CV202, HW #2

Oren Freifeld and Meitar Ronen
The Department of Computer Science, Ben-Gurion University

Release Date: 29/3/2020
Submission Deadline: 7/5/2020, 20:00

Abstract

Please make sure you have carefully read the general instructions in the course website at https://www.cs.bgu.ac.il/~cv202/HW_Instructions. These instructions address, among other things, what to do or not do with figures, Jupyter Notebooks, *etc.*, as well as what compression files are legit or not.

This assignment focuses on several topics: linear and affine maps; linear least squares; transformation groups; filtering. The data file for this assignment is https://www.cs.bgu.ac.il/~cv202/wiki.files/hw2_data.tar.xz.

Version Log

- 1.01, 15/4/2020. Problem 21: corrected an expression in the hint (changed $\boldsymbol{\theta}^T\boldsymbol{V}$ to $\boldsymbol{V}\boldsymbol{\theta}$ to fix the dimension mismatch). Also added a remark there to draw your attention to a common mistake students often do in that problem.
- 1.00, 29/3/2020. Initial release.

Contents

1 A Few Words Before You Start

1. Throughout all computer exercises below, do not implement the convolutions yourself; rather, use standard implementations such as those in scipy.ndimage or OpenCV (*e.g.*, cv2.filter2D). For oft-used filters, there are usually already functions that will spare you the need to pass the filter as an argument (*e.g.*, cv2.GaussianBlur).

2. Get the data:
   https://www.cs.bgu.ac.il/~cv202/wiki.files/hw2_data.tar.xz

3. Some of the data is saved as (Matlab) mat files. To read these in python, use scipy.io.loadmat, which returns a dictionary. For example:

```
Code:
import scipy.io as sio
mdict = sio.loadmat("are_these_separable_filters.mat")
print ([k for k in mdict.keys() if not k.startswith("__")])
Output:
['K3', 'K2', 'K1']
```

   Then you can get the data more directly (in effect, as numpy arrays) via:

```
K1=mdict["K1"]
K2=mdict["K2"]
K3=mdict["K3"]
```

2 Linear and Affine Maps

Problem 1 *A linear map from $\mathbb{R}^n$ to $\mathbb{R}^n$, $f : \boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$, is invertible if and only if $\boldsymbol{A}$ satisfies a condition. What is this condition?* ♢

Problem 2 *Let $f : \mathbb{R} \to \mathbb{R}$ be defined by $f : x \mapsto ax + b$ for some $a, b \in \mathbb{R}$. In other words, f describes a straight line. Show that if $b \neq 0$ then f is a nonlinear function. Remark: people often refer to such functions as linear, and occasionally we may also do it ourselves, but, strictly speaking, this is incorrect.*♢

Problem 3 *Under what condition(s) an affine map, $f : \mathbb{R}^n \to \mathbb{R}^m$, $f : \boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ (where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$), is invertible?* ♢

Problem 4 (Composition of affine maps) *Let $h = g \circ f$ where $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}^k$ are two affine maps:*

$$\begin{aligned} f &: \boldsymbol{x} \mapsto \boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{b}_1 \qquad \boldsymbol{A}_1 \in \mathbb{R}^{m\times n} \qquad \boldsymbol{b}_1 \in \mathbb{R}^m\,; \\ g &: \boldsymbol{x} \mapsto \boldsymbol{A}_2\boldsymbol{x} + \boldsymbol{b}_2 \qquad \boldsymbol{A}_2 \in \mathbb{R}^{k\times m} \qquad \boldsymbol{b}_2 \in \mathbb{R}^k\,. \end{aligned} \tag{1}$$

Show that $h : \mathbb{R}^n \to \mathbb{R}^k$ is affine. ♢

Problem 5 (Composition of invertible affine maps) *Let $h = g \circ f$ where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^n$ are two invertible affine maps. Show that h is also an invertible affine map.* ◊

3 Matrix Lie groups: Definition and Basic Properties

Definition 1 (Matrix Lie groups) *Let n be a fixed positive integer. A* matrix Lie group *is a set G of $n \times n$ matrices together with the binary operation of matrix product on G (that is, the domain is $G \times G$) such that:*

(G1) $\boldsymbol{I}_{n\times n} \in G$;

(G2) $\boldsymbol{A}, \boldsymbol{B} \in G \Rightarrow \boldsymbol{AB} \in G$;

(G3) $\boldsymbol{A} \in G \Rightarrow \boldsymbol{A}^{-1}$ *exists and* $\boldsymbol{A}^{-1} \in G$. ◊

Remark 1 *The word "Lie" was the last name of the famous mathematician Sophus Lie. Note it is pronounced "LEE", and not as the English verb "lie".* ◊

- Matrix Lie groups are also called *matrix groups*, the terms being identical. It is possible to use a similar definition for matrix Lie groups whose elements take complex values; however, we restrict the discussion to real-valued matrix Lie groups.

- Let G be a matrix group. Basic linear algebra shows that:

  (G4) $\boldsymbol{A} \in G \Rightarrow \boldsymbol{AI}_{n\times n} = \boldsymbol{I}_{n\times n}\boldsymbol{A} = \boldsymbol{A}$

  (G5) $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in G \Rightarrow (\boldsymbol{AB})\boldsymbol{C} = \boldsymbol{A}(\boldsymbol{BC}) = \boldsymbol{ABC}$

  (G6) $\boldsymbol{A} \in G \Rightarrow \boldsymbol{AA}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_{n\times n}$.

- What (G1) through (G6) imply is that the set G, together with the binary operation of matrix product, is a group and that $\boldsymbol{I}_{n\times n}$ is the identity element of the group. So unsurprisingly, every matrix group is a group. When n is understood from the context, we will sometimes denote the identity matrix by $\boldsymbol{I}$ instead of $\boldsymbol{I}_{n\times n}$.

- If G is a set of $n \times n$ matrices, which may or may not form a matrix group once taken together with matrix product, it may be the case that there is some *other* binary operation, denoted by, say, $\diamond : G \times G \to G$, such that the pair $(G, \diamond)$ forms a group. This group, however, is not a matrix group, although it is a "group of matrices"; *e.g.*, if G is $n \times n$ matrices and $\diamond$ is matrix addition, than the group $(G, \diamond)$ is not a matrix group.

- Henceforth, by a slight abuse of notation, we will write expressions such as "G is a matrix Lie group", with the convention that we will mean that the *set* G, together with matrix product, is a matrix Lie group. Similarly, we will write "the matrix Lie group G is given by the following set", meaning that the set, together with matrix product, defines the matrix Lie group of interest.

Definition 2 (The difference between two elements in a matrix Lie group) *Let G be a matrix Lie group, and let $\boldsymbol{A}$ and $\boldsymbol{B}$ be in G. The* group difference *of $\boldsymbol{A}$ and $\boldsymbol{B}$ is given by $A^{-1}B$; it is not symmetric.* ◊

- Note that (G2) and (G3) imply the following:

  (G7) $\boldsymbol{A}, \boldsymbol{B} \in G \Rightarrow \boldsymbol{A}^{-1}\boldsymbol{B} \in G$.

  Properties (G2), (G3), and (G7), are referred to as the group (algebraic) *closure* under its operations of *composition*, *inversion*, and *difference*, respectively. Suppose our data can be represented as elements of a matrix Lie group. These closure properties imply that the representation is consistent.

Definition 3 (Abelian matrix Lie group) *If $\boldsymbol{AB} = \boldsymbol{BA}$ for any $\boldsymbol{A}$ and $\boldsymbol{B}$ in a matrix Lie group G, then G is called* Abelian. ◊

The class of matrix Lie groups is contained in the more general class of (finite-dimensional) *Lie groups*: every matrix Lie group is a Lie group while the converse is false. Matrix Lie groups are simpler to work with (and define) than the more general case of Lie groups. We will restrict discussion to matrix Lie groups.

Definition 4 (The general linear group of order n) *The* general linear group of order n, *denoted by* GL(n), *is given by*

$$\{\boldsymbol{Q} \,|\, \boldsymbol{Q} \in \mathbb{R}^{n\times n}, \det \boldsymbol{Q} \neq 0\}$$

Problem 6 *Show that* GL(n) *is a matrix group.* ◊

Problem 7 *Show that* GL(n) *is not Abelian.* ◊

Remark 2 GL(n) *is not a connected space. To see that, pick a matrix $\boldsymbol{A}$ in* GL(n) *with a positive determinant and a matrix $\boldsymbol{B}$ in* GL(n) *with a negative determinant. Then, set $p = \boldsymbol{A}$ and $q = \boldsymbol{B}$. As* $\det : \mathbb{R}^{n\times n} \to \mathbb{R}$ *is a continuous curve between p and q must pass through a matrix of zero determinant;* i.e., *the $\mathbb{R}^{n\times n}$-valued curve must make an excursion outside of* GL(n). *In fact, it can be shown that* GL(n) *has exactly two connected components: matrices of positive determinant; matrices of negative determinant.* ◊

Definition 5 (The identity component) *The* identity component *of a matrix Lie group is one that contains $\boldsymbol{I}_{n\times n}$. The identity component of a matrix Lie group is always a matrix Lie group by itself.* ◊

Definition 6 ($\mathbf{GL_+(n)}$) *The* identity component of GL(n), *denoted by* $\mathrm{GL}_+(\mathrm{n})$, *is given by*

$$\{\boldsymbol{Q} \,|\, \boldsymbol{Q} \in \mathbb{R}^{n\times n}, \det \boldsymbol{Q} > 0\}.$$

Problem 8 *Show that* $\mathrm{GL}_+(\mathrm{n})$ *is a matrix group.* ◊

Problem 9 *Show that*

$$\{\boldsymbol{Q} \,|\, \boldsymbol{Q} \in \mathbb{R}^{n\times n}, \det \boldsymbol{Q} < 0\}$$

is not a matrix group. ◊

Not every set of $n \times n$ invertible matrices is a matrix group. *E.g.*, consider SPD matrices. Even if $\boldsymbol{A}$ and $\boldsymbol{B}$ are SPD, $\boldsymbol{C} = \boldsymbol{AB}$ is usually not SPD. The problem is not with the positive definiteness, but with the symmetry.

Problem 10 *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two $n \times n$ symmetric matrices, namely, $\boldsymbol{A} = \boldsymbol{A}^T$ and $\boldsymbol{B} = \boldsymbol{B}^T$. Show that, usually, $\boldsymbol{AB} \neq (\boldsymbol{AB})^T$.* ♢

Definition 7 (US(n)) *The* uniform scale group*, denoted by* US(n)*, is given by*

$$\{\boldsymbol{Q} \,|\, \boldsymbol{Q} = S\boldsymbol{I}_{n\times n},\ S \in \mathbb{R}_{>0}\}.$$

Problem 11 *show that the uniform scale group is a matrix group.* ♢

Problem 12 *show that the uniform scale group is Abelian.* ♢

Problem 13 *What is the zero element of the linear space $\mathbb{R}^{n\times n}$?* ♢

Problem 14 *Show that any matrix group of $n \times n$ matrices is a nonlinear subspace of $\mathbb{R}^{n\times n}$. Hint: a linear subspace of a linear space must contain the zero element of the latter.* ♢

Remark 3 *Note well. While a matrix group G is a nonlinear space (as per the problem above), each of its elements, $\boldsymbol{A} \in G$, defines a linear map, from $\mathbb{R}^n$ to $\mathbb{R}^n$, via $\boldsymbol{x} \mapsto \boldsymbol{Ax}$. Thus, we may view a matrix group as a certain nonlinear space of (invertible) linear maps.* ♢

Let $h = g \circ f$ where $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}^k$ are two affine maps:

$$\begin{aligned} f &: \boldsymbol{x} \mapsto \boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{b}_1 \qquad \boldsymbol{A}_1 \in \mathbb{R}^{m\times n} \qquad \boldsymbol{b}_1 \in \mathbb{R}^m\,; \\ g &: \boldsymbol{x} \mapsto \boldsymbol{A}_2\boldsymbol{x} + \boldsymbol{b}_2 \qquad \boldsymbol{A}_2 \in \mathbb{R}^{k\times m} \qquad \boldsymbol{b}_2 \in \mathbb{R}^k\,. \end{aligned} \tag{2}$$

We saw that $h : \mathbb{R}^n \to \mathbb{R}^k$ is affine, with

$$h : \boldsymbol{x} \mapsto \boldsymbol{A}_2\boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{A}_2\boldsymbol{b}_1 + \boldsymbol{b}_2. \tag{3}$$

We also saw that the composition of two invertible affine functions, from $\mathbb{R}^n$ to $\mathbb{R}^n$, is not only affine but also invertible. All this leads us to think about a group structure. Indeed, invertible affine maps, from $\mathbb{R}^n$ to $\mathbb{R}^n$, form a group (the binary operation of the group is composition), called the affine group. Perhaps the easiest way to see this, is when viewing such maps as $(n+1) \times (n+1)$ matrices.

Fact 1 *An affine map, $f : \mathbb{R}^n \to \mathbb{R}^n$, $f : \boldsymbol{x} \mapsto \boldsymbol{Ax} + \boldsymbol{b}$, is identified with an $(n+1) \times (n+1)$ matrix,*

$$f \leftrightarrow \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{0}_{1\times n} & 1 \end{bmatrix} \tag{4}$$

via the connection

$$\begin{bmatrix} f(\boldsymbol{x}) \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{0}_{1\times n} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix}. \tag{5}$$

♢

Problem 15 *Show that the space of* $(n+1) \times (n+1)$ *matrices, of the form*

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{0}_{1\times n} & 1 \end{bmatrix}, \det \boldsymbol{A} \neq 0\,, \tag{6}$$

is a matrix group. This group is known as the affine group. ♢

Problem 16 *Show that the space of* $(n+1) \times (n+1)$ *matrices, of the form*

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{0}_{1\times n} & 1 \end{bmatrix}, \det \boldsymbol{A} > 0\,, \tag{7}$$

is a matrix group (thus, it is a subgroup of the affine group). This group is called the identity component of the affine group, as it contains the identity map. ♢

Problem 17 *Show that the space of* $(n+1) \times (n+1)$ *matrices, of the form*

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{0}_{1\times n} & 1 \end{bmatrix}, \det \boldsymbol{A} < 0\,, \tag{8}$$

does not form a matrix group. ♢

Problem 18 *Let* f *and* g *be two invertible affine maps, from* $\mathbb{R}^n$ *to* $\mathbb{R}^n$*, where* $f : \boldsymbol{x} \mapsto \boldsymbol{A}_0\boldsymbol{x} + \boldsymbol{b}_0$ *and* $g : \boldsymbol{x} \mapsto \boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{b}_1$ *such that* $\det \boldsymbol{A}_0 > 0$ *and* $\det \boldsymbol{A}_1 < 0$*. Does there exist a continuous curve* c*, from the unit interval to the affine group, such that* $c(0) = f$ *and* $c(1) = g$*? In other words, is there a pair of a* t*-dependent* $\boldsymbol{A}(t) \in \mathbb{R}^{n\times n}$ *and a* t*-dependent* $\boldsymbol{b}(t) \in \mathbb{R}^n$*, such that the entries of* $\boldsymbol{A}(t)$ *and* $\boldsymbol{b}(t)$ *depend continuously on* t*, with* $\boldsymbol{A}(0) = \boldsymbol{A}_0$*,* $\boldsymbol{A}(1) = \boldsymbol{A}_1$*,* $\boldsymbol{b}(0) = \boldsymbol{b}_0$*,* $\boldsymbol{b}(1) = \boldsymbol{b}_1$*, and* $\det(\boldsymbol{A}(t)) \neq 0 \forall t \in (0,1)$*? Hint: a determinant of a matrix is a polynomial expression of the matrix entries.* ♢

4 Least Squares and Projections on a Linear Subspace

Problem 19
*In certain situations in computer vision, when the depth variation in the scene is small compared with the distance between the camera and the scene (*e.g.*, satellite images), it is customary to assume a simplified affine camera model (instead of using a more complicated camera model that is based on projective geometry). The latter models the pixel location,* $(u, v) \in \mathbb{R}^2$*, as the following affine projection of the corresponding 3D point,* (X, Y, Z)*:*

$$\begin{bmatrix} u \\ v \end{bmatrix} = \boldsymbol{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{9}$$

where

$$\boldsymbol{P} = \begin{bmatrix} P_{11} & P_{12} & 0 & P_{14} \\ P_{21} & P_{22} & 0 & P_{24} \end{bmatrix} \tag{10}$$

is called the camera matrix. Here, the Z-axis is the "depth" direction; i.e., *it is parallel to the viewing direction from the camera to the scene.* ***In this problem, we will also assume that the values of*** P_{14} ***and*** P_{22} ***are known;*** **i.e.,** $P_{14} = a$ ***and*** $P_{22} = b$ ***where*** a ***and*** b ***are two known scalars.*** *Thus,* $\boldsymbol{P}$ *contains only 4 unknown parameters:* P_{11}, P_{12}, P_{21}, *and* P_{24}. *We will rename these 4 unknowns as* $\theta_1 = P_{11}$, $\theta_2 = P_{12}$, $\theta_3 = P_{21}$, *and* $\theta_4 = P_{24}$. *Thus,* $\boldsymbol{P}$ *becomes*

$$\boldsymbol{P} = \begin{bmatrix} \theta_1 & \theta_2 & 0 & a \\ \theta_3 & b & 0 & \theta_4 \end{bmatrix} . \tag{11}$$

Suppose you are given N *pairs of 2D/3D correspondences; particularly, suppose that for every* $i \in \{1, \ldots, N\}$,

$$\boldsymbol{u}_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} \in \mathbb{R}^2 \qquad \boldsymbol{X}_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \in \mathbb{R}^3 \tag{12}$$

where $\begin{bmatrix} u_i & v_i \end{bmatrix}^T$ *is a 2D observation, and* $\begin{bmatrix} X_i & Y_i & Z_i \end{bmatrix}^T$ *is a known 3D point. Suppose further that the following observation noise model is assumed:*

$$\boldsymbol{u}_i = \boldsymbol{P} \begin{bmatrix} \boldsymbol{X}_i \\ 1 \end{bmatrix} + \boldsymbol{\varepsilon}_i \qquad \boldsymbol{\varepsilon}_i \in \mathbb{R}^2 \qquad \boldsymbol{\varepsilon}_i \overset{iid}{\sim} \mathcal{N}(\boldsymbol{0}_{2\times 1}, \sigma^2 \boldsymbol{I}_{2\times 2}) \qquad \sigma^2 > 0 \,. \tag{13}$$

Thus, the probability density function of $\boldsymbol{\varepsilon}_i$ *is*

$$\mathcal{N}(\boldsymbol{\varepsilon}_i; \boldsymbol{0}_{2\times 1}, \sigma^2 \boldsymbol{I}_{2\times 2}) \triangleq \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\boldsymbol{\varepsilon}_i^T \boldsymbol{\varepsilon}_i}{2\sigma^2} \right) . \tag{14}$$

Let $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix}^T \in \mathbb{R}^4$. *In this problem we view* $\boldsymbol{\theta}$ *as a deterministic and unknown quantity.*

Part (i) Find the least-squares estimator of $\boldsymbol{\theta}$.

Part (ii) Write down the likelihood function for this model. Recall that the likelihood is the probability density function of the data given the value of θ, *and is viewed as a function of* $\boldsymbol{\theta}$; i.e., $\boldsymbol{\theta} \mapsto p((u_i, v_i)_{i=1}^N; \boldsymbol{\theta})$.

Part (iii) Find the maximum-likelihood estimator of $\boldsymbol{\theta}$. ◊

Problem 20 (ridge regression)
Find

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^k}{\arg\min} \, \|\boldsymbol{H}\boldsymbol{\theta} - \boldsymbol{y}\|_{\ell_2}^2 + \lambda \|\boldsymbol{\theta}\|_{\ell_2}^2 \tag{15}$$

where $\boldsymbol{H} \in \mathbb{R}^{N\times k}$, $\boldsymbol{y} \in \mathbb{R}^N$, $\boldsymbol{\theta} \in \mathbb{R}^k$, *and* $\lambda > 0$. ◊

Problem 21 *Let* d *and* k *be two positive integers such that* $d > k$. *Let*

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 & \ldots & \boldsymbol{v}_k \end{bmatrix} \in \mathbb{R}^{d\times k}$$

be an orthogonal matrix; namely,

$$\boldsymbol{V}^T \boldsymbol{V} = \boldsymbol{I}_{k\times k} \,. \tag{16}$$

Let $\boldsymbol{x} \in \mathbb{R}^d$. *Find*

$$\arg\min_{\widehat{\boldsymbol{x}} \in \mathrm{span}(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_k)} \|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|_{\ell_2} \tag{17}$$

(justify your answer). Hint: recall that $\widehat{\boldsymbol{x}} \in \mathrm{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$ *if and only if* $\widehat{\boldsymbol{x}} = \sum_{j=1}^k \theta_j \boldsymbol{v}_j$ *for some* $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_k \end{bmatrix} \in \mathbb{R}^k$, *and note that* $\sum_{j=1}^k \theta_j \boldsymbol{v}_j = \boldsymbol{V}\boldsymbol{\theta}$.

Note well: usually,

$$\boldsymbol{x} \neq \arg\min_{\widehat{\boldsymbol{x}} \in \mathrm{span}(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_k)} \|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|_{\ell_2} . \tag{18}$$

In fact, equality holds if and only if $\boldsymbol{x} \in \mathrm{span}(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k)$. *Otherwise, there is bound to be a difference between* $\widehat{\boldsymbol{x}}$ *and* $\boldsymbol{x}$. *Related to that is the fact (to which we will return later in our class when we discuss linear subspaces and linear dimensionality reduction) that for a* $d \times k$ *"skinny-tall" orthogonal matrix (*i.e., $d > k$*),* $\boldsymbol{V}\boldsymbol{V}^T$ *is* ***not*** $\boldsymbol{I}_{d\times d}$ *despite the fact that* $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_{k\times k}$. *For a trivial example, if* $\boldsymbol{V} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$, *then* $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_{2\times 2}$ *while* $\boldsymbol{V}\boldsymbol{V}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \neq \boldsymbol{I}_{3\times 3}$. ♢

5 Filtering

5.1 Convolution

Let I and h be two 2D digital arrays, of possibly different sizes. We will usually think of I as the input image, and of h as a filter, which is usually smaller.

Definition 8 (The correlation operator) *The* ***correlation operator****,* $\otimes$*, creates a new image,* $I \otimes h$*, via*

$$(I \otimes h)(i,j) = \sum_{k,l} I(i+k, j+l) h(k,l) \overset{\substack{k'=i+k \\ l'=j+l}}{=} \sum_{k',l'} I(k',l') h(k'-i, l'-j) \quad ♢$$

Definition 9 (The convolution operator) *The* ***convolution operator****,* $*$*, creates a new image,* $I * h$*, via*

$$(I * h)(i,j) = \sum_{k,l} I(i-k, j-l) h(k,l) \overset{\substack{k'=i-k \\ l'=j-l}}{=} \underbrace{\sum_{k',l'} I(k',l') h(i-k', j-l')}_{(I \otimes h_{\mathrm{flipped}})(i,j)}$$

$$= (h * I)(i,j) \quad ♢$$

The summation in the definitions above is done over all relevant pixels (*i.e.*, where h is defined), and it is assumed that the central pixel of h is the origin; *e.g.*, if h is 3×3, the notation $\sum_{k,l} I(i-k, j-l)h(k,l)$ is short for

$$\sum_{k=-1}^{1} \sum_{l=-1}^{1} I(i-k, j-l) h(k,l) .$$

Both the convolution and correlation operators are linear.

A slightly different way of writing the same thing is as

$$\sum_{\boldsymbol{x}_i} I(\boldsymbol{x}-\boldsymbol{x}_i)h(\boldsymbol{x}_i) = \sum_{\boldsymbol{x}_i} h(\boldsymbol{x}-\boldsymbol{x}_i)I(\boldsymbol{x}_i) \tag{19}$$

where $\boldsymbol{x}$ is the pixel under consideration. Note that $\boldsymbol{x}$ enters the computation only via its difference from each of the $\boldsymbol{x}_i$'s.

For example, if the filter h is Gaussian, and $g = I * h$, then

$$g(\boldsymbol{x}) = \frac{1}{c}\sum_{\boldsymbol{x}_i} \exp\left(-\frac{1}{2}\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^2}{\sigma^2}\right) I(\boldsymbol{x}_i) \tag{20}$$

where c typically is taken as a normalizer: $c = \sum_{\boldsymbol{x}_i} \exp\left(-\frac{1}{2}\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^2}{\sigma^2}\right)$.

Remark 4 *This expression will also appear when we discuss the Lucas-Kanade method for optical flow.* ♢

Remark 5 *Usually, if $g = I * h$, we consider g only on the same domain as that of I. For example, we do not bother with computing $g(-1, 17)$, since the first argument, -1, is outside the domain of I. A different question, however, is what do when computing, say, $g(1, 17)$ (assuming 1-based indexing), which is a pixel at the boundary of g; the problem is that if h is, say, 3×3, then the summation includes terms such as $I(-1, 17)h(-1, 0)$, $I(-1, 16)h(-1, -1)$, and $I(-1, 18)h(-1, 1)$ but $I(-1, 16)$, $I(-1, 17)$, and $I(-1, 18)$ are undefined. In other words, we need to decide how to handle boundary effects. There are several ways to go about it. For example, treating I as zero at every out-of-domain pixel. In this class (unlike standard classes on signal/mage processing) we will not explore the implications of the various choices one can make for handling the boundary effects.* ♢

Problem 22 (convolution is associative) *Let a, b and c be three digital images. Show that*

$$(a * b) * c = a * (b * c)$$

(thus we can drop the parentheses and simply write $a * b * c$). ♢

Definition 10 (2D discrete-domain impulse signal)

$$\delta(i, j) = \begin{cases} 1 & \text{if } i = 0 \text{ and } j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

♢

A filter h is also called the "impulse response" (of some system) since its convolution with an impulse signal is $h * \delta = \delta * h = h$.

Problem 23 *Show that $h * \delta = \delta * h = h$.* ♢

Computer Exercise 1 *The image used here is **mandrill.png**. Display it and the results for convolving it with each one of the following filters: an isotropic 7×7 Gaussian with $\sigma = 3$; an isotropic 21×21 Gaussian with $\sigma = 10$; a uniform 21×21 blur kernel (*e.g.*, in Python: h=np.ones((21,21))/(21**2)).* ♢

`Some Python commands you may find useful for the exercise above: scipy.ndimage.gaussian_filter; scipy.ndimage.uniform_filter; cv2.blur,cv2.GaussianBlur.`

Remark 6 *A mandrill is a monkey whose central part of the nose is distinctly RED, surrounded by a blue region. Any one who watched The Lion King would know that. If you end up with the central part of the nose looking bluish, this suggests you are confusing BGR and RGB ordering. Particularly, matplotlib functions (imread, imshow, etc.) assume RGB ordering by default, while opencv's counterparts assume BGR. So if,* e.g.*, you imread using one library and imshow using the other, you will end up displaying colors in the wrong order.*

Of course, you can always revert the ordering if you want to mix the libraries, img=img[:,:,::-1], to move from to the other. Alternatively, some functions (e.g. opencv's imread) can also take an optional argument which forces BGR or RGB ordering.This seems like a lot of hassle though, so I would suggest sticking with one library at the time, unless there is a compelling reason not to do it. ◊

5.1.1 Viewing convolution as as one big matrix operation

Let $x \in \mathbb{R}^{M\times N}$ be an input image. Let $h \in \mathbb{R}^{m\times n}$. Let $y = x * h$. Since convolution is linear, and since y is also $M \times N$, we may rewrite this operation as

$$\underbrace{\boldsymbol{y}}_{(MN)\times 1} = \underbrace{\boldsymbol{H}}_{(MN)\times(MN)} \underbrace{\boldsymbol{x}}_{(MN)\times 1} \tag{22}$$

where $\boldsymbol{x}$ is a vectorized version of x and $\boldsymbol{y}$ is a vectorized version of y. The values of $\boldsymbol{H}$ are determined by h. If h is small (or large but sparse) then $\boldsymbol{H}$ is sparse (*i.e.*, most of its elements are zero).

Problem 24 *Assume a zero-boundary condition. In effect, in 1-based indexing, we set $I(i-k, j-l) = 0$ whenever at least one of the following is true: $i-k \le 0$; $i-k > M$; $j-l \le 0$; $j-l > N$. Assume the vectorization is done by stacking the rows on top of each other, and that $h \in \mathbb{R}^{3\times 3}$:*

$$\begin{aligned}
\boldsymbol{x} &= \begin{bmatrix} x(1,1) & \dots & x(1,N) & \dots & x(M,1) & \dots & I(M,N) \end{bmatrix}^T ; \\
\boldsymbol{y} &= \begin{bmatrix} y(1,1) & \dots & y(1,N) & \dots & y(M,1) & \dots & y(M,N) \end{bmatrix}^T ; \\
h &= \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ h_{2,1} & h_{2,2} & h_{2,3} \\ h_{3,1} & h_{3,2} & h_{3,3} \end{bmatrix} \in \mathbb{R}^{3\times 3} .
\end{aligned}$$

Write the expression for the entries of $\boldsymbol{H}$; i.e.*, write down the expression for $\boldsymbol{H}_{i,j}$ for every i and j such that $i \in \{1, \dots, MN\}$ and $j \in \{1, \dots, MN\}$ where $\boldsymbol{H}_{i,j}$ is the entry of $\boldsymbol{H}$ in the i^{th} row and j^{th} column. Note that most these entries are zero;* i.e.*, $\boldsymbol{H}$ is (very) sparse.* ◊

Computer Exercise 2 *In continue to the previous exercise, let $M = 12$ and $N = 16$. Implement $\boldsymbol{H}$ for the three different filters,*

$$h_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h_2 = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad h_3 = h_2^T , \tag{23}$$

and display (using imshow with "interpolation" set to "None"; use colorbar and informative titles) the resulting $\boldsymbol{H}$ *matrices (as opposed to, say, displaying some image convolved with* h*).*

Also note that: h_1 *is a uniform blur filter;* h_2 *is the horizontal Sobel filter;* h_3 *is the vertical Sobel filter.*

Remark: if M *and* N *are much larger (e.g.,500), we can still easily implement* $\boldsymbol{H}$ *using a sparse matrix (e.g., using scipy.sparse.lil). But for displaying purposes, we will have to convert it to a dense matrix, and then memory might be an issue.* ◊

`Some Python commands you may find useful in the exercise above: a.ravel; a.reshape (when a is a numpy array); plt.title; plt.colorbar.`

Problem 25 *Let*

$$h = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{24}$$

What is the effect of convolving an image with this h*? (optional: try it yourself on a computer to verify your answer)* ◊

5.1.2 Separable Filters

If we are given two 1D filters, we can build a separable filter from them (see the practical session). Now we will address the other direction: given a 2D filter, we would like to know whether it is separable or not. We will do this using SVD (defined below).

Definition 11 (a square orthogonal matrix) *An* $n \times n$ *matrix* $\boldsymbol{Y}$ *is called orthogonal if* $\boldsymbol{Y}\boldsymbol{Y}^T = I_{n\times n}$. ◊

Problem 26 *Let* $\boldsymbol{Y}$ *be an orthogonal square matrix. Let* $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ *denote the columns of* $\boldsymbol{Y}$*.*

Part (i) *Can you uniquely determine* $\det(\boldsymbol{Y})$*? If not, is there still something you can say about* $\det(\boldsymbol{Y})$*?*

Part (ii) *Show that* $\boldsymbol{Y}^T$ *is an orthogonal square matrix.*

Part (iii) *Let* $i \in \{1, \ldots, n\}$*. Find* $\boldsymbol{y}_i^T \boldsymbol{y}_i$*.*

Part (iv) *Let* $i \in \{1, \ldots, n\}$*. Find* $\|\boldsymbol{y}_i\|_{\ell_2}$*.*

Part (v) *Let* $i, j \in \{1, \ldots, n\}$ *where* $i \neq j$*. Find* $\boldsymbol{y}_i^T \boldsymbol{y}_j$*. What is the angle between* $\boldsymbol{y}_i$ *and* $\boldsymbol{y}_j$*?*

Part (vi) *Fix* n*, a positive integer. Show that* $n \times n$ *orthogonal matrices form a matrix group*[1]*. This group is called the orthogonal group – that is the standard name, but perhaps the "orthonormal group" would be a more informative name.* ◊

[1] For the definition of a matrix group, see the beginning of HW #5.

Definition 12 (the Singular Value Decomposition (SVD) of a square matrix) *The SVD of an $n \times n$ matrix $\boldsymbol{A}$ is*

$$\boldsymbol{A} = \underbrace{\boldsymbol{U}}_{n\times n}\underbrace{\boldsymbol{S}}_{n\times n}\underbrace{\boldsymbol{V}^T}_{n\times n} \tag{25}$$

$\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal square matrices. Columns of $\boldsymbol{U}$ (resp. $\boldsymbol{V}$) are called the left (right) singular vectors of $\boldsymbol{A}$. The matrix $\boldsymbol{S}$ is diagonal. Its diagonal elements, $\{s_i\}_{i=1}^n$, are nonnegative and called the singular values of $\boldsymbol{A}$. SVD routines usually return the singular values sorted with $s_1 \geq s_2 \geq s_n$. ♢

Fact 2 *Let $\boldsymbol{K}$ be a square 2D filter. It is separable if and only if its SVD has exactly one non-zero singular value.* ♢

Example 1 *The SVD of a 3×3 $\boldsymbol{K}$ is*

$$\begin{aligned}\boldsymbol{K} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T &= \left[\begin{array}{ccc} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 \end{array}\right] \left[\begin{array}{ccc} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{array}\right] \left[\begin{array}{ccc} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{array}\right]^T \\ &= \sum_{i=1}^{3} s_i \boldsymbol{u}_i \boldsymbol{v}_i^T \,. \end{aligned} \tag{26}$$

If $s_1 > 0$ and $s_2 = s_3 = 0$ then

$$\boldsymbol{K} = s_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T \tag{27}$$

and $\boldsymbol{K}$ is evidently separable, with, e.g.*,*

- *a vertical filter $\sqrt{s_1}\boldsymbol{u}_1$.*
- *a horizontal filter $\sqrt{s_1}\boldsymbol{v}_1^T$.* ♢

Computer Exercise 3 *Get K_1, K_2 and K_3 from* `are_these_separable_filters.mat`*. Using svd, determine, based on the singular values of these filters, which one is separable and which is not. Remark: due to numerical errors, you are not going to get perfect zeros even if the filter is separable. You will have to settle for treating very small numbers (*e.g.*, 10^{-12}) as zero.* ♢

`Some Python commands you may find useful for the exercise above: scipy.linalg.svd, np.diag, np.allclose.`

Fact 3 *If $\boldsymbol{A}$ and $\boldsymbol{B}$ are two, possibly-rectangular, matrices such that their matrix product, AB, is defined, then*

$$\mathrm{rank}(AB) \leq \min(\mathrm{rank}(A), \mathrm{rank}(B)) \,.$$

♢

Problem 27 *Let $\boldsymbol{K}$ be an $n \times n$ separable filter. Is the $n \times n$ matrix $\boldsymbol{K}$ invertible? (note well, you are not asked whether the operation of convolving an image with $\boldsymbol{K}$ is invertible).* ♢

5.2 More General Filters

The convolution operator (similarly to the correlation operator) is not just a linear operation, it is also space invariant (also called location invariant).

More generally, we can define the following operation: I and h is given by

$$g(i,j) = \sum_{k,l} I(i-k, j-l) h_{i,j}(k,l)\,. \tag{28}$$

The difference from convolution is that here the filter depends on the location, (i,j). For example, consider a situation where we want to blur the image but also have reasons to expect more noise on the right part of the image. In which case, we may want to use a Gaussian filter such that its standard deviation increases with x. While this filter is space-dependent (and thus applying it is not a convolution with some filter), this filter is still linear and still does not depend on I.

An even more general approach is to adapt the filter to the image itself; *i.e.*, make it a function of I. For example, suppose we want to blur an image but do it in a way that preserves edges (more on that later in this document). There are several ways to go about it. One of them is what is called bilateral filtering:

$$g(\boldsymbol{x}) = \frac{1}{c} \sum_{\boldsymbol{x}_i} f^r(|I(\boldsymbol{x}) - I(\boldsymbol{x}_i)|) f^d(|\boldsymbol{x} - \boldsymbol{x}_i|) I(\boldsymbol{x}_i) \tag{29}$$

$$c = \sum_{\boldsymbol{x}_i} f^r(|I(\boldsymbol{x}) - I(\boldsymbol{x}_i)|) f^d(|\boldsymbol{x} - \boldsymbol{x}_i|) \tag{30}$$

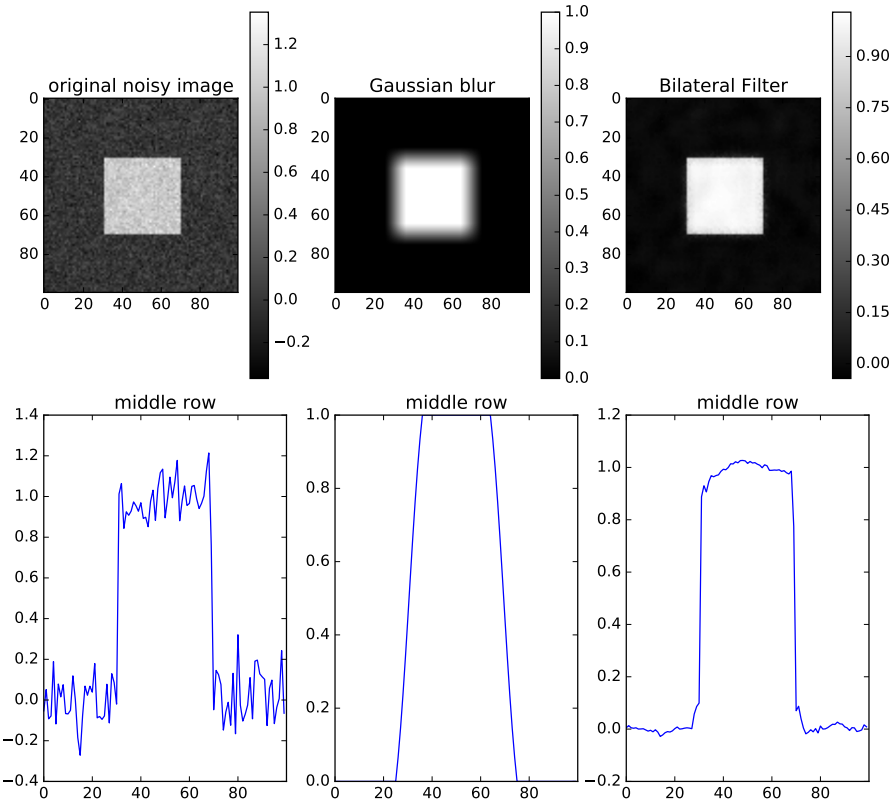
where f_r controls the weight as a function of the distance in the range (*i.e.*, intensity difference) and f_d controls the weight as as function of the distance in the domain (*i.e.*, spatial distance) For example we can take both these to be Gaussians, of different variances:

$$g(\boldsymbol{x}) = \frac{1}{c} \sum_{\boldsymbol{x}_i} \exp\left(-\frac{1}{2}\frac{I(\boldsymbol{x}) - I(\boldsymbol{x}_i))^2}{\sigma_r^2}\right) \exp\left(-\frac{1}{2}\frac{(\boldsymbol{x} - \boldsymbol{x}_i)^2}{\sigma_d^2}\right) I(\boldsymbol{x}_i) \tag{31}$$

$$c = \sum_{\boldsymbol{x}_i} \exp\left(-\frac{1}{2}\frac{I(\boldsymbol{x}) - I(\boldsymbol{x}_i))^2}{\sigma_r^2}\right) \exp\left(-\frac{1}{2}\frac{(\boldsymbol{x} - \boldsymbol{x}_i)^2}{\sigma_d^2}\right)\,. \tag{32}$$

Problem 28 *Is bilateral filtering a linear operation? Explain.* ◊

The main purpose of the bilateral filter is to preserve edges while blurring, as shown in the figure below:



Computer Exercise 4 *Get the (noisy) image from* `bilateral.mat` *(shown in the left column in the figure above), and, using bilateral filtering and playing with its parameters, get a result similar to the one in the rightmost figure above.* ♢

`A Python command you may find useful for the exercise above: cv2.bilateralFilter`

5.3 Laplacian and Laplacian of Gaussian

Problem 29 *Calculate, by taking analytical derivatives, the Laplacian of an isotropic Gaussian (in a continuous setting);* i.e.*,*

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

where

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

and show it is indeed given by

$$\nabla^2 G(x,y,\sigma) = \left(\frac{x^2+y^2}{\sigma^4} - \frac{2}{\sigma^2}\right) G(x,y,\sigma)\,.$$

(an expression you saw in class). ♢

Since $G(x,y,\sigma) = G(x^2+y^2,\sigma)$ (a sloppy notation indicating that $G(x,y,\sigma)$ depends on x and y only through $r^2 \triangleq x^2+y^2$) and since $\left(\frac{x^2+y^2}{\sigma^4} - \frac{2}{\sigma^2}\right)$ can be written as another function of x^2+y^2 (and σ), it is obvious that, at a given point (x,y), $\nabla^2 G(x,y,\sigma)$ is invariant w.r.t. spatial rotations of I about this point. It is tempting to think that this is solely because G is rotational invariant. However, it turns out that the Laplacian of I itself is also rotationally invariant:

Problem 30 *Let*

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}$$

and show that, independently of the value of $\theta \in [0, 2\pi)$, it always holds (assuming I is twice differentiable) that

$$\nabla^2 I(x,y) = \nabla^2 I(x'(x,y), y'(x,y))\,,$$

Equivalently,

$$\frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2} = \frac{\partial^2 I(x'(x,y),y'(x,y))}{\partial (x')^2} + \frac{\partial^2 I(x'(x,y),y'(x,y))}{\partial (y')^2}\,.$$

$\Diamond$

Two 3×3 simple discrete approximations of the Laplacian operator are convolutions with one of the following filters:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}. \tag{33}$$

Consistently with the previous exercise, we can see some (approximated) radial symmetry (more so in the second filter).

5.4 Directional Filters

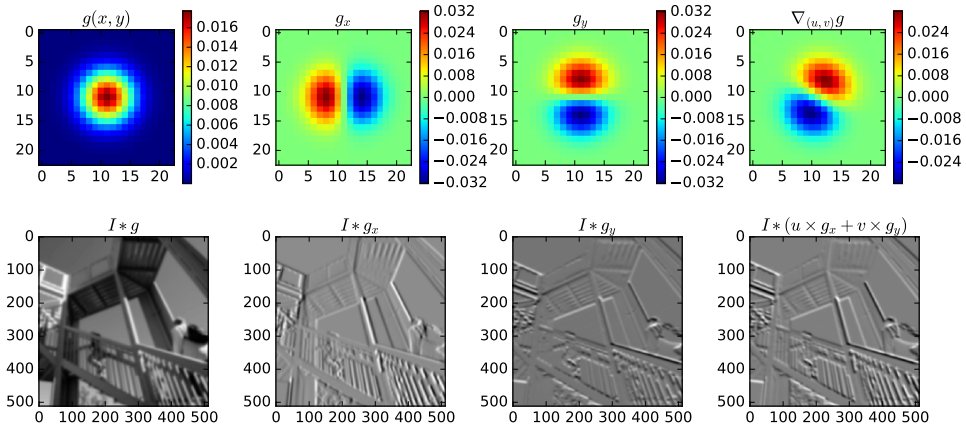
Let

$$g(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}\right).$$

Note that:

$$\frac{\partial}{\partial x} g(x,y) = -\frac{x}{\sigma^2} g(x,y) \qquad \frac{\partial}{\partial y} g(x,y) = -\frac{y}{\sigma^2} g(x,y) \tag{34}$$

The top row of the figure below shows g, $\frac{\partial}{\partial x} g$, $\frac{\partial}{\partial x} g$ and $u\frac{\partial}{\partial x} g + v\frac{\partial}{\partial y} g$ where $u = \cos\theta$ and $v = \sin\theta$ with $\theta = 115°$. In other words, the last filter provides an approximated directional derivative in the (u,v) direction. The second row shows the results of convolving an image (from `ascent.jpg`) with these filters.



Computer Exercise 5 *The image used here is* `ascent.jpg`. *Create 5 filters that correspond to the directional derivatives in 5 different angles of your choosing. Show the filters and the results of applying them to the image.* ♢

`Some Python commands you may find useful in the exercise above: np.mgrid, np.exp,np.cos,np.sin, a.sum() (where a is a numpy array), ndimage.convolve, plt.subplots_adjust`

5.5 Computing the Spatial Derivatives (will be Required in a Later Assignment for Optical Flow)

Computer Exercise 6 *Get img1 from* `imgs_for_optical_flow.mat`. *Using derivative filters, compute, for I=img1, the following images:*

$$I_x = \frac{\partial}{\partial x} I = I * h_x, \quad I_y = \frac{\partial}{\partial y} I = I * h_y,$$
$$I_{xx} = \frac{\partial^2}{\partial x^2} I = I * h_{xx}, \quad I_{yy} = \frac{\partial^2}{\partial y^2} I = I * h_{yy}\,.$$

where h_x is a filter which approximates the derivative in the x direction, h_y is a filter which approximates the derivative in the y direction, h_{xx} is a filter which approximates the second derivative in the x direction, and h_{yy} is a filter which approximates the second derivative in the y direction. You may want to blur[1] *I a little by convolving it with g_0, a Gaussian of small standard deviation, σ_0, before taking these derivatives, or, equivalently, convolve I with the derivatives of g_0.* ♢

`A Python command you may find useful for the exercise above: cv2.getDerivKernels.`

`Note that the output of cv2.getDerivKernels will in fact be two 1D filters,`

[1] To avoid amplifying noise and/or spatial "aliasing" – the latter is a topic covered in signal/image processing classes.

```
which you can use as input to cv2.sepFilter2D. Alternatively, you can take
these two 1D filters and create a 2D filter from them:
h1,h2=cv2.getDerivKernels(1,0,3)
h1.shape is (3, 1)
h2.shape is (3, 1)
h2.dot(h1.T) then gives
array([[-1.,  0.,  1.],
       [-2.,  0.,  2.],
       [-1.,  0.,  1.]], dtype=float32)

Regarding how to compute the second derivative:
In principle, you can use cv2.sobel twice if you want. But it is better to do
it in a single operation.E.g., suppose you are using 3x3 filters.
cv2.getDerivKernels(1,0,3) gives you the "d/dx" filter.
cv2.getDerivKernels(0,1,3) gives you the "d/dy" filter.
cv2.getDerivKernels(2,0,3) gives you the "d^2/dx^2" filter.
cv2.getDerivKernels(0,2,3) gives you the "d^2/dy^2" filter.
```