= ((\mathbf{I} * \hat{s}) * \hat{s})(x,y),\end{aligned} \tag{6}$$

where $\hat{s} = \frac{s_1}{S(s_1)} = \frac{s_2}{S(s_2)}$. Now, Equation 6 can be rewritten as

$$\begin{aligned}\left(\mathbf{I} * \frac{\mathbf{s}}{S(\mathbf{s})}\right)(x,y) &= ((\mathbf{I} * \hat{s}) * \hat{s})(x,y) \\ &= \sum_j \hat{s}(j) \sum_i \mathbf{I}(x-i, y-j)\hat{s}(i) \\ &= \sum_j \hat{s}(j)\mathbf{I}_{\hat{s}}(x, y-j),\end{aligned} \tag{7}$$

where $\mathbf{I}_{\hat{s}}(x,y) = \sum_i \mathbf{I}(x-i,y)\hat{s}(i)$, i.e., we can compute the result by first convolving in the x-direction, storing the intermediate image, and then convolving in the y-direction to get the final result. This is the standard procedure for computing separated convolutions, and is what we will do throughout this note.

Next, we consider the kernel consisting of the sum of two Gaussians, $\mathbf{d}$. For this kernel, we note that the distributive property of convolutions yields

$$\mathbf{I} * \mathbf{d} = \mathbf{I} * \mathbf{d}_1 + \mathbf{I} * \mathbf{d}_2, \tag{8}$$

where

$$\mathbf{d}_1 = a_1\sqrt{\frac{\pi}{b_1}}e^{-\frac{\pi^2}{b_1}(x^2+y^2)} \text{ and } \mathbf{d}_2 = a_2\sqrt{\frac{\pi}{b_2}}e^{-\frac{\pi^2}{b_2}(x^2+y^2)}. \tag{9}$$

Above we showed how the convolution with a single two-dimensional Gaussian could be separated into two convolutions with one-dimensional Gaussians. We apply the same technique for the two terms in Equation 8. That is, we separate $\mathbf{d}_1$ and $\mathbf{d}_2$ into two one-dimensional Gaussians, d_{11} and d_{12} for $\mathbf{d}_1$ and d_{21} and d_{22} for $\mathbf{d}_2$, i.e.,

$$\mathbf{d}_1(x,y) = a_1\sqrt{\frac{\pi}{b_1}}e^{-\frac{\pi^2}{b_1}(x^2+y^2)} = \sqrt{a_1\sqrt{\frac{\pi}{b_1}}e^{-\frac{\pi^2}{b_1}x^2}}\sqrt{a_1\sqrt{\frac{\pi}{b_1}}e^{-\frac{\pi^2}{b_1}y^2}} = d_{11}(x)d_{12}(y) \tag{10}$$

and

$$\mathbf{d}_2(x,y) = a_2\sqrt{\frac{\pi}{b_2}}e^{-\frac{\pi^2}{b_2}(x^2+y^2)} = \sqrt{a_2\sqrt{\frac{\pi}{b_2}}e^{-\frac{\pi^2}{b_2}x^2}}\sqrt{a_2\sqrt{\frac{\pi}{b_2}}e^{-\frac{\pi^2}{b_2}y^2}} = d_{21}(x)d_{22}(y). \tag{11}$$

As for the single Gaussian in Equation 4, we note that the two one-dimensional Gaussians in Equation 10 are the same, and so are the two one-dimensional

Gaussians in Equation 11. We now compute the result of each term in Equation 8 using Equation 5, but with $\mathbf{s}$ replaced with $\mathbf{d}_1$ and $\mathbf{d}_2$, and sum the results to obtain the filtered image. Formally, we have

$$\begin{aligned}\mathbf{I} * \mathbf{d}(x,y) &= (\mathbf{I} * \mathbf{d}_1)(x,y) + (\mathbf{I} * \mathbf{d}_2)(x,y) \\ &= \sum_j d_{12}(j) \sum_i \mathbf{I}(x-i, y-j) d_{11}(i) \\ &+ \sum_j d_{22}(j) \sum_i \mathbf{I}(x-i, y-j) d_{21}(i).\end{aligned} \tag{12}$$

Finally, we need to figure out the normalization factors for the one-dimensional versions of the two Gaussians $\mathbf{d}_1$ and $\mathbf{d}_2$. The two-dimensional filter, $\mathbf{d}$, is normalized similar to $\mathbf{s}$, i.e., by division by

$$\begin{aligned}S(\mathbf{d}) &= \sum_{i;j} \mathbf{d}(i,j) = \sum_{i;j} \mathbf{d}_1(i,j) + \mathbf{d}_2(i,j) \\ &= \sum_i d_{11}(i) \sum_j d_{12}(j) + \sum_i d_{21}(i) \sum_j d_{22}(j) \\ &= S(d_{11})S(d_{12}) + S(d_{21})S(d_{22}) = M^2,\end{aligned} \tag{13}$$

which, as is implied by the last row in the equation, we can compute directly using the one-dimensional filters. Thus, we get the correct, normalized result by dividing the one-dimensional kernels by M, i.e.,

$$\begin{aligned}\left(\mathbf{I} * \frac{\mathbf{d}}{S(\mathbf{d})}\right)(x,y) &= \sum_j \frac{d_{12}(j)}{M} \sum_i \mathbf{I}(x-i, y-j) \frac{d_{11}(i)}{M} \\ &+ \sum_j \frac{d_{22}(j)}{M} \sum_i \mathbf{I}(x-i, y-j) \frac{d_{21}(i)}{M} \\ &= \sum_j \hat{d}_1(j) \sum_i \mathbf{I}(x-i, y-j) \hat{d}_1(i) \\ &+ \sum_j \hat{d}_2(j) \sum_i \mathbf{I}(x-i, y-j) \hat{d}_2(i),\end{aligned} \tag{14}$$

where $\hat{d}_1 = \frac{d_{11}}{M} = \frac{d_{12}}{M}$ and $\hat{d}_2 = \frac{d_{21}}{M} = \frac{d_{22}}{M}$. As in the case with a single Gaussian, we use intermediate images in our implementation, yielding the result in Equation 14 as

$$\left(\mathbf{I} * \frac{\mathbf{d}}{S(\mathbf{d})}\right)(x,y) = \sum_j \hat{d}_1(j) \mathbf{I}_{\hat{d}_1}(x, y-j) + \sum_j \hat{d}_2(j) \mathbf{I}_{\hat{d}_2}(x, y-j), \tag{15}$$

where $\mathbf{I}_{\hat{d}_1}(x,y) = \sum_i \mathbf{I}(x-i, y) \hat{d}_1(i)$ and $\mathbf{I}_{\hat{d}_2}(x,y) = \sum_i \mathbf{I}(x-i, y) \hat{d}_2(i)$. As the sums run over the same parts of the image $\mathbf{I}$ in both terms, we can (and do) compute both intermediate images, $\hat{\mathbf{I}}_{d;1}$ and $\hat{\mathbf{I}}_{d;2}$, in one loop and then the final result in a second. In those same loops, we also compute the convolutions in Equation 7.

2 Feature Detection Filters

In this section, we will derive the one-dimensional kernels for the feature detection filters. The detection filters consist of edge detection and point detection kernels, which, respectively, include the first and second derivative of the same Gaussian. The kernel functions we will consider are the following:

$$\mathbf{g}(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{16}$$

$$\frac{\partial}{\partial x}\mathbf{g}(x, y) = -\frac{x}{\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{17}$$

$$\frac{\partial^2}{\partial x^2}\mathbf{g}(x, y) = \frac{1}{\sigma^2}\left(\frac{x^2}{\sigma^2} - 1\right)e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{18}$$

and the corresponding y-derivatives, where σ can be found in Andersson et al.'s paper [Andersson et al., 2020].

We note that the kernel functions above can be separated into one function in x and one in y as

$$\begin{aligned}\frac{\partial}{\partial x}\mathbf{g}(x, y) &= -\frac{x}{\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \left(-\frac{x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}}\right)e^{-\frac{y^2}{2\sigma^2}} \\ &= \left(\frac{d}{dx}g(x)\right)g(y),\end{aligned} \tag{19}$$

where $g(x) = e^{-\frac{x^2}{2\sigma^2}}$, and

$$\begin{aligned}\frac{\partial^2}{\partial x^2}\mathbf{g}(x, y) &= \frac{1}{\sigma^2}\left(\frac{x^2}{\sigma^2} - 1\right)e^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \left(\frac{1}{\sigma^2}\left(\frac{x^2}{\sigma^2} - 1\right)e^{-\frac{x^2}{2\sigma^2}}\right)e^{-\frac{y^2}{2\sigma^2}} \\ &= \left(\frac{d^2}{dx^2}g(x)\right)g(y).\end{aligned} \tag{20}$$

In our implementation, we use that $\frac{d}{dx}g = -\frac{x}{\sigma^2}g$ and $\frac{d^2}{dx^2}g = \frac{1}{\sigma^2}\left(\frac{x}{\sigma^2} - 1\right)g$.

The feature detection in ∃LIP produces four results, namely

$$\mathbf{I} * \frac{\partial}{\partial x}\mathbf{g}, \quad \mathbf{I} * \frac{\partial^2}{\partial x^2}\mathbf{g}, \quad \mathbf{I} * \frac{\partial}{\partial y}\mathbf{g}, \quad \mathbf{I} * \frac{\partial^2}{\partial y^2}\mathbf{g}. \tag{21}$$

We focus on the first result, $\mathbf{I} * \frac{\partial}{\partial x}\mathbf{g}$, and note that the derivation of the other

three results are analogous. We have

$$\begin{aligned}\left(\mathbf{I} * \frac{\partial}{\partial x}\mathbf{g}\right)(x,y) &= \sum_j \sum_i \mathbf{I}(x-i,y-j)\frac{\partial}{\partial x}\mathbf{g}(i,j) \\ &= \sum_j \sum_i \mathbf{I}(x-i,y-j)\left(\frac{\mathrm{d}}{\mathrm{d}x}g(i)\right)g(j) \\ &= \sum_j g(j)\sum_i \mathbf{I}(x-i,y-j)\frac{\mathrm{d}}{\mathrm{d}x}g(i) \\ &= \sum_j g(j)\mathbf{I}_{g'_x}(x,y-j),\end{aligned} \tag{22}$$

where $\mathbf{I}_{g'_x}(x,y) = \sum_i \mathbf{I}(x-i,y)\frac{\mathrm{d}}{\mathrm{d}x}g(i)$, which separates the two-dimensional convolution into two one-dimensional ones. Here, the summations run from $-r_f$ to r_f, where r_f can be found in the ∃LIP paper [Andersson et al., 2020].

Normalization is slightly more involved for the feature detection compared to the spatial filtering. The complication lies in that we want to normalize the kernel such that its positive weights sum to 1 and its negative weights sum to -1. We note that the values of the Gaussian, g, are all positive. Thus, we have that an element $\frac{\partial}{\partial x}\hat{\mathbf{g}}(x,y)$ in the normalized, two-dimensional filter, $\frac{\partial}{\partial x}\hat{\mathbf{g}}$, can be written

$$\frac{\partial}{\partial x}\hat{\mathbf{g}}(x,y) = \frac{\frac{\partial}{\partial x}\mathbf{g}(x,y)}{S^{\pm}(\frac{\partial}{\partial x}\mathbf{g},x,y)} = \frac{\frac{\mathrm{d}}{\mathrm{d}x}g(x)}{S^{\pm}(\frac{\mathrm{d}}{\mathrm{d}x}g,x)}\frac{g(y)}{S(g)}, \tag{23}$$

where

$$S^{\pm}(f,x,y) = \begin{cases} S(f^+), & \text{if } f(x,y) \geq 0, \\ S(f^-), & \text{otherwise,}\end{cases} \tag{24}$$

and similar for $S^{\pm}(f,x)$, with f^+ (f^-) denoting the array containing the positive (negative) elements of the array f. Now, just as with the spatial filters in the previous section, we have found how to normalize the separated kernels in order to achieve the same result as with the normalized two-dimensional kernel. The final convolution, carried out in our implementation, is thus

$$\begin{aligned}\left(\mathbf{I} * \frac{\partial}{\partial x}\hat{\mathbf{g}}\right)(x,y) &= \sum_j \frac{g(j)}{S(g)}\sum_i \mathbf{I}(x-i,y-j)\frac{\frac{\mathrm{d}}{\mathrm{d}x}g(i)}{S^{\pm}\left(\frac{\mathrm{d}}{\mathrm{d}x}g(i),x\right)} \\ &= \sum_j \hat{g}(j)\sum_i \mathbf{I}(x-i,y-j)\frac{\mathrm{d}}{\mathrm{d}x}\hat{g}(i) \\ &= \sum_j \hat{g}(j)\mathbf{I}_{\hat{g}'_x}(x,y-j),\end{aligned} \tag{25}$$

where $\hat{g} = \frac{g}{S(g)}$, $\frac{\mathrm{d}}{\mathrm{d}x}\hat{g} = \frac{\frac{\mathrm{d}}{\mathrm{d}x}g}{S^{\pm}(\frac{\mathrm{d}}{\mathrm{d}x}g;x;y)}$, and $\mathbf{I}_{\hat{g}'_x}(x,y) = \sum_i \mathbf{I}(x-i,y)\frac{\mathrm{d}}{\mathrm{d}x}\hat{g}(i)$. The remaining results in Equation 21 are found in a similar fashion, with first derivatives replaced by second derivatives and x's replaced by y's.

For our implementation, we note that, due to the associative and commutative properties of the convolution, we have

$$\begin{aligned}\left(\mathbf{I} * \frac{\partial}{\partial y}\mathbf{g}\right)(x,y) &= \left(\left(\mathbf{I} * \frac{\partial}{\partial y}g\right) * g\right)(x,y) \\ &= \left((\mathbf{I} * g) * \frac{\partial}{\partial y}g\right)(x,y) \\ &= \sum_j \frac{\mathrm{d}}{\mathrm{d}y}g(j)\sum_i \mathbf{I}(x-i,y-j)g(i) \\ &= \sum_j \left(\frac{\mathrm{d}}{\mathrm{d}y}g(j)\right)\mathbf{I}_g(x,y-j),\end{aligned} \tag{26}$$

where $\mathbf{I}_g = \sum_i \mathbf{I}(x-i,y)g(i)$. For the convolution with the second derivative in y, we similarly have

$$\left(\mathbf{I} * \frac{\partial^2}{\partial y^2}\mathbf{g}\right)(x,y) = \sum_j \left(\frac{\mathrm{d}^2}{\mathrm{d}y^2}g(j)\right)\mathbf{I}_g(x,y-j). \tag{27}$$

Noting that the intermediate images are the same in the last two equations, this allows us to store only three, rather than four, intermediate images for the results in Equation 21. In the first convolution loop, we compute $\mathbf{I}_g$, $\mathbf{I}_{g'_x}$, and $\mathbf{I}_{g''_x}$ (where $\mathbf{I}_{g''_x}(x,y) = \sum_i \mathbf{I}(x-i,y)\frac{\mathrm{d}^2}{\mathrm{d}x^2}g(i)$). In the second loop, we then plug in the first intermediate image into the computations in Equations 26 and 27 to find $\mathbf{I} * \frac{\partial}{\partial y}\mathbf{g}$ and $\mathbf{I} * \frac{\partial^2}{\partial y^2}\mathbf{g}$, respectively. The second intermediate image is inserted into the computations in Equation 25 to find $\mathbf{I} * \frac{\partial}{\partial x}\mathbf{g}$, and the third intermediate image is used similarly to compute $\mathbf{I} * \frac{\partial^2}{\partial x^2}\mathbf{g}$. Finally, we note that, in our implementation, we exclude the $1/\sigma^2$ factor in Equations 17 and 18 when the kernels are set up, as that factor cancels out during normalization and can therefore be safely discarded.

3 Acknowledgements

Thanks to Killian Herveau for sharing his derivations and, along with Vinh Truong, inspiring us to better optimize the ∃LIP code and to Tomas Akenine-Mœller for proofreading this document.

References

[Andersson et al., 2020] Andersson, P., Nilsson, J., Akenine-Mœller, T., Oskarsson, M., Åström, K., and Fairchild, M. D. (2020). ∃LIP: A difference evaluator for alternating images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2):15:1{15:23.