Basic Transforms

• 3-D Cartesian coordinates system
  \((X,Y,Z)\)

• Translation
  
  \[
  X' = X + X_0 \\
  Y' = Y + Y_0 \\
  Z' = Z + Z_0
  \]
**Matrix Operation**

\[
\begin{bmatrix}
X + X_0 \\
Y + Y_0 \\
Z + Z_0 \\
1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & X_0 \\
0 & 1 & 0 & Y_0 \\
0 & 0 & 1 & Z_0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]

\[
\text{Vector} = \text{Matrix} \times \text{Vector}
\]

\[
v' = Av
\]
Other Transforms

- Scale

\[
\begin{pmatrix}
S_x X \\
S_y Y \\
S_z Z \\
1
\end{pmatrix}
= \begin{pmatrix}
S_x & 0 & 0 & 0 \\
0 & S_y & 0 & 0 \\
0 & 0 & S_z & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
\]
Rotation

\( \alpha \)

\( \beta \)

\( \theta \)
Rotation about X axis

\[ R_\alpha = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha & 0 & 0 \\ 0 & -\sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \]
Rotation about Y axis

\[ R_\beta = \begin{pmatrix}
\cos\beta & 0 & -\sin\beta & 0 \\
0 & 1 & 0 & 0 \\
\sin\beta & 0 & \cos\beta & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix} \]
Rotation about Z axis

$$R_{\theta} = \begin{pmatrix}
\cos \theta & \sin \theta & 0 & 0 \\
-sin\theta & \cos \theta & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}$$
Concatenation

• Several transforms can be expressed as a single concatenated 4x4 transform

\[ v' = R(S(Tv)) \]

\[ v' = A v \]
Rotation

\[ R = R_\alpha R_\beta R_\theta \]
Inverse

- These matrices have inverses

\[
M = \begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}, \quad S^{-1} = \begin{pmatrix}
1/S_x & 0 & 0 & 0 \\
0 & 1/S_y & 0 & 0 \\
0 & 0 & 1/S_z & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}, \quad M' = \begin{pmatrix}
S_x*X \\
S_y*Y \\
S_z*Z \\
1
\end{pmatrix}
\]
Image Formation: Perspective Transform

Lens center

Image plane

\((x,y)\)

\((X,Y,Z)\)

\(f\)

\(z,Z\)

\(m\)
From similar triangles

“Camera” plane               World Space

\[
\frac{x}{f} = \frac{X}{f - Z} \quad \frac{y}{f} = \frac{Y}{f - Z}
\]

\[
\frac{x}{f} = \frac{fX}{f - Z} \quad \frac{y}{f} = \frac{fY}{f - Z}
\]

Note: These eqs are non-linear because the division by the variable Z
Perspective projection by Matrix

\[
\begin{bmatrix}
X \\
Y \\
Z \\
-(Z/f) + 1
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -1/f & 1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]

Convert to Image Plane

\[
\begin{align*}
x &= \frac{fX}{f - Z} \\
y &= \frac{fY}{f - Z} \\
z &= \frac{fZ}{f - Z}
\end{align*}
\]
Projection with H-Coords

\[
\begin{bmatrix}
kX \\
kY \\
kZ \\
-(kZ/f) + k
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & -1/f & 1
\end{bmatrix}
\begin{bmatrix}
\text{m}_h \\
\text{kX} \\
\text{kY} \\
\text{kZ} \\
k
\end{bmatrix}

\text{Homogeneous camera coordinates} \rightarrow \text{Cartesian}

\[
\begin{bmatrix}
\text{f}_kX \\
\text{f}_kY \\
\text{f}_kZ \\
-(kZ/f - k)
\end{bmatrix}
\]
Special Case Projection

\[ f = +\text{inf} \]

\[
\begin{pmatrix}
  kX \\
  kY \\
  kZ \\
  k
\end{pmatrix}
= 
\begin{pmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  kX \\
  kY \\
  kZ \\
  k
\end{pmatrix}
\]
"Orthographic Projection"

\[ f = +\infty \]
Inversing a projection

\[ M_h = P^{-1} m_h \]

\[
P^{-1} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1/f & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

\[
m_h = \begin{pmatrix}
kx_0 \\
k y_0 \\
0 \\
k
\end{pmatrix}
\]

Remember, If we have just an image point.
Inversing a projection

\[ M_h \begin{pmatrix} kx_0 \\ ky_0 \\ 0 \end{pmatrix} = P^{-1} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 1 \\ 0 & 0 & 0 & k \end{pmatrix} \begin{pmatrix} kx_0 \\ ky_0 \\ 0 \end{pmatrix} \]

This is really a direction!
Any point on the line \( p(x_0, y_0, 0) \) and \( (0, 0, f) \)
Inversing a projection

Where the point is along this ray, we don't know!
Very important!

• It is not possible to completely recover a 3D point from its image!

• An image is a projection of 3D points onto a 2D plane ($\mathbb{R}^3 \to \mathbb{R}^2$)

• Computer Vision to the rescue
  - 3D Reconstruction
  - Stereopsis, Shape from Shading, etc . . .
A real camera: Two coordinate systems!
A real camera: Two coordinate systems!

Translation
Rotation
Projection
A real camera: Two coordinate systems!

\[
m_h = \begin{pmatrix} P & R & T & M_h \end{pmatrix}
\]

- **m_h**: Camera Point
- **P**: Projection
- **R**: Rotation
- **T**: Translation
- **M_h**: World Point
Image Operations; Labeling

Image Operations

- **Arithmetic** (p and q are images)
  - p+q
  - p-q
  - p*q (also stated as pq, or p × q)
  - p%q
- **Logic Operations**
  - p AND q
  - p OR q
  - Not q
  - p XOR q
Image Operations

• Subtraction
  - Very useful for determining the difference between two images
  - Used for “detection”

![Image Operations Example](image1.png)

Image Operation

• Addition can blend two images

![Image Operation Example](image2.png)
Logic Operators

• NOT

Logic Operators

• AND
Logic Operators

• OR

Logic Operators

• XOR
Logic Operators

- Can be combined

\[ \text{NOT} \quad \text{AND} \quad (\text{NOT} (A) \text{ AND } B) \]

Logic Operators

- Code example . . .

```java
for(int i=0; i < w*h; i++)
{
    c[i] = a[i] OPERATOR b[i];
}
```

\[ a[i], b[i] = \text{either 0 or non-zero, result } c[i] \text{ is 1 or 0} \]

so you can do this:
\[ c[i] = (a[i] \text{ OPERATOR } b[i]) \times 255; \]
Templates, windows, filters

\[
z = (w_1z_1 + w_2z_2 + w_3z_3 \ldots + w_9z_9) = \sum_{i=1}^{9} w_i z_i
\]

Basic Pixel Relationships

- Neighbors of a pixel
  - \( p = (x, y) \)
  - horizontal and vertical neighbors
    - \((x+1, y)\)
    - \((x-1, y)\)
    - \((x, y+1)\)
    - \((x, y-1)\)
  - These are called the N_4 neighbors of \( p \)
Basic Pixel Relationships

- Neighbors of a pixel
  - \( p = (x, y) \)
  - diagonal neighbors
    - \((x+1, y+1)\)
    - \((x+1, y-1)\)
    - \((x-1, y+1)\)
    - \((x-1, y-1)\)
  - These are called the N_D neighbors of \( p \)

Basic Pixel Relationships

- Neighbors of a pixel
  - \( p = (x,y) \)
  - \( N_8(p) = N_D(p) \cup N_4(p) \)
Connectivity

- Connectivity
  - concept used to establish boundaries of objects
- Two pixels are connected, if
  - their they are adjacent in some spatial sense
  - their gray levels satisfy a specified criterion
- Let V be the set of gray-level values to be connected:
  - V = {1} (binary image)
  - V = {32, 33, … , 64}  // some intensity range

Three types of connectivity

- 4-connectivity
  - Two pixels p and q with values V
  - if q is in the set N_4(p)
- 8-connectivity
  - Two pixels p and q with values V
  - if q is in the set N_8(p)
- m-connectivity
  - Two pixels p and q with values V
  - q is in N_4(p), or q is in N_D(p) and the set N_4(p) ∩ N_4(q) is empty.
Example

4-connectivity

Example

8-connectivity
Distance Measures

- For pixels p, q, and z
  - with coords (x,y), (s,t), and (u,v)

- $D$ is a distance function (or metric) if:
  - $D(p,q) \geq 0$ ( $D(q,p) = 0$ iff $p = q$ )
  - $D(p,q) = D(q,p)$
  - $D(q,z) \leq D(p,q) + D(q,z)$
Distances

- **Euclidean**
  - \( D_e(p,q) = [(x-s)^2 + (y-t)^2]^{(1/2)} \)

- **\( D_4 \) distance** (City block distance)
  - \( D_4(p,q) = |x - s| + |y - t| \)

- **\( D_8 \) distance** (Chessboard distance)
  - \( D_8(p,q) = \max(|x-s|, |y-t|) \)

- \( D_e, D_4, \) and \( D_8 \) depend on spatial coords

M-Connectivity Distance

- \( \sum \) connected “links” from (p, to q)
- Depends on pixel values

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

consider this layout

inf
Labeling of Connected Components

1 1 0 0 0 0 0 0 0
1 1 1 0 0 0 1 1
0 0 1 0 0 0 1 1
0 0 1 1 1 0 1 0
0 0 0 0 1 0 1 1
0 0 1 1 1 1 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0

Labeling of Connected Components

1 1 0 0 0 0 0 0 0
1 1 1 0 0 0 3 3
0 0 1 0 0 0 3 3
0 0 1 0 2 0 3 0
0 0 0 0 2 0 3 3
0 0 2 2 2 0 0 0
0 0 0 2 2 0 0 0
0 0 0 0 0 2 0 0
Labeling of Connected Components

```
1 1 0 0 0 0 0 0
1 1 1 0 0 0 3 3
0 0 1 0 0 0 3 3
0 0 1 0 2 0 3 0
0 0 0 0 2 0 3 3
0 0 2 2 2 0 0 0
0 0 0 0 2 0 0 0
0 0 0 0 4 0 0 0
```

Type of connectivity matters

Labeling Algorithm

- Scan image from left to right; (p) is a pixel at each step
  - if (p) == 0, ignore it
  - if (p) != 0, examine upper and left hand neighbor
    - if (only one neighbor has a label)
      » assign the label to P
    - if (both neighbors have unique labels)
      » assign one of the labels to P and make note of the equivalence
    - else
      » assign a new label to P
- At the end of the scan:
  - resolve equivalences into equivalence classes
  - assign a “unique” label to each class
  - make a 2nd pass through the image and assign the appropriate “equivalence” class to the pixels.
### Example

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Example

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example

\begin{array}{cccccc}
& a & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\end{array}

Example

\begin{array}{cccccc}
& a & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\end{array}
**Example**

<table>
<thead>
<tr>
<th>a</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>b</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Example

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Example

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>b</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Example

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>a</th>
<th>0</th>
<th>b</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>0</th>
<th>0</th>
<th>b</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>c</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>0</th>
<th>0</th>
<th>b</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>c</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>a</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>b</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>c</td>
<td>c</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(c = b)

### Example

<table>
<thead>
<tr>
<th>a</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>b</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>e</td>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(c = b)  
(d = c)  
(e = d)
### Example

<table>
<thead>
<tr>
<th>a</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>b</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>e</td>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(c = b)  
(d = c)  
(e = d)  

\[
\begin{align*}
\{ b \\ c \\ d \\ e \} & \Rightarrow f
\end{align*}
\]

### Example

<table>
<thead>
<tr>
<th>a</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>f</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>f</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(c = b)  
(d = c)  
(e = d)  

\[
\begin{align*}
\{ b \\ c \\ d \\ e \} & \Rightarrow f
\end{align*}
\]
Example

```
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
(c = b)
(d = c)
(e = d)
```

Alternative labeling algorithm

- Scan the image
- if p==1 get new label
  - call function(p_x, p_y, assignLabel)

- function assignLabel( x, y, label)
  - p = label
  - check neighbors
    - for each neighbor N_p == 1 (no label and != 0)
    » call assignLabel( N_x, N_y, currentLabel )
Recursion

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Recursion

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
## Recursion

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

## Recursion

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**Recursion**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Recursion**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>b</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>0</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>b</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Connected Components

- “Blob detection”

Blob Weight

- The number of pixels that make up a blob is often called its weight, or size.
  - Given a connected ID
    - Weight is the sum of all pixels with same ID
Calculating Blob Centroids

- The geometric center of the blob
- $c_x, c_y$

$$C_x = \frac{1}{\text{blob\_weight}} \sum_{x \in [p(x, y) = \text{blob\_id}]} x$$

$$C_y = \frac{1}{\text{blob\_weight}} \sum_{y \in [p(x, y) = \text{blob\_id}]} y$$

Summary

- Simple Image Operations
  - Arithmetic / Logic Operations
  - Intro to templates
- Pixel Relations
  - N_4, N_8, N_D
  - Distance Metrics
- Labeling
  - Connected Components
  - Binary Relation sets
  - 2 algorithms to find connected components