Module: **M4. 3D Vision** **Final exam**
Date: February 25, 2021
Teachers: Antonio Agudo, Josep Ramon Casas, Pedro Cavestany, Gloria Haro, Javier Ruiz, Federico Sukno
Time: 2h

Problem 1 *1.8 Points*

Geometric transformations and rectification

(a) Suppose an affinely rectified image. Let $\mathbf{l} = (l_1, l_2, l_3)$ and $\mathbf{m} = (m_1, m_2, m_3)$ be the image of two lines that are orthogonal in the world. Describe the steps of the algorithm that performs the metric rectification of the image from $\mathbf{l}$ and $\mathbf{m}$.

(b) List the 3D transformations that an object can undergo. For each transformation you should explain the anatomy of their operator, their degrees of freedoms and their invariants.

(a) The steps of the algorithm that performs the metric rectification of the image from $\mathbf{l}$ and $\mathbf{m}$ are:

(i) $\mathbf{l}^T M \mathbf{m} = 0$. This equation writes

$$(l_1 m_1, l_1 m_2 + l_2 m_1, l_2 m_2)\vec{s} = 0,$$

where $\vec{s}^T = (s_1, s_2, s_3)^T$ is the vector with the entries of the 2×2 symmetric matrix

$$S = AA^T = \begin{pmatrix} s_1 & s_2 \\ s_2 & s_3 \end{pmatrix}$$

(ii) **The image of two such orthogonal pairs provide two equations and they permit to compute** (s_1, s_2, s_3) as its null vector (is an homogenous system of two equations with three unknowns).

(iii) **Knowing S, use the Cholesky decomposition** of S to compute **an upper triangular matrix** K such that $S = KK^T$.

(iv) **The matrix K is a possible matrix A that can be used to metrically rectify the image**.

(v) Define $H_{a \leftarrow s} = \begin{pmatrix} K & \vec{0} \\ \vec{0}^T & 1 \end{pmatrix}$ and $H = H_{\mathrm{a}\leftarrow\mathrm{s}}^{-1} H_{\mathrm{a}}$, where

$$H_{\mathrm{a}\leftarrow\mathrm{s}}^{-1} = H_{\mathrm{s}\leftarrow\mathrm{a}} = \begin{pmatrix} K^{-1} & \vec{0} \\ \vec{0}^T & 1 \end{pmatrix}$$

Then, $H\Omega_\infty H^T = \Omega_\infty$. and therefore H is a similarity. Let's denote it by H_{s}. That is, $H_{\mathrm{a}} = H_{\mathrm{a}\leftarrow\mathrm{s}} H_{\mathrm{s}}$.

(vi) The rectified image can be defined by

$$u_{\text{metrect}}(\vec{x}_s) = u_{\text{affrect}}([H_{\text{a}\leftarrow\text{s}}\mathbf{x}_s]),$$

where $\mathbf{x}_s = (\vec{x}_s, 1)$ and $[\cdot]$ denotes $[(x_1, x_2, x_3)] = (x_1/x_3, x_2/x_3)$ when $x_3 \neq 0$.

(b) Let $\mathbf{X}$ and $\mathbf{X}'$ be homogeneous coordinates in $\mathbb{P}^3$. The 3D transformations that can be apply between them are:

(i) Isometry
Anatomy:

$$\mathbf{X}' = H_e\mathbf{X} = \begin{pmatrix} R & \vec{t} \\ \vec{0}^T & 1 \end{pmatrix}\mathbf{X}.$$

where H_e is a 4×4 matrix,
R is a rotation 3×3 matrix (orthogonal matrix) and
$\vec{t}$ is a 3×1 translation vector.
Degrees of freedom: $6 \rightarrow 3$ for the rotation angles + 3 for the translation coefficients
Invariants: lengths, angles.

(ii) Similarity
Anatomy:

$$\mathbf{X}' = H_s\mathbf{X} = \begin{pmatrix} sR & \vec{t} \\ \vec{0}^T & 1 \end{pmatrix}\mathbf{X}.$$

where H_e is a 4×4 matrix,
s is an isotropic scaling factor,
R is a rotation 3×3 matrix (orthogonal matrix) and
$\vec{t}$ is a translation vector.
Degrees of freedom: $7 \rightarrow 3$ for the rotation angles + 3 for the translation coefficients + 1 for scaling factor
Invariants: angles, ratios, the absolute conic Ω_∞.

(iii) Affine transformation
Anatomy:

$$\mathbf{X}' = H_a\mathbf{X} = \begin{pmatrix} A & \vec{t} \\ \vec{0}^T & 1 \end{pmatrix}\mathbf{X}.$$

where A is a non-singular 3×3 matrix ,
$\vec{t}$ is a translation vector.
Degrees of freedom: $12 \rightarrow 9$ for A + 3 for the translation coefficients
Invariants: parallelism, ratio of two volumes, the plane at infinity $\mathbf{\Pi}_\infty$.

(iv) Projective transformation
Anatomy:

$$\mathbf{X}' = H_p\mathbf{X} = \begin{pmatrix} A & \vec{t} \\ \vec{v}^T & v \end{pmatrix}\mathbf{X}.$$

where H_p is a homography

$$H_p = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{pmatrix}$$

Degrees of freedom: $15 \rightarrow 4 \times 4$ elements - 1 multiplicative factor
Invariants: concurrency, collinearity, order of contact, cross ratio.

Problem 2 *1.2 Points*

Calibration and pose estimation

(a) Describe the camera resectioning problem, clearly state which are the unknowns and what is the available data.

(b) Derive the expression of the matrix that describes the linear system of equations in the algebraic method for resectioning (DLT method). Justify what is the minimum amount of data we need to solve the problem.

(c) Enumerate two practical scenarios where we can use camera resectioning methods.

(d) What is the main difference between camera resectioning and pose estimation?

(a) Camera resectioning is the estimation of the camera projection matrix that describes a pinhole camera model. The unknowns are thus the 12 elements of the camera matrix. Usually, the available data is a set of 3D to 2D correspondences.

(b) We have a set of correspondences $\mathbf{x_i} \longleftrightarrow \mathbf{X_i}$, where $\mathbf{x_i} \in \mathbb{P}^2$ and $\mathbf{X_i} \in \mathbb{P}^3$. For each correspondence, we have that $\mathbf{x_i} = P\mathbf{X_i}$. Since this equation is written in projective coordinates we have that $\mathbf{x}_i = (x_i, y_i, 1)^T$ and $P\mathbf{X}_i$ are colinear. We can express the colinearity in two ways:

(i) Using an unknown scalar factor λ:

$$\lambda \begin{pmatrix} x_i \\ y_i \\ w_i \end{pmatrix} = P\,\mathbf{X}_i = \begin{pmatrix} p_1^T \\ p_2^T \\ p_3^T \end{pmatrix} \mathbf{X}_i,$$

where p_1^T, p_2^T, p_3^T denote the three rwos of matrix P.

(ii) Using the cross product:

$$\mathbf{x}_i \times P\mathbf{X}_i = 0. \tag{1}$$

We develop the equations in (i) or in (ii) and we get:

$$\underbrace{\begin{pmatrix} \mathbf{0}^T & -w_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \end{pmatrix}}_{A_i} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

So we can see that from a correspondence we get 2 equations. As matrix P has 11 degrees of freedom (12 elements minus the scaling factor), we need a minimum of 6 correspondences to estimate P. If we use $N \geq 6$ correspondences, the final system of equations writes:

$$\underbrace{\begin{pmatrix} A_1 \\ \dots \\ A_N \end{pmatrix}}_{2N\times 12} \underbrace{\begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix}}_{12\times 1} = \underbrace{\mathbf{0}}_{2N\times 1}.$$

(c) (i) Calibration of a camera with a 3D calibration object.

(ii) Calibration of a new camera in an incremental structure from motion procedure once we have already estimated a set of 3D points.

(d) In the pose estimation problem the internal parameters of teh camera(intrinsics) are known and we only need to estimate the position and orientation (extrinsics) of the camera. In resectioning both intrinsics and extrinsics are unknown.

Problem 3 *1.8 Points*

We plan to estimate the fundamental matrix F between two images I and I' taken by the same camera using the 8-point algorithm.

a) Given two correspondences $p_1 = (1, 2)$, $p'_1 = (3, 4)$ and $p_2 = (5, 6)$, $p'_2 = (7, 8)$ write the first 2 rows of the matrix W that allows us to estimate the fundamental matrix F (expressed as a vector column f) with a homogeneous system ($Wf = 0$).

Suppose that the singular value decomposition of the matrix W above can be expressed as:

$$W = \begin{bmatrix} 0 & 0 & -1 & 1 & -1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 1 \\ -1 & 1 & 0 & 1 & -1 & 1 & 1 & -1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ -1 & 1 & 0 & 0 & -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & -1 & 1 & -1 & 1 \end{bmatrix} D \begin{bmatrix} 1 & 0 & -1 & 1 & -1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 1 & 1 & 1 \\ -1 & 1 & 0 & 1 & -1 & 1 & 1 & -1 & 1 \\ 1 & 0 & -1 & 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & -1 & 1 & -1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & -1 & 1 & -1 & 1 \end{bmatrix}^T$$

b) Obtain a first approximation of the fundamental matrix.

Suposse now than the singular value decomposition of the fundamental matrix obtained in the previous question is:

$$F_b = \begin{bmatrix} 0 & -1 & 0 \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ 0 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}^T$$

c) Obtain a second approximation of the fundamental matrix that ensures all the epipolar lines cross at the same point (epipole).

d) Is $e = (0, -1)$ the epipole in Image I?

e) Taken into account the fundamental matrix obtained above, are the two images I and I' rectified?

f) Compute the epipolar line l' in image I' for point $p_1 = (1, 2)$.

a) $W = \begin{bmatrix} 1*3 & 2*3 & 3 & 1*4 & 2*4 & 4 & 1 & 2 & 1 \\ 5*7 & 6*7 & 7 & 5*8 & 6*8 & 8 & 5 & 6 & 1 \\ & & & \cdots & & & & & \end{bmatrix} = \begin{bmatrix} 3 & 6 & 3 & 4 & 8 & 4 & 1 & 2 & 1 \\ 35 & 42 & 7 & 40 & 48 & 8 & 5 & 6 & 1 \\ & & & \cdots & & & & & \end{bmatrix}$

b) Last column of V, $F_b = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$

c) $F = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & \frac{\sqrt{2}}{2} \\ -1 & 0 & \frac{\sqrt{2}}{2} \end{bmatrix}$

d) No, as $F\tilde{e}$ is not 0.

e) No, as epipoles should be at infinity in rectified images.

f) $l' = F\tilde{p}_1 \rightarrow 3u' + (-1 + \frac{\sqrt{2}}{2})v' + (-1 + \frac{\sqrt{2}}{2}) = 0$

Problem 4 *1.5 Points*

Depth estimation and new view synthesis.

(a) Describe the main steps for estimating the depth from a pair of stereo rectified images with a local method. How is disparity related to depth?

(b) Define the Sum of Squares Differences and Normalized Cross Correlation. Compare their advantages/disadvantages.

(c) Describe different possible failure cases when estimating the disparity through local methods and two views. What are possible solutions to these problems?

(d) Explain what are the inputs and outputs of the Neural Radiance Field (NeRF) method. Which is the loss used to train the network?

(a) The disparity is inversely proportional to depth, more precisely, $disparity = focal\,length \cdot baseline/depth$. The local methods estimate the disparity independently for every pixel by comparing the image content in the left and right image inside a local neighborhood (window/patch). A matching cost is used to estimate the dissimilarity of the content in the two windows. The basic steps are described below.

For every pixel in the left image:

- Slide a window along the same line in the right image and compare its content to that of the reference window in the left image.
- Pick the disparity with minimum matching cost (WTA: Winner-Takes-All approach).

(b) SSD: Sum of Squared Differences ($m = 2$)

$$C(\mathbf{p},\mathbf{d}) = \sum_{\mathbf{q}\in N_\mathbf{p}} w(\mathbf{p},\mathbf{q})\,|I_1(\mathbf{q}) - I_2(\mathbf{q}+\mathbf{d})|^m, \qquad \sum_{\mathbf{q}\in N_\mathbf{p}} w(\mathbf{p},\mathbf{q}) = 1$$

$N_\mathbf{p} = \{\mathbf{q} = (q_1,q_2)^T \,|\, p_1 - \frac{n}{2} \le q_1 \le p_1 + \frac{n}{2},\ p_2 - \frac{n}{2} \le q_2 \le p_2 + \frac{n}{2}\}$

NCC: Normalized Cross Correlation

$$NCC(\mathbf{p},\mathbf{d}) = \frac{\sum_{\mathbf{q}\in N_\mathbf{p}} w(\mathbf{p},\mathbf{q})(I_1(\mathbf{q}) - \bar{I}_1)(I_2(\mathbf{q}+\mathbf{d}) - \bar{I}_2)}{\sigma_{I_1}\sigma_{I_2}}$$

$\bar{I}_1 = \sum_{\mathbf{q}\in N_\mathbf{p}} w(\mathbf{p},\mathbf{q}) I_1(\mathbf{q}); \quad \bar{I}_2 = \sum_{\mathbf{q}\in N_\mathbf{p}} w(\mathbf{p},\mathbf{q}) I_2(\mathbf{q}+\mathbf{d});$

$\sigma_{I_1} = \sqrt{\sum_{\mathbf{q}\in N_\mathbf{p}} w(\mathbf{p},\mathbf{q})(I_1(\mathbf{q}) - \bar{I}_1)^2}$; $\sigma_{I_2} = \sqrt{\sum_{\mathbf{q}\in N_\mathbf{p}} w(\mathbf{p},\mathbf{q})(I_2(\mathbf{q}+\mathbf{d}) - \bar{I}_2)^2}$

The SSD is faster to compute than the NCC. The NCC is more robust to illumination changes but loses discriminatory power compared to SSD.

(c) Some problems are:

1) Textureless areas
2) Occlusions
3) Repetitions (self-similarity)
4) Non-Lambertian surfaces, specularities

The use of a 3rd view may help with problems 2) and 4). Increasing the smoothness with larger window size helps with 1), 3) and 4) (bilateral weights can be used to keep sharpness).

(d) The inputs are the three coordinates of a 3D point and the two angles of a viewing direction in the 3D space. The output is the volume density and view-dependent emitted radiance at the location expressed by the input. The training process uses a set of images with known camera poses. With the help of classic volume rendering techniques, views are synthesized by projecting output colors and densities into an image. The L_2 norm of the synthesized image and the ground truth one is the training loss.

Problem 5 *1.4 Points*

Formulate the structure-from-motion problem by assuming P 2D point tracks in a monocular video with I images (indicating the corresponding size of every matrix) in terms of a measurement matrix

W, assuming: 1) a perspective camera and a rigid shape, 2) an orthographic camera and a non-rigid shape (you can assume a linear low-rank shape model of rank K). You should provide the entries in every matrix in the most representative way as possible.

To compute shape and motion by factorization, which rank do we have to enforce in every case and how can this be done?

The problem in 1) is:

$$\underbrace{\begin{bmatrix} \mathbf{x}_1^1 & \dots & \mathbf{x}_P^1 \\ 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ \mathbf{x}_1^I & \dots & \mathbf{x}_P^I \\ 1 & \dots & 1 \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} \mathbf{P}^1 \\ \vdots \\ \mathbf{P}^I \end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix} \mathbf{X}_1 & \dots & \mathbf{X}_P \\ 1 & \dots & 1 \end{bmatrix}}_{\mathbf{X}}$$

where the size of every matrix in order of appearance is: 3IxP, 3Ix4 and 4xP. Every P is 3x4, and it is to include both rotation and translation entries.

The problem in 2) is:

$$\underbrace{\begin{bmatrix} \mathbf{x}_1^1 & \dots & \mathbf{x}_P^1 \\ \vdots & \ddots & \vdots \\ \mathbf{x}_1^I & \dots & \mathbf{x}_P^I \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} l_1^1\mathbf{R}^1 & \dots & l_K^1\mathbf{R}^1 \\ \vdots & \ddots & \vdots \\ l_1^I\mathbf{R}^I & \dots & l_K^I\mathbf{R}^I \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} \mathbf{b}_{11} & \dots & \mathbf{b}_{1P} \\ \vdots & \ddots & \vdots \\ \mathbf{b}_{K1} & \dots & \mathbf{b}_{KP} \end{bmatrix}}_{\mathbf{B}}$$

where the size of every matrix in order of appearance is: 2IxP, 2Ix3K and 3KxP. l_k^1 and $\mathbf{b}_k$ are to encode weight coefficients and shape bases in the linear subspace, respectively.

In both cases we will use a SVD factorization, enforcing a rank of 4 and 3K, respectively.

Problem 6 *1.4 Point*

In 2022, a production company in Barcelona named ReVi commits to creating contents for *"Real Virtuality"*, a new line of VR series promoted in a joint initiative of HBO and Netflix media platforms. ReVi will **capture real world** scenes with **2D and 3D sensor**s to be later rendered for virtual reality glasses. The *real* look of the 3D rendered scenes is expected to beat contents from synthetic graphics and attract new audiences towards the VR sector, which has focused more in gaming than in content from real scenes until present.
ReVi decides to hire specialists from the Computer Vision Master and request from them to answer a few questions first. Could you provide short answers to their questions?

a) What kind of sensors should ReVi use to capture 3D outdoor scenes? Should they combine different sensor types, such as adding standard RGB cameras? If so, why would they need to combine such sensors and how can we achieve this "fusion" of 2D/3D captured data?

b) How many sensors should be used in order for the content to be freely navigated (Free Viewpoint) when displayed on VR glasses? Should ReVi care about the number of sensors capturing the scene?

c) Would it be useful to capture the background of an empty set separately, when talent (actors and actresses) are not performing? Which are the cases in which this would work best? How could the foreground (performers and foreground scene objects) added to the already captured background if captured separately?

d) Would we need to limit the possible points of view when rendering the captured scenes for the VR glasses of the users? Why? Discuss both the foreground and the background cases.

e) Propose 3D data processing tools that may be needed for the fusion of sensors in questions 1+2, and for the separate capture and fusion of foreground with background in question 4. Remember that the foreground sensors may be moving during foreground capture. Name two libraries for point cloud processing that might be useful for this.

f) Should HBO and Netflix be worried about compression schemes for streaming this type of "real virtuality" contents we are talking about? Why? Which tools are available at the moment?

g) How do you think the quality of the rendered scenes would compare to HD/4K video contents or even rendering quality of synthetic scenes in gaming environments?

a) ReVi would need to reconstruct the scene in 3D for later rendering on the VR glasses. Range scanners (Lidars, for outdoors) ease the direct capture of 3D geometries for dynamic (moving) scenes. As Lidars have limited resolution and do not capture color, high-resolution RGB cameras should be added during capture. Video feeds of RGB cameras should be registered with the geometry of pointclouds resulting from the Lidars (e.g. same capture rig).

b) 3D sensors should be placed 360^{o} around the scene trying to capture all viewpoints that the viewer may later choose for display. Any part of the scene not being captured will not be visible or will have to be interpolated (guessed?) from neighboring views. Depending on the level of occlusion of the scene (the more cluttered, the more occlusions) the number of sensors needed will grow, but as Lidars are active sensors, they also have the limitation that they should not interfere with each other (different working frequencies or different modulations for their beam lights may be a solution for this)

c) Separate capture of the background of an empty scene would be useful to simplify the setup of the sensors (looking out for the background, looking in for the foreground). This would work only possible when there is no interaction between foreground (talent) and background. It would be best if the background is static, as then it may be captured at higher resolution with range scanners that do not capture video. In order to fuse foreground and background, the 3D geometries of the two captures should be registered.

d) Probably yes, otherwise the number of sensors should increase exponentially to allow the viewers wander everywhere in a scene without limits! The geometry of a closed background may limit the navigable scene. For the foreground, just avoiding object penetration might do.

e) Background suppression/extraction, registration, geometry and texture interpolation... ICP or even SLAM techniques may help in registering moving sensors and their output streams with the geometry of the scene for 3D reconstruction. PCL library or Open3D.

f) For compression purposes, MPEG Point Cloud Coding should be explored.

g) Most likely, 3D reconstructed and rendered scenes will suffer from low level quality due to insufficient resolution of range sensors, but HD textures will possibly aid to increase the perceived quality of the renderings. The fact that scenes are real, could add a plus for the viewers... hopefully like experiencing Google street view with a larger quality.

Problem 7 *0.90 Points*

We wish to use deep learning to process point-cloud data.

a) Would you consider a good idea using a neural network consisting only on fully-connected layers? (Yes / No / Explain why)

b) Explain what is the strategy followed by PoinNet to address the difficulty to get a meaningful ordering of the input points.

c) Another option is to have the point cloud represetned as a graph and use Graph Convolutional Networks (GCNs) which, as their name indicate, allow constructing convolutional layers that can be applied to graphs. Given two function $\mathbf{f}$ and $\mathbf{g}$ defined over the vertices of a graph, explain in detail the *trick* used by GCNs to compute the convolution $\mathbf{f} * \mathbf{g}$ over the graph (this question refers to the work by Bruna et al.).

a) No, because 1) the network would have a huge number of paramenters and 2) there would be no parameter sharing, thus making it hard for the network to learn meaningul shift-invariant features.

b) PointNet uses funcations that are invariant to permutations of the input points. This is achived by using symmetric functions, i.e. functions that return the same output regardless of the ordering of their inputs, such as the summation or the maximum. Specifically, the PointNet architecture consists of a few alignment layers that process each point separately followed by a max() function that combines them. After this point, further layers can be added in a standard manner since the representation is already permutation invariant.

c) GCNs compute the convolution over the graph by transforming the signals to the spectral domain, in which the not-well defined convolution becomes a product, which is indeed well defined. Mapping into the spectral domain is accomplished by using the eigendecomposition of the Laplacian operator, which we can generically define as $\mathbf{\Delta}$ (different definitions of this operator are possible). Eigendecomposition yields $\mathbf{\Delta} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T$, where $\mathbf{P}$ is the eigenvector matrix and $\mathbf{\Delta}$ is the diagonal matrix of eigenvalues. We can map signals from the graph fomain to the spectral domain by pre-multiplying them with $\mathbf{P}^T$, while the opposite mapping (from the spectral to the graph domain) is achieved by pre-multiplying with $\mathbf{P}$. Thus:

$$\mathbf{f} * \mathbf{g} = \mathbf{P}\big((\mathbf{P}^T\mathbf{f}) \circ (\mathbf{P}^T\mathbf{g})\big)$$

where $\circ$ is the element-wise product.