+1/1/60+

EPFL

1

Teacher : Prof. Pascal Fua
CS-442 Computer Vision
26/06/2026
Duration : 90 minutes

Student 1

SCIPER : **999000**

Do not turn the page before the start of the exam. This document is double-sided, has 16 pages, the last ones possibly blank. Do not unstaple.

- Place your student card on your table.
- **A one page two-sided hand-written cheat-sheet** is allowed to be used during the exam.
- Using a **calculator** or any electronic device is not permitted during the exam.
- All questions have one or more correct answers.
- The grading scheme is such that random answering is discouraged:
  - Each answer of a multiple choice question is awarded $+1$ point if correct and -1 point if incorrect. If the **whole** question is left unanswered no points (positive nor negative) are awarded. Note that "correct" means that a true answer should be ticked and that a false one should be left unticked.

	Correct answers:	Student's answers:	Grading:
a)	■	■	+1
b)	☐	■	-1
c)	■	☐	-1
d)	☐	☐	+1

  - The scores for separate questions are **not clipped to** 0, that is, you can get negative score for a question.
- Use a **black or dark blue ballpen** and clearly erase with **correction fluid** if necessary.
- If a question is wrong, the teacher may decide to nullify it.

Respectez les consignes suivantes | Observe this guidelines | Beachten Sie bitte die unten stehenden Richtlinien

choisir une réponse | select an answer Antwort auswählen	ne PAS choisir une réponse | NOT select an answer NICHT Antwort auswählen	Corriger une réponse | Correct an answer Antwort korrigieren
☒ ☑ ■	☐	☐

ce qu'il ne faut **PAS** faire | what should **NOT** be done | was man **NICHT** tun sollte

For your examination, preferably print documents compiled from auto-multiple-choice.

First part: Multiple choice questions

For each question, mark the box corresponding to the correct answer. Each question has **at least one** correct answer.

Question 1 Regarding Sobel and Prewitt operators, which statement(s) is (are) correct?

- [x] Both estimate image derivatives using convolution masks.
- [x] Sobel introduces additional smoothing compared to Prewitt.
- [ ] Applying Sobel or Prewitt to an $N \times N$ image necessarily requires a total of $9N^2$ multiplications and additions.
- [x] Once the horizontal and vertical derivatives have been computed, the directional derivative along any orientation can be expressed as a linear combination of them.

Question 2 The function below implements a 2D convolution between an input image I and a filter F.

```
def applyImageFilter(I, F, padding='same'):
    N_, M_ = F.shape

    N = np.int64((N_ - 1) / 2)
    M = np.int64((M_ - 1) / 2)

    H, W = I.shape

    if padding == 'same':
        I = np.pad(I, ((N, N), (M, M)), mode='constant')
        R = np.zeros((H, W), dtype=np.float64)

    elif padding == 'valid':
        R = np.zeros((H - 2*N, W - 2*M), dtype=np.float64)

    H_R, W_R = R.shape

    for x in range(H_R):
        for y in range(W_R):
            for i in range(-N, N+1):
                for j in range(-M, M+1):
                    R[x, y] += I[x-i+N, y-j+M] * F[i+N, j+M]

    return R
```

`np.pad` adds extra values around an array's borders, with the amount of padding and filling method specified by its arguments. Which statement(s) is (are) correct?

- [ ] The function always produces an output image with the same dimensions as the input image, independently of the selected padding mode.
- [ ] Passing a 4×4 filter to the function would still allow the convolution to be computed correctly.
- [x] The terms $x - i$ and $y - j$ in $R[x, y] \mathrel{+}= I[x - i + N, y - j + M] \cdot F[i + N, j + M]$ are used because convolution requires the filter kernel to be spatially flipped before being applied to the image.
- [x] The offsets $+N$ and $+M$ are used to convert filter coordinates from the mathematical ranges $[-N, N]$ and $[-M, M]$ into valid non-negative Python array indices.

Question 3 Regarding noise in images, which of the following statement(s) is (are) correct?

- ☐ Applying differentiation in the spatial domain corresponds to dividing the Fourier transform by the frequency variable.
- ■ In the Fourier domain, differentiation amplifies high-frequency components because it multiplies the signal spectrum by the frequency variable.
- ☐ Performing differentiation after Gaussian smoothing is mathematically equivalent to smoothing the differentiated image only when the Gaussian kernel is separable.
- ■ Smoothing and differentiation can be combined by convolving the image directly with the derivative of a Gaussian filter.

Question 4 Which of the following statement(s) is (are) correct about classical and learning-based edge detection methods?

- ☐ Convolution-based edge detectors are inherently insensitive to smooth illumination variations and therefore only respond to true object boundaries.
- ☐ Learning-based edge detectors can typically achieve high performance without requiring large annotated training datasets.
- ☐ Gradient-based edge detectors admit a single universal choice of thresholds and scale parameters that performs optimally across all images and imaging conditions.
- ■ Learning-based edge detection methods typically estimate the probability that a pixel belongs to an image boundary using features learned from training data.

Question 5 Which of the following statement(s) is (are) correct about the Canny edge detector?

- ■ Non-maximum suppression thins the detected edges by preserving only local maxima of the gradient magnitude along the gradient direction.
- ■ Choosing hysteresis thresholds that are too low may significantly increase the number of noise-induced edge detections in the final edge map.
- ■ Increasing the Gaussian smoothing parameter σ generally improves robustness to noise, but may decrease the precision of edge localization.

Question 6 Which statement(s) apply to the Live Wire method?

- ■ The cost of Dynamic programming methods grow rapidly with the dimension of the state space.
- ☐ Low gradient magnitudes correspond to lower local edge costs in the graph.
- ☐ The computational cost scales exponentially with the total number of pixels.
- ■ It uses Dijkstras algorithm for optimal path search.

Question 7 Which statement(s) regarding the Generalized Hough Transform is (are) correct?

- ☐ The algorithm scales efficiently in terms of computational cost for shapes defined by a high number of parameters.
- ■ The numerical value in the accumulator corresponds to the number of edge pixels voting for that shape location hypothesis.
- ☐ It requires continuous and unbroken edge contours to successfully detect a shape.

Question 8 Consider the construction phase of an R-table for the Generalized Hough Transform. Suppose the chosen reference center of a shape is at $X_c = (0, 2)$. A boundary pixel is detected at position $X = (0, 5)$ with a local gradient orientation of $\phi = 0$. The R-table stores the displacement vector (pointing from the boundary pixel to the reference center) in polar coordinates (r, α). What are the values of the parameters r and α for this displacement vector?

- ☐ The parameters are $r = 5$ and $\alpha = -\frac{\pi}{2}$.
- ■ The parameters are $r = 3$ and $\alpha = -\frac{\pi}{2}$.
- ☐ The parameters are $r = 5$ and $\alpha = \frac{\pi}{2}$.
- ☐ The parameters are $r = 3$ and $\alpha = \frac{\pi}{2}$.

Question 9 Which statement(s) regarding the use of a standard "Vanilla U-Net" for delineation tasks is (are) correct?

- ■ It requires a labeled dataset for training purposes.
- ■ It frames delineation as a pixel-wise classification problem.
- ■ It is prone to topological errors, such as producing disconnected segments.
- ■ Its architecture relies on convolutional layers to extract visual features from the input image.

Question 10 For binary segmentation purposes, let I be a grayscale image, and let $x_i \in \{0, 1\}$ be the label that can be assigned to pixel i (0 = background, 1 = foreground). We model the image as a graph $G(V, E)$, where V is a set of vertices, each vertex representing a pixel; and E is the set of edges, each edge being the connection between neighboring pixels. The weight $w_{i,j}$ reflects the similarity between pixels i and j. Let $\psi_i(x_i)$ be the cost of assigning label x_i to pixel i. For segmentation purposes, which of the following objective function(s) make sense for the s-t min-cut to minimize.

- ☐ $E(x) = \sum_{i \in V} \psi_i(x_i) + \lambda \sum_{(i,j) \in E} w_{i,j} (x_i - x_j)$
- ☐ $E(x) = \sum_{i \in V} \psi_i(x_i) - \lambda \sum_{(i,j) \in E} w_{i,j} \mathbf{1}(x_i \neq x_j)$
- ■ $E(x) = \sum_{i \in V} \psi_i(x_i) + \lambda \sum_{(i,j) \in E} w_{i,j} \mathbf{1}(x_i \neq x_j)$
- ☐ $E(x) = \sum_{i \in V} x_i + \lambda \sum_{(i,j) \in E} w_{i,j} (x_i - x_j)^2$

Question 11 Which of the following is (are) true about the Lambertian shading model?

- ☐ The Lambertian model naturally explains strong specular highlights on shiny objects.
- ☐ For an ideal Lambertian surface, the observed radiance depends mainly on the viewing direction.
- ■ Under distant illumination and constant albedo, image intensity can be interpreted through a reflectance map that constrains the surface normal.
- ■ For an ideal Lambertian surface, the observed radiance depends on the angle between the surface normal and the light direction.

Question 12 Which of the following is (are) true about recovering 3D shape from shading?

- ☐ A single shaded image always determines a unique 3D surface if the object is Lambertian.
- ■ Shading provides constraints on surface normals but does not directly give depth values.
- ☐ Boundary conditions are irrelevant once the light source direction is known.
- ■ Recovering a surface from normals amounts to solving a differential problem, for which boundary conditions are important.

Question 13 In this question, we implement a simple k-means clustering algorithm to segment a noisy depth image `depthImage`. Each pixel in the depth image contains an intensity value proportional, up to a scale factor, to the inverse distance between the camera and scene objects. Because the depth image is noisy, clustering is preferred over simple thresholding.
The algorithm iteratively assigns pixels to the nearest centroid and updates the centroid values until convergence or until a maximum number of iterations `max_iters` is reached.
A key part of the algorithm is the stopping criterion, which checks whether the cluster centroids have sufficiently changed between two iterations.
Consider the following incomplete code snippet:

```
# Initialization of 3 centroids
mu = np.array([0, 500, 1000])
old_mu = mu + 100
distances = np.zeros((3, H, W))
epsilon = 0
k = 3

while (count_iter < max_iters and ________):
    for k_id in range(k):
        distances[k_id] = np.abs(depthImage - mu[k_id])

    cluster_assignments = np.argmin(distances, axis=0)

    old_mu = np.copy(mu)
    for k_id in range(k):
        mu[k_id] = np.mean(depthImage[cluster_assignments == k_id])

    count_iter += 1
```

Explanation of selected NumPy functions:

- `np.argmin`: returns the index of the minimum value in an array.
- `np.abs`: returns the element-wise absolute value.
- `np.all`: returns `True` if all elements are `True` or nonzero along an axis; otherwise returns `False`.
- `np.any`: returns `True` if at least one element is `True` or nonzero; otherwise returns `False`.

Which of the following expressions correctly completes the stopping criterion so that the algorithm stops when the centroids have converged?

- ☐ `np.any(np.abs(old_mu - mu) <= epsilon)`
- ■ `np.any(np.abs(old_mu - mu) > epsilon)`
- ☐ `np.all(np.abs(old_mu - mu) <= epsilon)`
- ☐ `np.all(np.abs(old_mu - mu) > epsilon)`

Question 14 Which of the following is (are) true about shape from texture under classical geometric assumptions?

- ☐ It assumes that texture deformation is caused only by changes in object color.
- ☐ It can recover surface shape from any arbitrary image.
- ■ It assumes that texture elements reside on the surface and have no thickness.
- ■ Changes in the apparent size and shape of repeated texture elements can provide cues about surface orientation.

Question 15 You are given the following image of a grape leaf with approximately uniform green albedo under a directional light source. To recover its 3D shape, which of the below method(s) is (are) applicable?

- [ ] Shape-from-contours, because the outline of the leaf and its visible internal edges are sufficient to reconstruct the detailed folds and local curvature of the surface.
- [ ] Shape-from-stereo, because neighboring regions of the same leaf can be treated as different viewpoints for triangulating the 3D surface.
- [x] Shape-from-shading, because the leaf surface has approximately uniform albedo, and the observed intensity variations are mainly caused by changes in surface orientation with respect to the light direction.
- [ ] Shape-from-texture, because the fine vein patterns on the leaf can be treated as a regular repeated texture whose deformation directly reveals the surface orientation.

Question 16 In the context of stereo, select the correct options:

- [ ] A two-camera setup can have more than two epipoles.
- [x] Epipolar lines are lines on which the corresponding point from another perspective must lie.
- [ ] Epipolar lines always converge at a scene vanishing point.
- [x] Epipolar lines may have different orientations depending on the camera geometry.

Question 17 What is (are) the consequence(s) of image rectification in regular stereo vision?

- [ ] It makes all objects have the same depth.
- [x] The correspondence search for any given point can be conducted on the same image row.
- [ ] It removes the need for finding corresponding points.
- [x] It transforms the images so that epipolar lines become parallel.

Question 18 Which statement(s) about depth discontinuity contours and occluding contours is (are) correct?

- [ ] Both have well-defined 3D locations independent of viewpoint.
- [ ] Neither has well-defined 3D locations.
- [ ] Occluding contours have well-defined 3D locations, while depth discontinuity contours depend on viewpoint.
- [x] Depth discontinuity contours have well-defined 3D locations, while occluding contours depend on viewpoint.

Question 19 Which of the following statement(s) about the color consistency assumptions used to estimate the motion field is (are) true?

- ☐ Temporal consistency assumes that surface patches move at a constant speed.
- ☐ The motion field should be constant across all pixels.
- ☐ Brightness consistency requires globally uniform and evenly distributed illumination.
- ■ Surfaces are not always Lambertian, which can invalidate the color consistency assumption.

Question 20 When performing structure from motion,

- ☐ The accuracy of the camera model does not affect the accuracy of the reconstructed shape.
- ■ To reduce drift, the system should recognize previously seen scenes and perform global optimization.
- ☐ Thanks to bundle adjustment and global optimization, structure from motion does not require a textured environment.
- ☐ By minimizing the error between measured and projected image points, bundle adjustment optimizes absolute camera poses, surface color, and the illumination model of nearby frames.

Question 21 In Exercise Session 7, *Simple Stereo Visual Odometry*, we implemented several designs. Which of the following statement(s) is (are) correct?

- ☐ If two descriptors are very similar, then the corresponding stereo match is correct.
- ■ This visual odometry method assumes that feature points remain consistent under camera motion.
- ☐ Given only 2D-2D correspondences between consecutive images, PnP can be used to estimate the camera poses.
- ■ Matches with small disparity should be rejected because they yield large depth values and are sensitive to small errors.

Second part, open questions

Answer in the empty space below each question. Your answer should be carefully justified, and all the steps of your argument should be discussed in details.
Leave the check-boxes empty, they are used for grading.

Problem statement

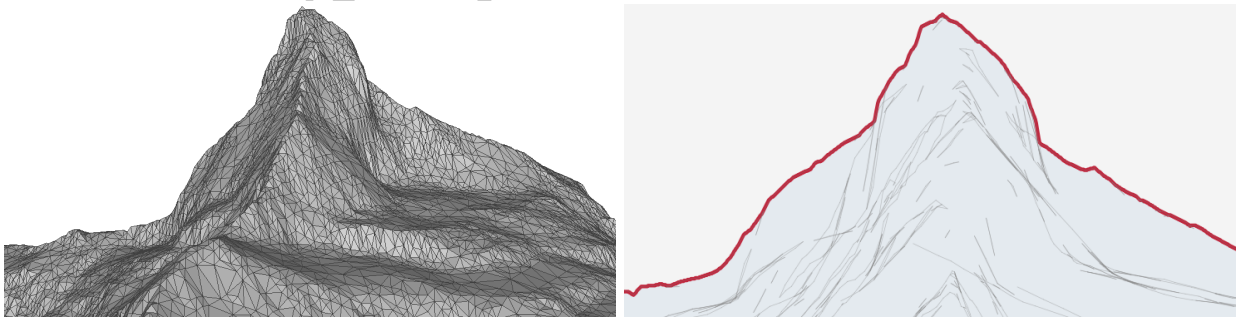
In this problem, we will design a computer-vision solution to a uniquely Swiss problem: identifying whether a given mountain on the horizon is the Matterhorn or not. To this end, the user will hold their phone in landscape mode and approximately level with the horizon. If the Matterhorn is found, its outline will be highlighted and a message will appear. If not, nothing will happen. In particular, we do not attempt to identify other mountains.

To achieve this, we will generate two outlines: the mesh outline on one end, computed from the known geometry of the mountain and the known viewpoint, and the image outline on the other end. Intuitively, if these two outlines are similar, then we have found Matterhorn.

Deriving an outline from the 3D mesh

We will use a **3D surface mesh** of the mountain, similar to the surfaces seen in class during this semester. Specifically, a triangle mesh is denoted $(\boldsymbol{V}, \boldsymbol{F})$, where $\boldsymbol{V} \in \mathbb{R}^{\boldsymbol{N_v} \times \boldsymbol{3}}$ is a matrix of vertex positions and the $\boldsymbol{i}$-th row is the 3D coordinates of the $\boldsymbol{i}$-th vertex. $\boldsymbol{F} \in \mathbb{N}^{\boldsymbol{N_f} \times \boldsymbol{3}}$ is a list of triangles, such that the $\boldsymbol{i}$-th row contains the indices of 3 vertices in $\boldsymbol{V}$ that are connected as a triangle. In the following, it may also be useful to consider $\boldsymbol{E} \in \mathbb{N}^{\boldsymbol{N_e} \times \boldsymbol{2}}$, the set of edges in the mesh which can be inferred from $\boldsymbol{F}$.

Our goal is now to compute a **mesh outline**, which is a 2D line in image space representing the horizon as seen from $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. This mesh outline is an estimation of what the mesh $(\boldsymbol{V}, \boldsymbol{F})$ would look like as seen by a camera with projection matrix $\boldsymbol{P}$. An example outline generated from the 3D mesh is visualized below.



To obtain such a mesh, a helicopter circles the Matterhorn and takes 50 photographs from known positions and orientations. We initially considered applying **shape from stereo** to these images, but the lack of texture of the snow-covered peak made this task challenging. Instead, we manually segment a binary foreground mask of the visible Matterhorn in each image and would like to use these segmentations to perform 3D reconstruction.

Question 22: *This question is worth 2 points.*

☐0 ☐1 ■2

Which algorithm from the course can use this **segmented data** to reconstruct a 3D shape (1pt)? Explain it in two sentences (1pt).

Answer: We use shape from silhouette/contour (1pt), in which each silhouette back-projects to a cone in 3D from the corresponding camera (1pt). The mesh is recovered as the intersection of all such cones.

Question 23: *This question is worth 2 points.*

☐0 ☐1 ■2

What is the usual limitation of this method (1pt)? Does this impact the horizon outline (1pt)? The horizon outline is the border between the sky and the mountains in the image.

Answer: Shape from silhouette reconstructs the visual hull, the largest volume consistent with all silhouettes. Adding more views can improve the visual hull, but it cannot recover concavities that are not visible in the silhouette (1pt). Thankfully, this does not matter in our problem

setting since the outline does not depend on interior concavities but only on the silhouette (1pt).

Let us now assume that we have created a satisfactory mesh (V, F). Using GPS localization and user altitude, we get the Matterhorn position (X, Y, Z) and user position (x_u, y_u, z_u). From these coordinates we derive a projection matrix P of the picture taken by the user looking directly towards the mountain. P is computed from the camera intrinsics K and the look-at direction $(X - x_u, Y - y_u, Z - z_u)$. For a mesh vertex $\tilde{v} = (v_x, v_y, v_z, 1)^T$, the camera projection gives $P\tilde{v} = (u, v, w)^T$, and the pixel coordinates are $(i, j) = (u/w, v/w)$. In other words, P can be used to project 3D positions to 2D pixel coordinates.

Question 24: *This question is worth 2 points.*

0 1 2

Using the mesh edges E, design an algorithm that computes the outline of Matterhorn as could theoretically be seen by the user (2pts). Use expressions from the previous paragraph.

Answer: Project all the edges in E from 3D to 2D with P (1pt). Mark the corresponding pixels red. Keep only the edges at the top in image space (1pt): for each column in the image, keep only the top red pixel and unmark all pixels under it.
Alternative answer: project only edges where one adjacent triangle is front-facing and the other is back-facing (i.e. sign changes in the dot product between the face normal and the view direction).

Image outline

In this part, we compute an outline from the user's photo, representing the horizon in the image. Some examples are displayed below, with the output outline in blue.

To detect edges, we use a simplified version of the Canny edge detector. As seen in the lectures and labs, the pipeline has 4 steps which we will implement as follows:

```
def image_to_outline(img, sigma, ksize, tau):
    smoothed = gaussian_smoothing(img, sigma, ksize)
    M, theta = sobel_gradients(smoothed)
    M_thin   = non_max_suppression(M, theta)
    e        = M_thin > tau
    return edges_to_outline(e)
```

Let us assume an array `arr` representing an image of shape $H \times W$. In your implementation, you can freely use any of the following functions:

Helper functions from the lab

`applyImageFilter(arr, F)`	Convolves image `arr` with filter `F`, returning an array of the same shape as `arr`.
`gaussian_filter(fSize, fSigma)`	Returns a two-dimensional Gaussian kernel of size `fSize`×`fSize` with standard deviation `fSigma`, normalised to sum to 1.

`numpy` functions

`np.sqrt(arr)`	Element-wise square root.
`np.any(arr, axis=0)`	Reduces along rows, returning a one-dimensional boolean array of length W. The value at column c is `True` when that column contains at least one `True`.
`np.argmax(arr, axis=0)`	Reduces along rows, returning, for each column, the row index of the first maximum. Applied to a boolean array, this gives the row index of the first `True`.
`np.where(cond, a, b)`	Element-wise conditional: returns `a` where `cond` is `True`, and `b` otherwise.

Instructions:

- Write your answer **in the dedicated space**, not in the question itself, and do not include the lines which do not change (e.g. `def ...`).
- Exact syntax is not required and deviations won't be penalized as long as the operation is correct.
- For full credit, do not use explicit Python loops, which are poorly parallelized.

Question 25: *This question is worth 2 points.*

Fill the following function to implement Step 1:

```
def gaussian_smoothing(img, sigma, ksize):
    """
    img   : numpy array of shape (H, W)
    sigma : standard deviation of the Gaussian
    ksize : kernel side length
    Returns the smoothed image.
    """
    F = ...
    return ...
```

Answer:

```
F = gaussian_filter(ksize, sigma) (1pt)
return applyImageFilter(img, F) (1pt)
```

Question 26: *This question is worth 5 points.*

Implement Step 2. The Sobel filters can be obtained by combining a one-dimensional derivative filter and a one-dimensional smoothing filter: $\boldsymbol{d} = \begin{bmatrix}-1 & 0 & 1\end{bmatrix}$, $\boldsymbol{s} = \begin{bmatrix}1 & 2 & 1\end{bmatrix}$. The x-derivative kernel differentiates horizontally and smooths vertically, while the y-derivative kernel differentiates vertically and smooths horizontally.

```
def sobel_gradients(img):
    """
    img : np.ndarray of shape (H, W).

    Returns the gradient magnitude M of shape (H, W)
    and gradient angle theta in [0, pi) of shape (H, W).
    """
    Sx = ...
    Sy = ...
    gx = ...
    gy = ...
    M  = ...
    theta = np.arctan2(gy, gx) % np.pi    # given: angle in [0, pi)
    return M, theta
```

Answer:

```
Sx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]) (1pt)
Sy = np.array([[-1, -2, -1], [ 0,  0,  0], [ 1,  2,  1]]) (1pt)
gx = applyImageFilter(img, Sx) (1pt)
gy = applyImageFilter(img, Sy) (1pt)
M  = np.sqrt(gx**2 + gy**2) (1pt)
```

Normalizing S_x and S_y is acceptable but not necessary here.

We assume that the function `non_max_suppression(M, theta)` is provided. After non-maximum suppression, the main pipeline thresholds the thinned magnitude using $e = M_{thin} > \tau$.

Question 27: *This question is worth 6 points.*

0 1 2 3 4 5 6

Implement the helper function `edges_to_outline`. The outline is computed by identifying every pixel column that has at least one edge pixel, then keeping only the topmost edge in each such column. Rows are indexed from top to bottom, so the topmost edge corresponds to the smallest row index.

```
def edges_to_outline(e):
    """
    e : np.ndarray of shape (H, W), boolean (can only take values in {0, 1}).

    Returns
    -------
    outline : np.ndarray of shape (W,), dtype int.
        outline[c] = row of the topmost edge pixel in column c,
        or -1 if column c contains no edge pixel.
    """
    has_edge = ...
    top_edge = ...
    return ...
```


Answer:

```
has_edge = np.any(e, axis=0) (2pt)
top_edge = np.argmax(e, axis=0) (2pt)
return np.where(has_edge, top_edge, -1) (2pt)
```

Outline matching

We now have two one-dimensional outlines:

- the **mesh outline**, computed by projecting the 3D Matterhorn mesh,

- the **image outline**, computed from the user's photo.

Let $m \in \mathbb{Z}^W$ be the mesh outline and $o \in \mathbb{Z}^W$ be the image outline. For both outlines, the i-th value is the row index of the horizon at the i-th column in the corresponding image, or -1 if no outline point is available in that column.

Question 28: *This question is worth 4 points.*

0 1 2 3 ■4

Assume first that the two outlines are already perfectly aligned in the image. Propose a simple distance D between m and o (2pts), and a criterion that determines whether the picture is in fact of Matterhorn using a threshold distance D_t (2pts).

Answer: Be mindful that we compare only columns where both outlines are defined $\Omega = \{c \mid m_c \neq -1 \text{ and } o_c \neq -1\}$, and that if Ω is empty the distance is considered infinite. Then the mean absolute vertical error is (2pts):

$$D(m, o) = \frac{1}{|\Omega|} \sum_{c \in \Omega} |m_c - o_c|.$$

Our criteria will therefore be that the distance is below a threshold $D(m, o) < D_t$ and there are enough columns that are comparable $|\Omega| > \frac{W}{10}$ (2pts).
Note: An L2 distance could work too, but L1 is preferred here because the image outlines tend to have a lot of outliers.

Question 29: *This question is worth 2 points.*

0 1 ■2

In practice, how would you set the value of D_t? Assume you have access to a set of images that do feature Matterhorn, and a set of images that doesn't but has other mountains instead.

Answer: We set D_t to maximize the accuracy over this set of images.

Question 30: *This question is worth 4 points.*

0 1 2 3 4

To make the matching more robust to small errors in camera orientation, we would like to expand our search to also consider small horizontal and vertical shifts of the mesh outline. We consider potential shifts of the outline in image space, where a shift is represented as $(\Delta h, \Delta v)$. How can we use these candidate shifts in our previous method (2pts)? Which candidate shift should be kept (2pts)?

Answer: For each candidate shift $(\Delta h, \Delta v)$, we compare the image outline o with the shifted mesh outline $S(m, \Delta h, \Delta v)$ and compute a matching score (2pts). We then keep the shift with the smallest distance (2pts):

$$D_{\min} = \min_{\Delta h, \Delta v} D(o, S(m, \Delta h, \Delta v))$$

We can do the same with small variations in scale and rotation, too. This is reminiscent of the Hough transform seen in class.

Question 31: *This question is worth 2 points.*

☐0 ☐1 ■2

Give one practical reason why the matching may produce *false negatives* (Matterhorn is on the picture but not detected, 1pt) and one for *false positives* (Matterhorn is not on the picture but detected regardless, 1pt).

Answer: The image outline may be incorrect because of clouds, increasing the outline distance and leading to a false negative(1pt). Another mountain or cloud may have a similar outline to the Matterhorn from a particular viewpoint and produce a false positive, but this is very unlikely (1pt).

Question 32: *This question is worth 4 points.*

☐0 ☐1 ☐2 ☐3 ■4

Now, we drop the assumption that we have access to the user's GPS coordinates. What key input from our previous method do we lose (2pts)? What can we do instead (2pts)?

Answer: We lose the assumption that the **projection matrix P** is known (2pts). A possible solution is to generate a set of candidate mesh outlines, instead of a single one, then match the image outline against all these candidate mesh outlines. If any of them has a low distance, we have found Matterhorn, and also an estimate of the projection matrix P (2pts).