1. Multiple view geometry formulas

We are interested in examining the relationships between some of the following objects:

- a 3D point $P_{\text{world}} \in \mathbb{R}^3$, expressed in the world frame
- the same 3D point $P_{\text{camera}} \in \mathbb{R}^3$, expressed in the camera frame
- the extrinsic parameters $R$ and $t$ of a camera, expressed as a $3 \times 3$ rotation matrix, and translation vector in $\mathbb{R}^3$, respectively
- $K$, the $3 \times 3$ matrix of the camera’s intrinsic parameters
- $M$, the $3 \times 4$ camera projection matrix
- $A$ and $b$, the left-hand $3 \times 3$ and right-hand $3 \times 1$ submatrices of $M$
- $p_{\text{sensor}} \in \mathbb{R}^2$, the projection of the 3D point onto the sensor plane
- the 3D point $\Omega \in \mathbb{R}^3$, the world-frame coordinates of the camera center
- the ray direction $w_{\text{world}}$ of the ray passing through $P_{\text{world}}$ and $\Omega$

Now provide an equation or homogeneous equivalence specifying each of the following:

a. $P_{\text{camera}}$ in terms of $P_{\text{world}}$ and the extrinsics
b. $p_{\text{sensor}}$ in terms of $P_{\text{camera}}$ and the intrinsics
c. $A$ and $b$ in terms of $K$, $R$, and $t$
d. $\Omega$ in terms of $A$ and $b$
e. $w_{\text{world}}$ in terms of $A$ and $p_{\text{sensor}}$

2. Affine perspective

The Swarthmore amphitheater measures approximately 65 meters from the stage to the back row of seats. Imagine that a standard desk chair has been placed on the stage of the amphitheater. Now consider the following two tasks in multiple view geometry:

- reconstructing a 3D model of the stationary chair from a series of pictures taken around the outer circumference of the amphitheater
- reconstructing a 3D model of the amphitheater by rolling the chair around the stage and taking pictures in multiple directions

In which of the two tasks above would it be more appropriate to use the affine (weak) perspective model, and why?
3. Multiple view geometry methods

For each of the methods below, place a check in each column where the information is known, and place a question mark in the column(s) corresponding to information estimated by the method.

<table>
<thead>
<tr>
<th>Method</th>
<th>intrinsics</th>
<th>extrinsics</th>
<th>3D point locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. camera calibration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b. triangulation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c. stereo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d. structure from motion</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. Supervised vs. unsupervised learning

We have a dataset of \(m\) grayscale images of size \(s \times s\), each labeled as face or not face. For each of the following methods applied to the dataset, explain whether it is supervised, unsupervised, or both. Provide a short justification of your answers.

a. train a neural network to directly classify faces vs non-faces

b. train a single-layer perceptron to classify faces vs non-faces using histograms over 500 representative 6×6 subimages identified using \(k\)-means

c. run PCA on the dataset to identify the 25 directions in \(\mathbb{R}^{s^2}\) along which the input images vary the most

5. Asymptotic runtime of classifiers

For the same classification task as the above problem, consider two concrete approaches:

- Train a convolutional neural network to classify face vs. not-face, which operates as follows:
  - Convolve the \(s \times s\) input image with \(k\) kernels (with learned weights) of size \(f \times f\), padding the borders by repeating edge pixels. The result is of size \(s \times s \times k\) (which you can think of as \(s^2\) vectors in \(\mathbb{R}^k\)).
  - Split the image into four quadrants and use max-pooling to reduce the convolution results to size \(4k\) (one vector in \(\mathbb{R}^k\) per image quadrant).
– Use a “traditional” neural network to handle the rest: the $4k$ outputs of max-pooling are reduced to a hidden layer of size $2k$ using the tanh activation function.

– Finally, the hidden layer is reduced to a single output node using the logistic activation function to produce a probability of face or not-face.

• Train a Viola-Jones-style face classifier (albeit without cascade classification):
  – the classification result is the weighted sum of $k$ 1D thresholding classifiers
  – the input to each classifier is a weighted sum of up to four rectangles of pixels in the input image, where the sums are computed via integral images

Now let’s examine the asymptotic (big-O notation) runtime of classifying an image with each approach:

a. What is the runtime of computing the convolutions for the convolutional network? Note you can not assume any performance boosts from separable kernels.

b. How long does the max-pooling step take?

c. How long does it take to map from max-pooling to hidden layer?

d. How long from hidden layer to output?

e. What is the combined runtime of the convolutional network? (Note you can not collapse unlike terms without knowing concrete values of $s$, $k$, and $f$.)

f. For the Viola-Jones-style classifier, how long does it take to build the integral image?

g. Once the integral image has been built, how long does it take to compute the output of the final classifier?

h. What is the combined runtime of the Viola-Jones-style classifier? (Again, note you cannot combine unlike terms without knowing values of $s$ and $k$.)