Agenda

• Project 4
  • discussion

• Project 5
  • out tomorrow
  • Due 12/04 (last class)

• Topics: face recognition
  • color-based skin detection
  • PCA/Eigenfaces

Slides from Steve Seitz, Michael Black
Last 3 classes

- 11/25 Recognition (material for Proj 5)
- 11/27 No class (enjoy holidays!)
- 12/02 Recognition cont’d
- 12/04 Review for final
Project 4

- Half of you were able to complete
- Material will be on final
Homography constraint

Assume we are given a point \( x_1 \) in image 1 and point \( x_2 \) in image 2 in correspondence.
Assume they are represented in homogenous coordinates.

\[
\begin{align*}
x_1 &= \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \\
x_2 &= \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}
\end{align*}
\]

A homography matrix \( H_{3 \times 3} \) defines the following mapping

\[
\lambda x_2 = H x_1
\]
Write as linear constraint

\[ x_2 \times H x_1 = 0 \]

Using the matrix form of the cross product, we can expand the above into

\[
\begin{bmatrix}
0 & -1 & y_2 \\
1 & 0 & -x_2 \\
-y_2 & x_2 & 0
\end{bmatrix}
\begin{bmatrix}
h_{11} & h_{12} & h_{13} \\
h_{21} & h_{22} & h_{23} \\
h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
y_1 \\
1
\end{bmatrix}
= \begin{bmatrix} 0 \\
0 \\
0 \end{bmatrix}
\]
Simplify

It will be helpful to write $H$ in terms of its rows

$$
\begin{bmatrix}
h_{11} & h_{12} & h_{13} \\
h_{21} & h_{22} & h_{23} \\
h_{31} & h_{32} & h_{33}
\end{bmatrix}
= \begin{bmatrix} h_1^T \\
h_2^T \\
h_3^T \end{bmatrix}
$$

where, for example, $h_1^T = [h_{11} \ h_{12} \ h_{13}]$. We can now write the vectorized form of matrix $H$ as $\mathcal{H}_{9 \times 1} = \begin{bmatrix} h_1 \\
h_2 \\
h_3 \end{bmatrix}$. 

---

Deva Ramanan
November [5. 

Assume we are given $\mathcal{H}_{9 \times 1}$ and want to estimate $H$. To do this, we will use the fact that $H$ can be represented as $H = AH$, where $A$ is a $3 \times 3$ matrix that can be written in terms of its $1 \times 9$ rows $a_1, \ldots, a_3$. Then

$$
\begin{bmatrix}
A_1 \\
A_2 \\
A_3
\end{bmatrix}
= \begin{bmatrix} a_1 \\
a_2 \\
a_3 \end{bmatrix}
$$

By using $AH = 0$, we can simplify $\mathcal{H}_{9 \times 1}$ to $\mathcal{H}_{3 \times 1}$ and eliminate redundant constraints.
Multiply stuff through

\[
\begin{bmatrix}
0 & -1 & y_2 \\
1 & 0 & -x_2 \\
-y_2 & x_2 & 0
\end{bmatrix}
\begin{bmatrix}
h_{11} & h_{12} & h_{13}
h_{21} & h_{22} & h_{23}
h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
y_1 \\
1
\end{bmatrix}
= \begin{bmatrix} 0 \\
0 \\
0 \end{bmatrix}
\]

\[
\begin{bmatrix}
0 & -1 & y_2 \\
1 & 0 & -x_2 \\
-y_2 & x_2 & 0
\end{bmatrix}
\begin{bmatrix}
h_1^T \\
h_2^T \\
h_3^T
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
= 0
\]

\[
\begin{bmatrix}
0 & -x_1^T & y_2x_1^T \\
x_2^T & 0 & -x_2x_1 \\
-y_2x_1 & x_2x_1 & 0
\end{bmatrix}
\begin{bmatrix}
h_1 \\
h_2 \\
h_3
\end{bmatrix}
= 0
\]

\[A_i \mathcal{H} = 0\]
Constraint from multiple correspondences

\[ A\mathcal{H} = 0 \]

\[
A = \begin{bmatrix}
A_1 \\
A_2 \\
\vdots \\
A_n
\end{bmatrix}
\]

0 is a vector of \(3n \times 1\) zeros

Solve for \(H\) with singular value decomposition of \(A\)

\[
[U,S,V] = \text{svd}(A);
H = V(:,\text{end});
H = \text{reshape}(H,3,3)';
\]
Recognition problems

What is it?
• Object and scene recognition

Who is it?
• Identity recognition

Where is it?
• Object detection

What are they doing?
• Activities

All of these are **classification** problems
• Choose one class from a list of possible candidates
Face detection

How to tell if a face is present?
Skin detection

Skin pixels have a distinctive range of colors
  • Corresponds to region(s) in RGB color space

Skin classifier
  • A pixel $X = (R,G,B)$ is skin if it is in the skin (color) region
  • How to find this region?
Skin detection

Learn the skin region from examples
- Manually label skin/non pixels in one or more “training images”
- Plot the training data in RGB space
  - skin pixels shown in orange, non-skin pixels shown in gray
  - some skin pixels may be outside the region, non-skin pixels inside.
Skin classifier

Given \( X = (R,G,B) \): how to determine if it is skin or not?

- Nearest neighbor
  - find labeled pixel closest to \( X \)
- Find plane/curve that separates the two classes
  - popular approach: Support Vector Machines (SVM)
- Data modeling
  - fit a probability density/distribution model to each class
Probability

- X is a random variable
- \( P(X) \) is the probability that \( X \) achieves a certain value

\[
P(X)
\]

called a PDF
- probability distribution/density function
- a 2D PDF is a surface
- a 3D PDF is a volume

\[
0 \leq P(X) \leq 1
\]

\[
\int_{-\infty}^{\infty} P(X) \, dX = 1
\]

\[
\sum P(X) = 1
\]

- continuous \( X \)
- discrete \( X \)
Probabilistic skin classification

Model PDF / uncertainty

- Each pixel has a probability of being skin or not skin
  \[ P(\sim \text{skin}|R) = 1 - P(\text{skin}|R) \]

Skin classifier

- Given \( X = (R,G,B) \): how to determine if it is skin or not?
- Choose interpretation of highest probability

Where do we get \( P(\text{skin}|R) \) and \( P(\sim \text{skin}|R) \)?
Learning conditional PDF’s

\[ P(R|\text{skin}) = \frac{\text{#skin pixels with color } R}{\text{#skin pixels}} \]

We can calculate \( P(R \mid \text{skin}) \) from a set of training images

- It is simply a histogram over the pixels in the training images
  - each bin \( R_i \) contains the proportion of skin pixels with color \( R_i \)
- This doesn’t work as well in higher-dimensional spaces. Why not?

Approach: fit parametric PDF functions

- common choice is rotated Gaussian
  - center \( c = \bar{X} \)
  - covariance \( \sum_X (X - \bar{X})(X - \bar{X})^T \)
Learning conditional PDF’s

\[ P(R|\text{skin}) = \frac{\# \text{skin pixels with color } R}{\# \text{skin pixels}} \]

We can calculate \( P(R \mid \text{skin}) \) from a set of training images.

But this isn’t quite what we want:

- Why not? How to determine if a pixel is skin?
- We want \( P(\text{skin} \mid R) \) not \( P(R \mid \text{skin}) \)
- How can we get it?
## Bayes rule

\[
P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}
\]

- what we measure
- domain knowledge

In terms of our problem:

\[
P(\text{skin}|R) = \frac{P(R|\text{skin})P(\text{skin})}{P(R)}
\]

- what we want
- normalization term

\[
P(R) = P(R|\text{skin})P(\text{skin}) + P(R|\sim \text{skin})P(\sim \text{skin})
\]

What can we use for the prior \(P(\text{skin})\)?

- Domain knowledge:
  - \(P(\text{skin})\) may be larger if we know the image contains a person
  - For a portrait, \(P(\text{skin})\) may be higher for pixels in the center

- Learn the prior from the training set. How?
  - \(P(\text{skin})\) is proportion of skin pixels in training set
Bayesian estimation

- Goal is to choose the label (skin or ~skin) that maximizes the posterior \( \leftrightarrow \) minimizes probability of misclassification
  - this is called **Maximum A Posteriori (MAP) estimation**
Skin detection results

Figure 25.3. The figure shows a variety of images together with the output of the skin detector of Jones and Rehg applied to the image. Pixels marked black are skin pixels, and white are background. Notice that this preserve is relatively effective, and could certainly be used to focus attention on eye, face and hands. Figure from “Statistical color models with application to skin detection,” M.J. Jones and S. Rehg, Proc. Computer Vision and Pattern Recognition, 1999 © 1999, IEEE.
General classification

This same procedure applies in more general circumstances

- More than two classes
- More than one dimension

Example: face detection

- Here, X is an image region
  - dimension = # pixels
  - each face can be thought of as a point in a high dimensional space

Face recognition

Task: given an image of a face, label it with an identity
Running example: expression recognition

Stucturally, no different than face recognition
Images as Points (Feature detection)

Points in $K \times L$ dimensional space

$p_i = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{K \times L} \end{bmatrix}$

$p_j = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{K \times L} \end{bmatrix}$

Matching involves deciding how far apart they are in this space.
Template Matching

\[ p_i \cdot p_j = | p_i | \| p_j | \cos \theta \]

Angle between the vectors

Image patch as a vector

Template or filter as a vector

\[ | p | = \sqrt{p_1^2 + p_1^2 + \ldots + p_n^2} \]
Template Matching

\[ p_i \cdot p_j = \frac{|p_i \parallel p_j| \cos \theta}{|p_i \parallel p_j|} \]

- Correlation (sum of product of “signals”)
- Image patch as a vector
- Template or filter as a vector
- Normalized correlation:

\[ \frac{p_i \cdot p_j}{|p_i \parallel p_j|} = \cos \theta \]

- Angle between vectors
SSD Matching

- An alternative to correlation is to minimize the Sum of Squared Differences (SSD)

\[ E(p_1, p_2) = \sum_{i=1:n} (p_1(i) - p_2(i))^2 \]

- Distance metric.
- Euclidean distance = sqrt(E)
Images as Points

Points in n×m dimensional space

\[ p_i = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n\times m} \end{bmatrix} \]

\[ p_j = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n\times m} \end{bmatrix} \]

Matching involves deciding how far apart they are in this space.
Linear Dimension Reduction

What linear transformations of the images can be used to define a lower-dimensional subspace that captures most of the structure in the image ensemble?
Goal

Data point $n$

$$\tilde{x}^n \in \mathbb{R}^D$$

Low dim representation:

$$\tilde{z}^n \in \mathbb{R}^M \quad M \ll D$$

Map

$$\tilde{x}^n \rightarrow \tilde{z}^n$$
Observation

\[
\bar{x}^n = \sum_{i=1}^{M} a_i \tilde{u}_i + \sum_{j=M+1}^{D} b_j \tilde{u}_j
\]

Approximation $\tilde{x}^n$  
Error

Want the M bases that minimize the mean squared error over the training data

\[
\min E_M = \sum_{n=1}^{N} \left\| \bar{x}^n - \tilde{x}^n \right\|^2
\]
Bases Revisited

\[
\begin{bmatrix}
c_1 \\
c_2 \\
c_3 \\
\vdots \\
c_M
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1
\end{bmatrix}^T
\begin{bmatrix}
p_1 \\
p_2 \\
p_3 \\
\vdots \\
p_{n\times m}
\end{bmatrix}
\]

Projection of the image onto a set of basis vectors.
Simple 2D example

If I give you the mean and one vector to represent the data, what vector would you choose?

Why?
Simple 2D example

\[ \bar{x}^n \approx \bar{x} + a\bar{u} \]
Mouths

\[ X = \begin{bmatrix} \bar{x}^1 & \bar{x}^2 & \ldots & \bar{x}^N \end{bmatrix} = \begin{array}{c} \text{smiling} \end{array} \begin{array}{c} \text{neutral} \end{array} \begin{array}{c} \text{other} \end{array} \]

Recall our goal: \( \bar{x}^n \approx \bar{x} + a\hat{u} \)
Sample Mean

$$\bar{x} = \langle x \rangle = \frac{1}{N} \sum_{i=1}^{N} \bar{x}^i$$

Sample Variance

$$\sigma^2 = \langle (x - \bar{x})^2 \rangle = \text{var}(\bar{x}) = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2$$
Statistics Review

Multiple variables: covariance.

\[
\text{cov}(x, y) = \sigma_{xy} = \langle (x - \bar{x})(y - \bar{y}) \rangle
\]

\[
= \langle xy \rangle - \langle x \rangle \langle y \rangle - \langle y \rangle \langle x \rangle + \langle x \rangle \langle y \rangle
\]

\[
= \langle xy \rangle - \langle x \rangle \langle y \rangle - \langle y \rangle \langle x \rangle + \langle x \rangle \langle y \rangle
\]

\[
= \langle xy \rangle - \langle x \rangle \langle y \rangle
\]

Special case: variance.

\[
\text{cov}(x, x) = \langle x^2 \rangle - \langle x \rangle^2 = \sigma_x^2
\]
Covariance Matrix

For two random variables $x$ and $y$ we have

$$C = \begin{bmatrix}
\sigma_x^2 & \sigma_{xy} \\
\sigma_{yx} & \sigma_y^2
\end{bmatrix}$$

$$C = \frac{1}{N - 1} \sum_{n=1}^{N} (\bar{x}^n - \bar{x})(\bar{x}^n - \bar{x})^T$$
Correlated?

\[ C = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{bmatrix} \]

correlation: strength and direction of a linear relationship between two random variables
Correlated?

\[ C = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{bmatrix} \]
Correlated?

\[ C = \begin{bmatrix}
\sigma_x^2 \\
\sigma_{yx}
\end{bmatrix} \]
Correlation

\[ \rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{E((X - \mu_X)(Y - \mu_Y))}{\sigma_X \sigma_Y}, \]
Outer product

\[
\begin{bmatrix}
  x_1 \\
  x_2 \\
\end{bmatrix}
\begin{bmatrix}
  x_1 & x_2 \\
\end{bmatrix}
= 
\begin{bmatrix}
  x_1 x_1 & x_1 x_2 \\
  x_2 x_1 & x_2 x_2 \\
\end{bmatrix}
\]
Covariance Matrix

\[
X = \begin{bmatrix}
\bar{x}^1 & \bar{x}^2 & \ldots & \bar{x}^N \\
\end{bmatrix} = \\
\begin{bmatrix}
x_1^1 & x_1^2 & \ldots & x_1^N \\
x_2^1 & x_2^2 & \ldots & x_2^N \\
\vdots & \vdots & \ddots & \vdots \\
x_D^1 & x_D^2 & \ldots & x_D^N \\
\end{bmatrix}
\]

Mean

\[
\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x^i
\]
Covariance Matrix

\[ A = X - \bar{x} = \begin{bmatrix}
    x_1^1 - \bar{x}_1 & x_1^2 - \bar{x}_1 & \cdots & x_1^N - \bar{x}_1 \\
    x_2^1 - \bar{x}_2 & x_2^2 - \bar{x}_2 & \cdots & x_2^N - \bar{x}_2 \\
    \vdots & \vdots & \ddots & \vdots \\
    x_D^1 - \bar{x}_D & x_D^2 - \bar{x}_D & \cdots & x_D^N - \bar{x}_D
\end{bmatrix} \]

What is

\[ \frac{1}{N - 1} AA^T \]
Covariance Matrix

\[
AA^T = \begin{bmatrix}
x_1^1 - \bar{x}_1 & x_1^2 - \bar{x}_1 & \cdots & x_1^N - \bar{x}_1 \\
x_2^1 - \bar{x}_2 & x_2^2 - \bar{x}_2 & \cdots & x_2^N - \bar{x}_2 \\
\vdots & \vdots & \ddots & \vdots \\
x_D^1 - \bar{x}_D & x_D^2 - \bar{x}_D & \cdots & x_D^N - \bar{x}_D
\end{bmatrix}
\begin{bmatrix}
x_1^1 - \bar{x}_1 & x_1^1 - \bar{x}_2 & \cdots & x_1^1 - \bar{x}_D \\
x_2^1 - \bar{x}_1 & x_2^1 - \bar{x}_2 & \cdots & x_2^1 - \bar{x}_D \\
\vdots & \vdots & \ddots & \vdots \\
x_D^1 - \bar{x}_1 & x_D^1 - \bar{x}_2 & \cdots & x_D^1 - \bar{x}_D
\end{bmatrix}
\]

\[
AA^T = \begin{bmatrix}
\sum_{j=1}^{N} (x_1^j - \bar{x}_1)^2 & \sum_{j=1}^{N} (x_1^j - \bar{x}_1)(x_2^j - \bar{x}_2) & \cdots \\
\sum_{j=1}^{N} (x_2^j - \bar{x}_1)(x_1^j - \bar{x}_1) & \sum_{j=1}^{N} (x_2^j - \bar{x}_2)^2 & \cdots \\
\vdots & \vdots & \ddots & \vdots \\
\sum_{j=1}^{N} (x_D^j - \bar{x}_1)(x_1^j - \bar{x}_1) & \sum_{j=1}^{N} (x_D^j - \bar{x}_2)^2 & \cdots & \sum_{j=1}^{N} (x_D^j - \bar{x}_D)^2
\end{bmatrix}
\]

Size?
Intuition

\[ \bar{x}^n - \bar{x} = \sum_{i=1}^{M} a_i \bar{u}_i + \sum_{j=M+1}^{D} b_j \bar{u}_j \]

\[ \hat{x}^n = \sum_{i=1}^{M} a_i \bar{u}_i + \bar{x} \]

Projecting onto \( \bar{u}_1 \) captures the majority of the variance and hence projecting onto it minimizes the error.
Intuition

\[ \bar{x}^n - \bar{x} = \sum_{i=1}^{M} a_i \bar{u}_i + \sum_{j=M+1}^{D} b_j \bar{u}_j \]

\[
\min E_M = \sum_{n=1}^{N} \left\| \bar{x}^n - \hat{x}^n \right\|^2
\]

Note that these axes are orthogonal and **decorrelate** the data; ie in the coordinate frame of these axes, the data is uncorrelated (side note: this only works for Gaussians).
Principal Component Analysis

\[ C = U \Lambda U^T = \begin{bmatrix} \vec{e}_1 & \cdots & \vec{e}_D \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_D \end{bmatrix} \begin{bmatrix} \vec{e}_1^T \\ \vdots \\ \vec{e}_D^T \end{bmatrix} \]

First three eigenvectors:
\[ \bar{x}^n - \bar{x} = \sum_{i=1}^{M} a_i \bar{u}_i + \sum_{j=M+1}^{D} b_j \bar{u}_j \]

\[ \min E_M = \sum_{n=1}^{N} \left\| \bar{x}^n - \hat{x}^n \right\|^2 \]

So how do we find these directions of maximum variance? This is key.
Principal Component Analysis

Let \( X = [\bar{x}^1 \cdots \bar{x}^N] \)

Compute the mean column vector: \( \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x^i \)

Subtract the mean from each column.

\[
A = X - \bar{x} = [(\bar{x}^1 - \bar{x}) \cdots (\bar{x}^N - \bar{x})]
\]

Covariance matrix can be written

\[
C = \frac{1}{N-1} AA^T
\]
Principal Component Analysis

$C$ is real, symmetric, positive definite. We can write it

$$C = U \Lambda U^T = \begin{bmatrix} \vec{e}_1 & \cdots & \vec{e}_D \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_D \end{bmatrix} \begin{bmatrix} \vec{e}_1^T \\ \vdots \\ \vec{e}_D^T \end{bmatrix}$$

Orthonormal columns  
Eigenvectors  
eigenvalues
**Principal Component Analysis**

\[
C = U \Lambda U^T = \begin{bmatrix}
\bar{e}_1 & \cdots & \bar{e}_D
\end{bmatrix} \begin{bmatrix}
\lambda_1 \\
\vdots \\
\lambda_D
\end{bmatrix} \begin{bmatrix}
\bar{e}_1^T \\
\vdots \\
\bar{e}_D^T
\end{bmatrix}
\]

**First three eigenvectors:**

![Images of three eigenvectors]
Principal Component Analysis

\[ C = U \Lambda U^T = \begin{bmatrix} \bar{e}_1 & \cdots & \bar{e}_D \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & \lambda_D \end{bmatrix} \begin{bmatrix} \bar{e}_1^T \\ \vdots \\ \bar{e}_D^T \end{bmatrix} \]
Principal Component Analysis

- Eigenvectors are the *principal directions*, and the eigenvalues represent the variance of the data along each principal direction.

  $\lambda_k$ is the marginal variance along the principal direction $\vec{e}_k$.
Principal Component Analysis

• The first principal direction \( \vec{e}_1 \) is the direction along which the variance of the data is maximal, i.e. it maximizes

\[
\vec{e}^T C \vec{e} \quad \text{where} \quad \vec{e}^T \vec{e} = 1
\]

• The second principal direction maximizes the variance of the data in the orthogonal complement of the first eigenvector.

• etc.

Fleet & Szeliski
Principal Component Analysis

- **PCA Approximate Basis:** If \( \lambda_k \approx 0 \) for \( k > M \) for some \( M << D \), then we can approximate the data using only \( M \) of the principal directions (basis vectors):

  - If \( \mathbf{B} = [\bar{e}_1, ..., \bar{e}_M] \), then for all points

    \[
    \bar{x}^n \approx \mathbf{B} \bar{a}^n + \bar{x}
    \]

    where

    \[
    \bar{a}_k^n = (\bar{x}^n - \bar{x})^T \bar{e}_k
    \]
PCA

– Over all rank $M$ bases, $B$ minimizes the MSE of approximation

$$\sum_{j=M+1}^{D} \lambda_j$$

• Choosing subspace dimension $M$:
  – look at decay of the eigenvalues as a function of $M$
  – Larger $M$ means lower expected error in the subspace data approximation
Mouth images

\[ X = \begin{pmatrix}
\text{Images 72x88 pixels.} \\
\text{35 example mouths} \\
\text{A is N columns by 6336 pixels.}
\end{pmatrix}

\text{mean}
Mouth matrix

\[
X = \begin{bmatrix}
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\end{bmatrix}
\]