Computer Vision

Dr. Ashish Dutta
Professor
Dept. of Mechanical Engg
IIT Kanpur

Machine vision, computer vision, vision

- Machine vision is concerned with the sensing of vision data and its interpretation by a computer.

- Three main steps for using vision data:
  - Sensing and digitizing image data.
  - Image processing and analysis
  - Applications

Coordinates of points in space?

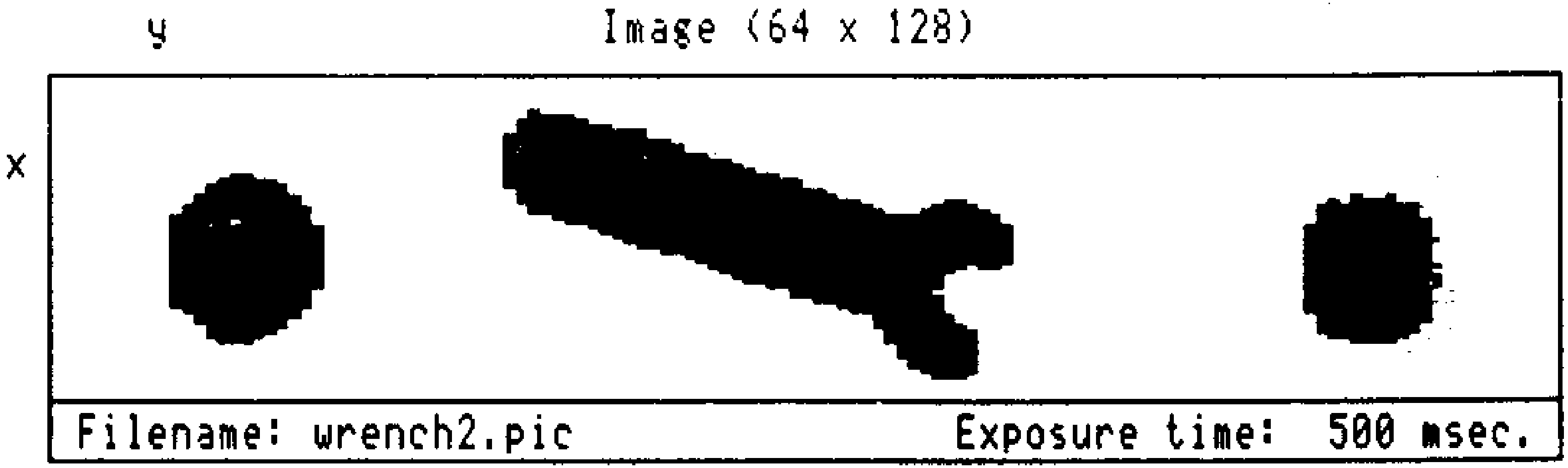
- Robot programs need to contain the coordinates of important points e.g. part location, assembly location, location to place a part.

- Locate the part before picking up.

- Inspection of parts.

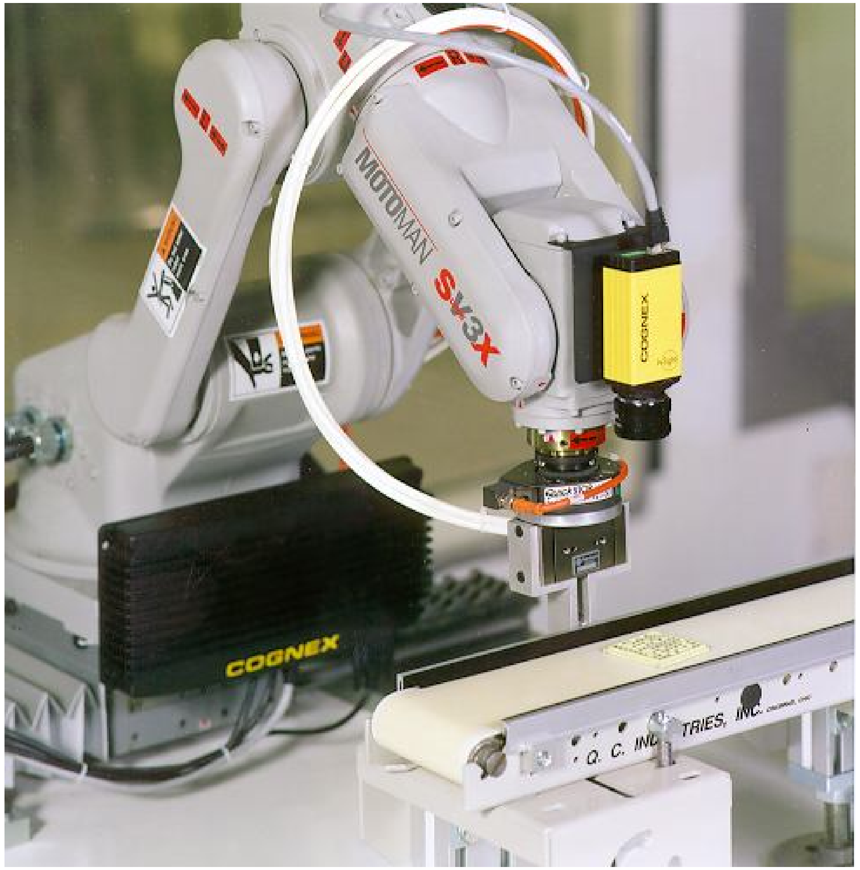
- Assembly

Find the required part ?



Why Use Vision?

- Cost reduction
- To find randomly placed parts
- To verify part type
- To inspect part
- Affordable
- Maintainable



Robots & Vision vs. People

ACTION	MACHINE	PERSON
Speed	Extremely Fast	Medium
Accuracy	High	Varies
Repeatability	High	Varies
Interpretation	Fixed	High

Applications

- Find locations
- Measure parts
- Verify presence
- Inspections
- Robot guidance

Robotic Vision Process Types

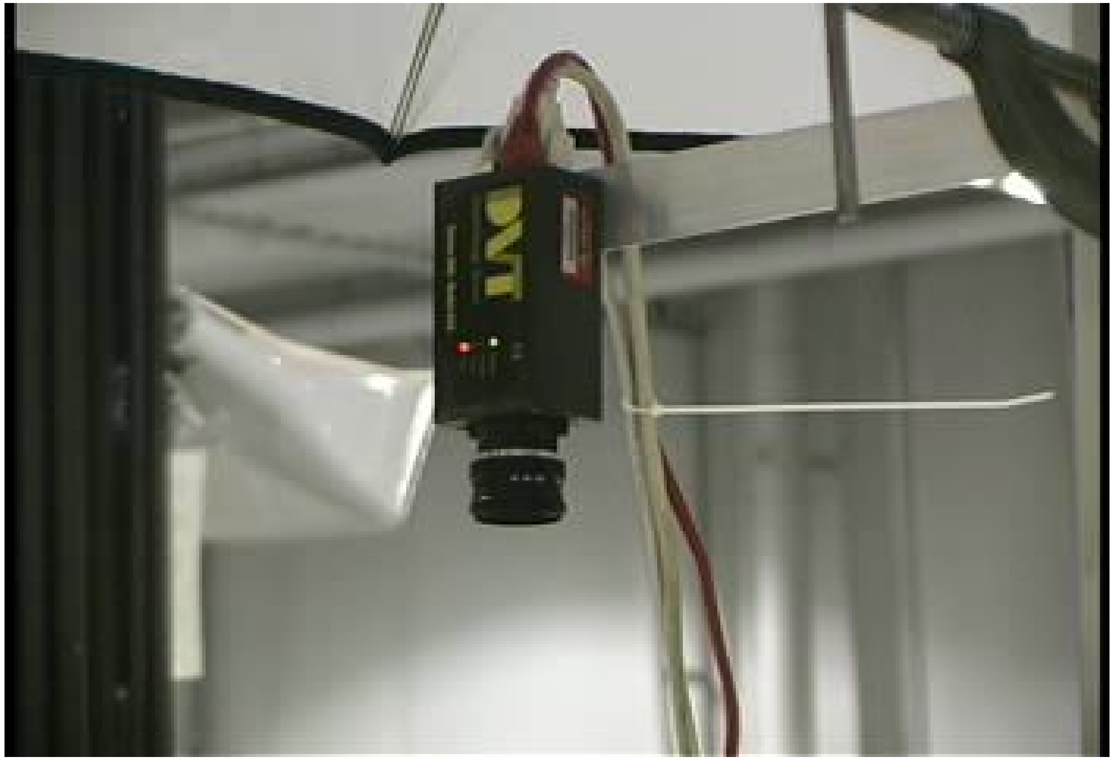
- Fixed camera vision
  - Single camera
  - Multiple cameras
    - High-speed

Robotic Vision Process Types

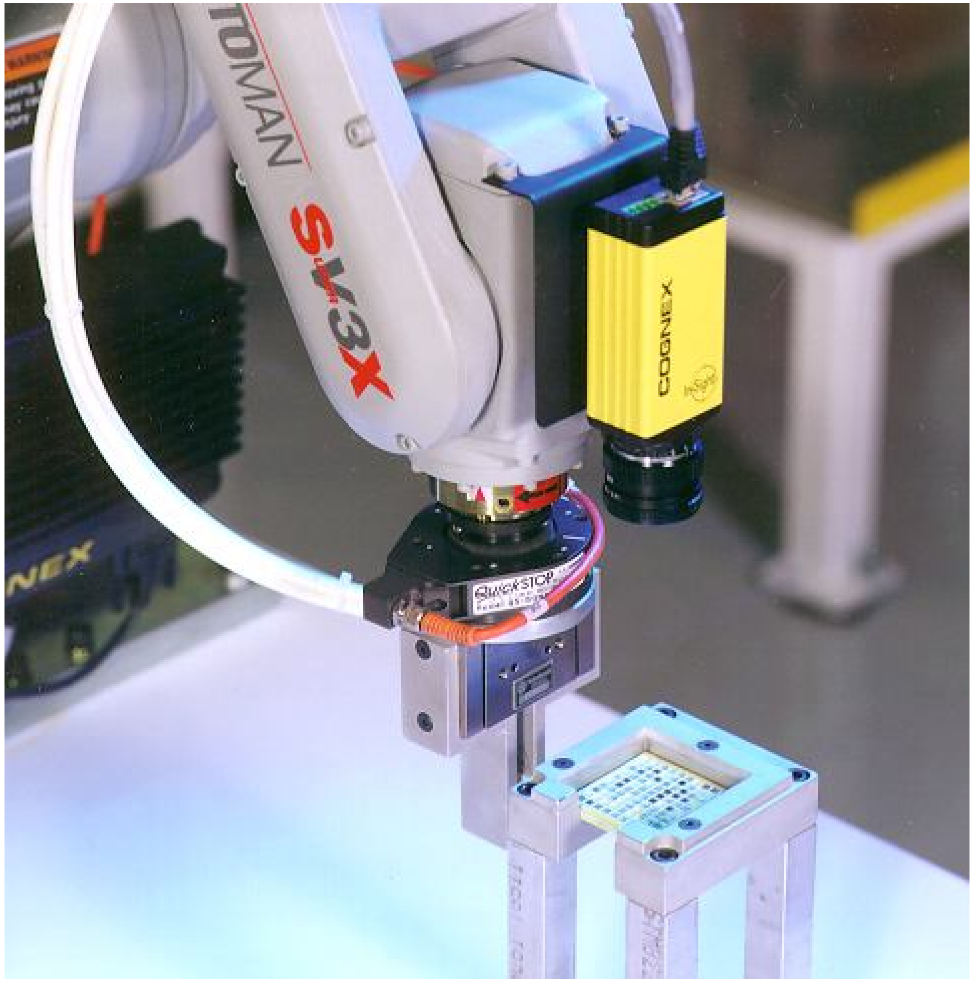
- Robot-mounted vision
  - Single field of view - tight access/multiple parts/clearance
  - Multiple field of view - low cycle time requirements

Flexible Feeding & Conveyor Tracking

- Conveyor-fed parts
- Vision finding
- Part staging for next operation



Part Finding/Inspection

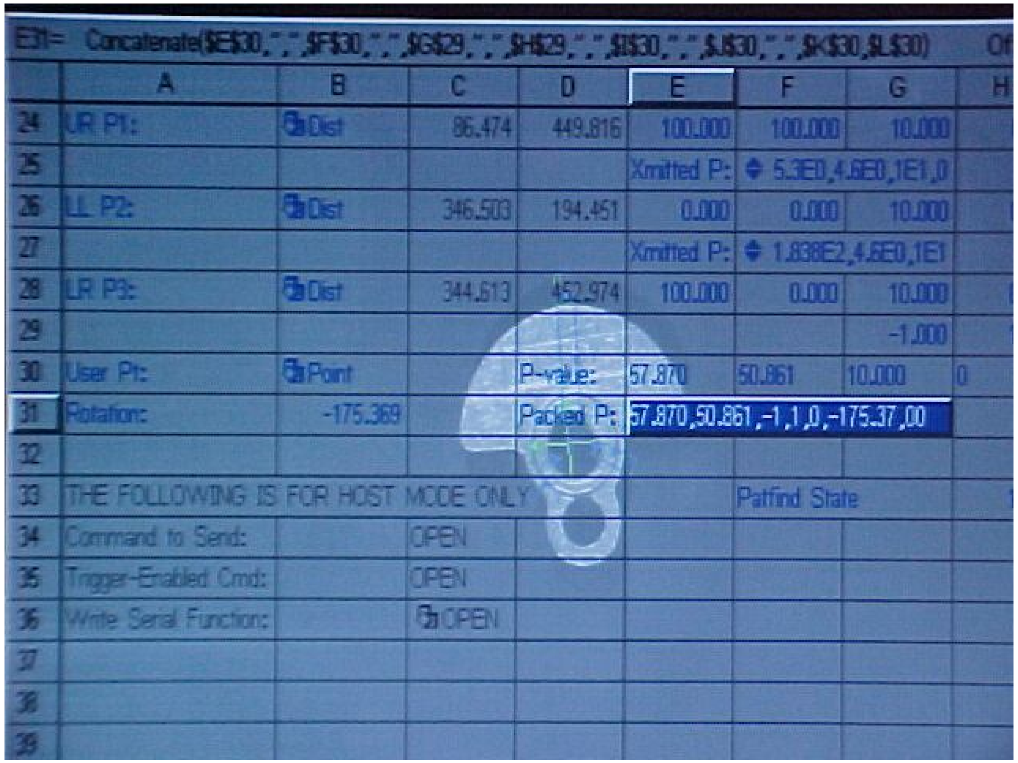


Typical Part

- Task
  - Locate the part and determine:
    - X and Y location
    - Flat on the D hole
  - Place the part on a assembly or shipping tray

Vision Code

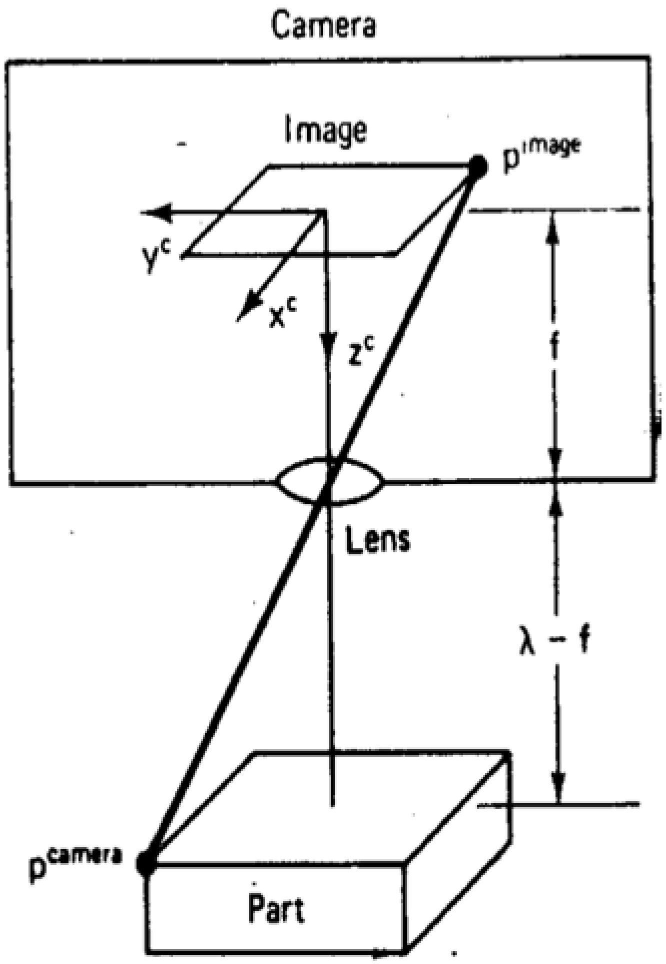
- Tools
  - Cognex Insight
    - Spreadsheet programming
    - Pattern matching
    - Radial rulers
    - Download P-point to controller through RS232 link

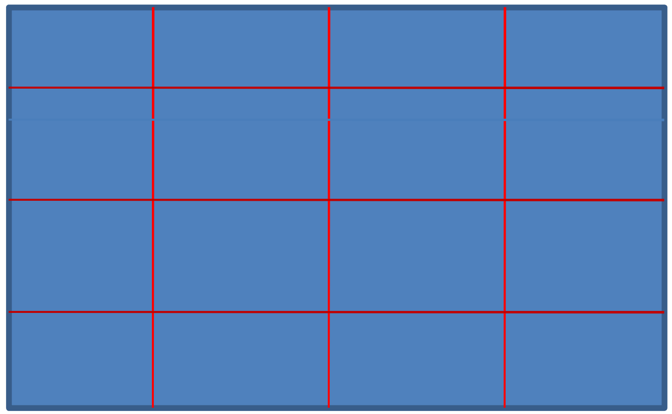


Low-cost Vision-based Quality Inspection

Mithun George Jacob

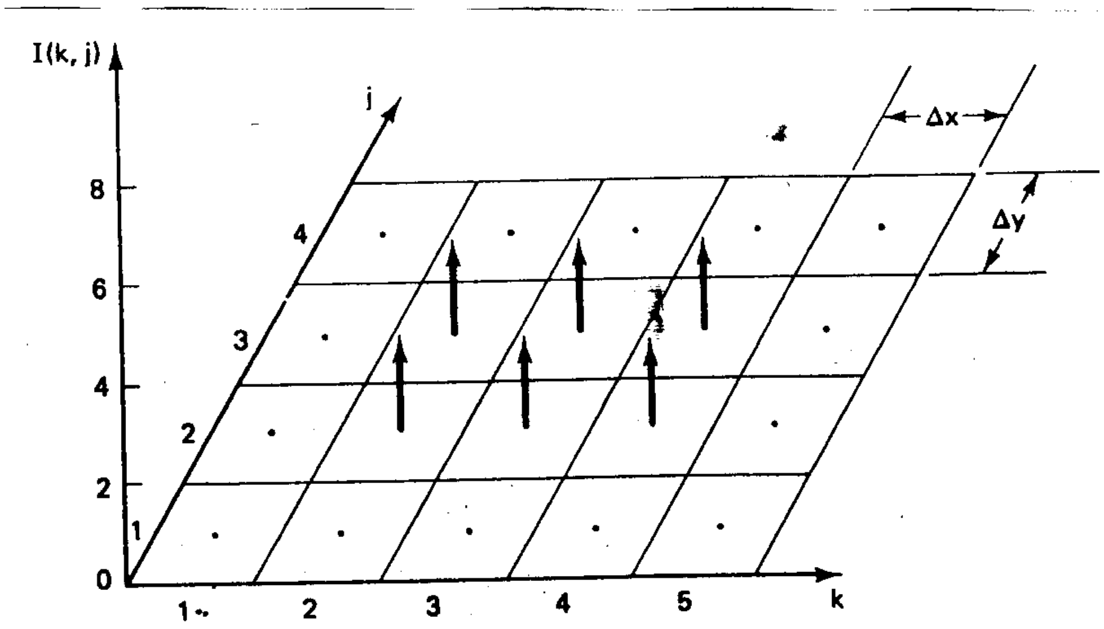
Working of a camera CCD





- Image plane
- Each sensing unit is a Pixel
- Light can be converted to black/white or colour in each pixel before recording

CCD Camera



$$I_a(k,\ j) \triangleq \frac{\int_0^{\Delta x} \int_0^{\Delta y} i[(k-1)\Delta x + x,\ (j-1)\,\Delta y + y]\, dy\, dx}{\Delta x\ \Delta y}$$

Template matching

$$\begin{array}{cc} & j \rightarrow \\ k \downarrow & \begin{bmatrix} 0 & 4 & 0 \\ 3 & 0 & 5 \\ 0 & 4 & 0 \end{bmatrix} \end{array}$$

$$\rho = \sum_{k=1}^{n} \sum_{j=i}^{n} I(k + x, j + y) - T(k, j)$$

$$
\begin{array}{c|ccccc|}
 & y \longrightarrow & & & & \\
\hline
 & 2 & 1 & 0 & 0 & 3 \\
x & 0 & 0 & 5 & 0 & 0 \\
\downarrow & 0 & 4 & 0 & 6 & 0 \\
 & 1 & 0 & 5 & 0 & 0 \\
\hline
\end{array}
$$

Figure 8-4 A 4×5 image.

As the template is 3×3 and the image 4×5, the possible translations are
and $0 \leq y \leq 2$. From Eq. (8-2-1), the performance index values correspon
translations of the template are:

$$\rho(0, 0) = 8$$

$$\rho(0, 1) = 32$$

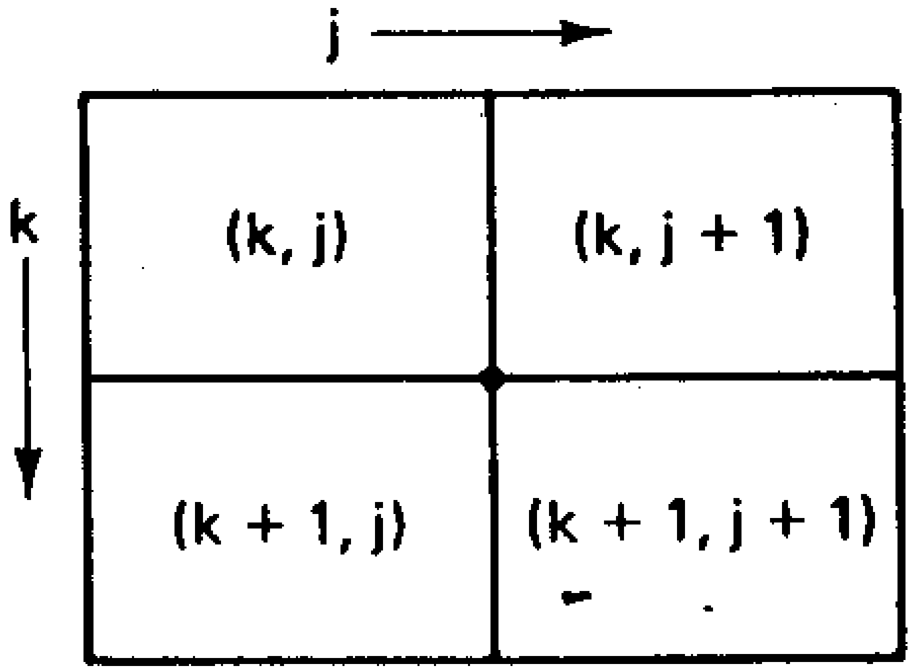
$$\rho(0, 2) = 16$$

$$\rho(1, 0) = 30$$

$$\rho(1, 1) = 4$$

$$\rho(1, 2) = 32$$

Edge detection



$$\nabla I_1(k, j) \triangleq \frac{[I(k+1, j) - I(k, j)] + [I(k+1, j+1) - I(k, j+1)]}{2\Delta x}$$

$$\nabla I_2(k, j) \triangleq \frac{[I(k, j+1) - I(k, j)] + [I(k+1, j+1) - I(k+1, j)]}{2\Delta y}$$

Corner point detection

$$
\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}
\quad
\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\quad
\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\quad
\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}
\quad
\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}
\quad
\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}
\quad
\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}
\quad
\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}
$$

Compression / encoding

- Large number of pixels will result in a very large image file size.
- Storing will be require a lot of space.
- Before storing the digital image is encoded or compressed.

Encoding – Run Length

y →

x ↓

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

Line descriptor

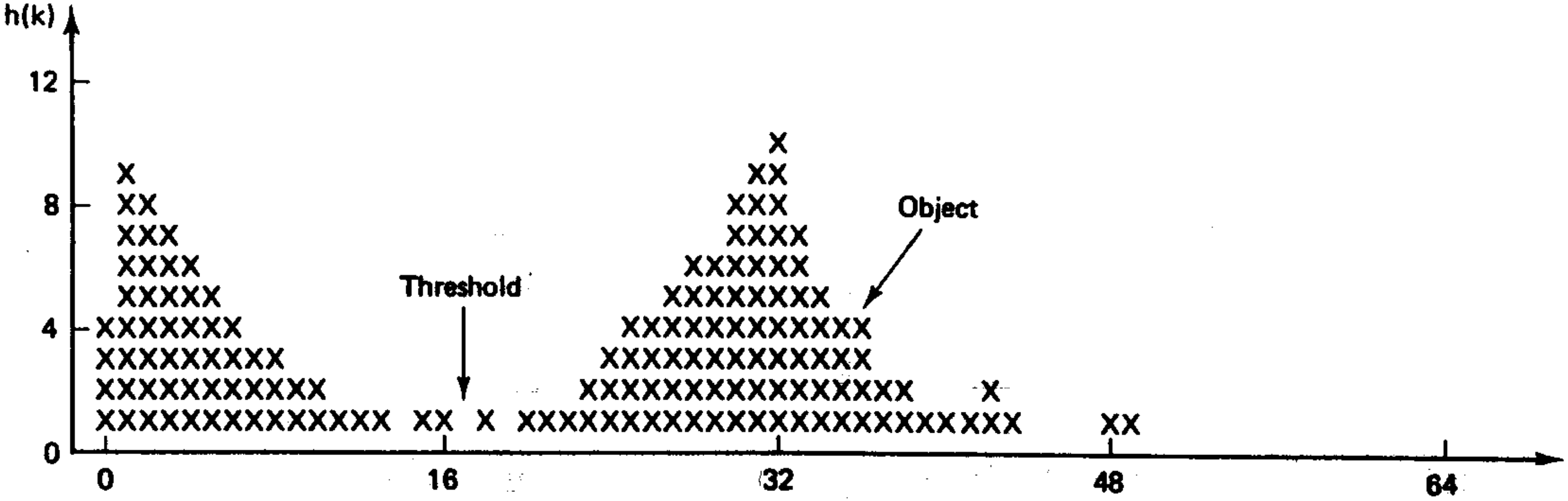
3	2	1
4	p	0
5	6	7

$$
\begin{array}{c}
j \longrightarrow \\
\begin{array}{cc}
k \big\downarrow &
\begin{array}{|ccccccccccccccccccc|}
\hline
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&1&0&0&0&0\\
0&0&0&1&1&1&1&1&0&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0\\
0&0&0&1&1&1&1&1&1&0&0&1&1&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
\hline
\end{array}
\end{array}
\end{array}
$$

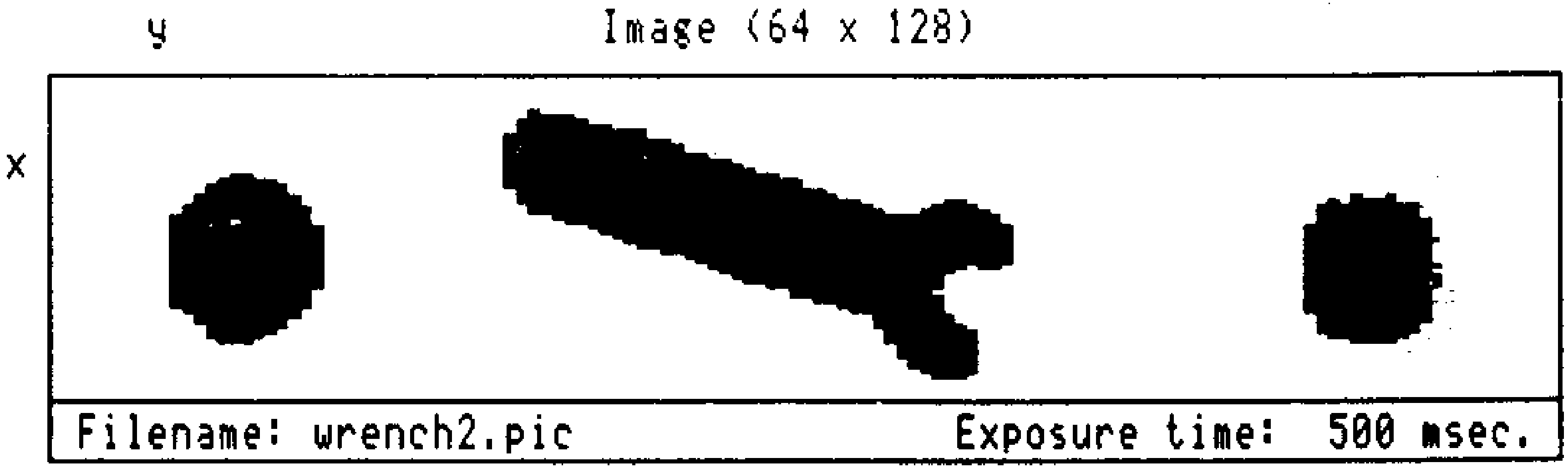
traversing counterclockwise. Then the length of t
and the chain code of the boundary is:

$a = [3, 3, 4, 3, 4, 4, 4, 5, 5, 4, 4, 4, 4, 5, 7,$

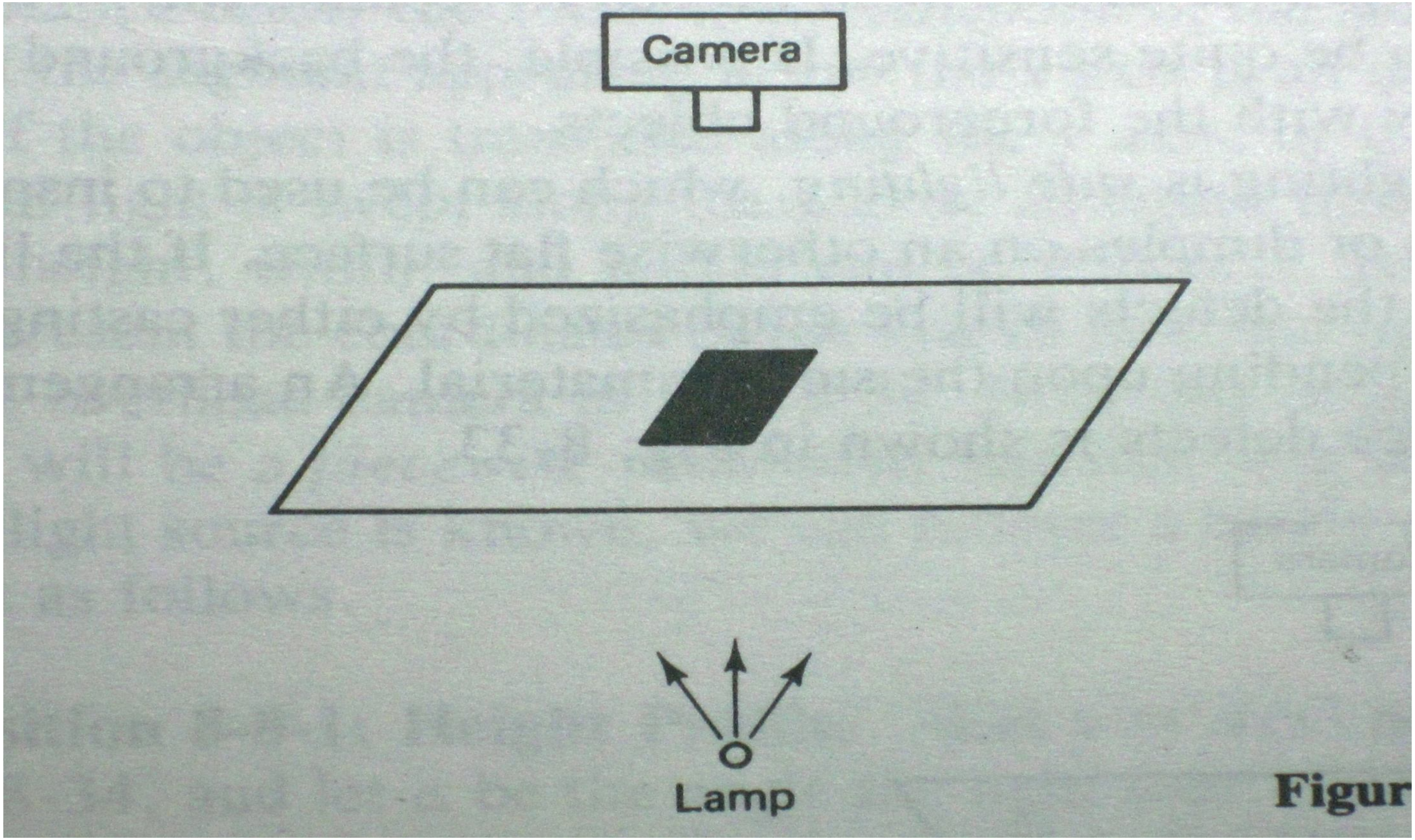
Segmentation – Thresholding to distinguish between object and background



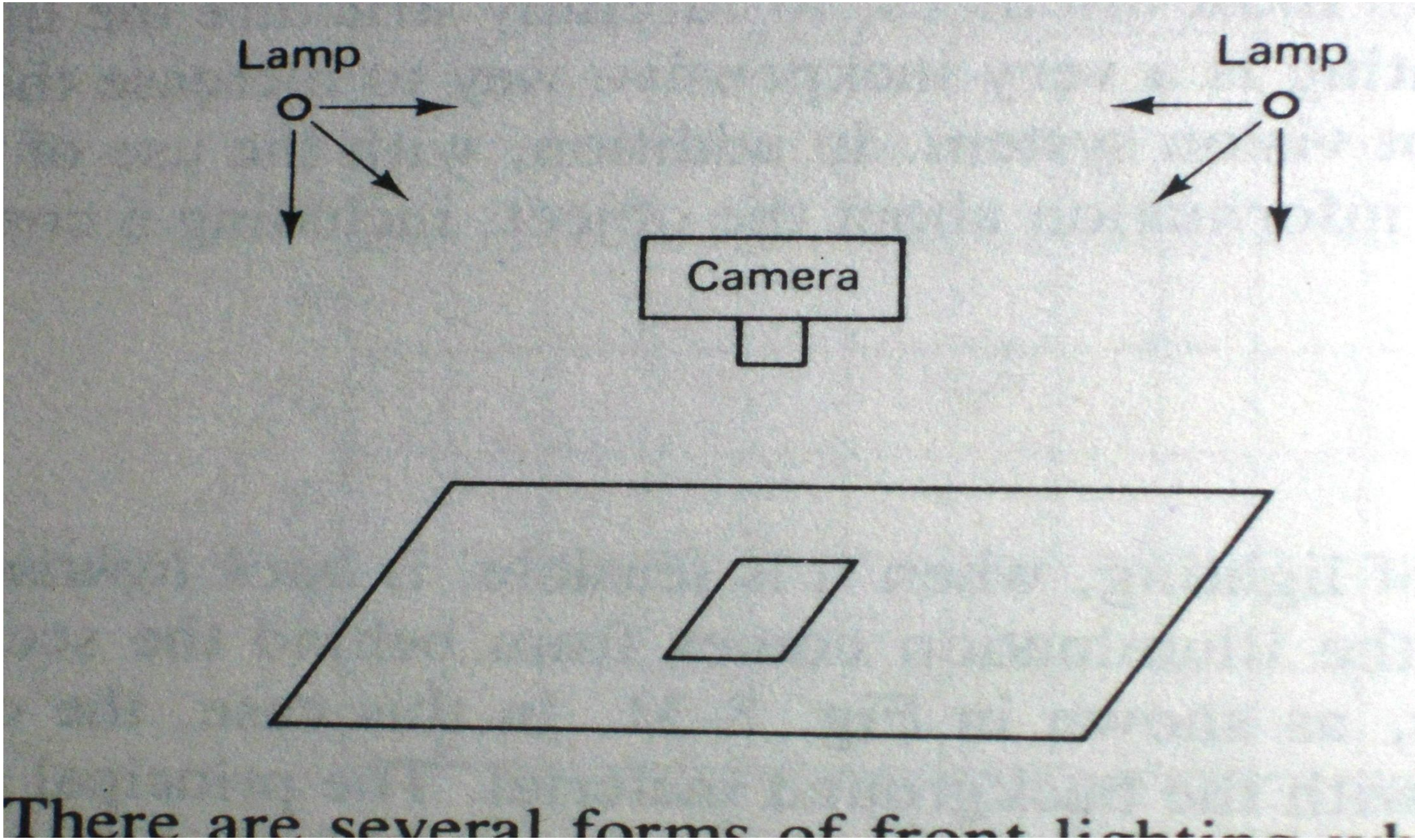
Detection of objects on conveyor
- Shrink operator
- Swell operator



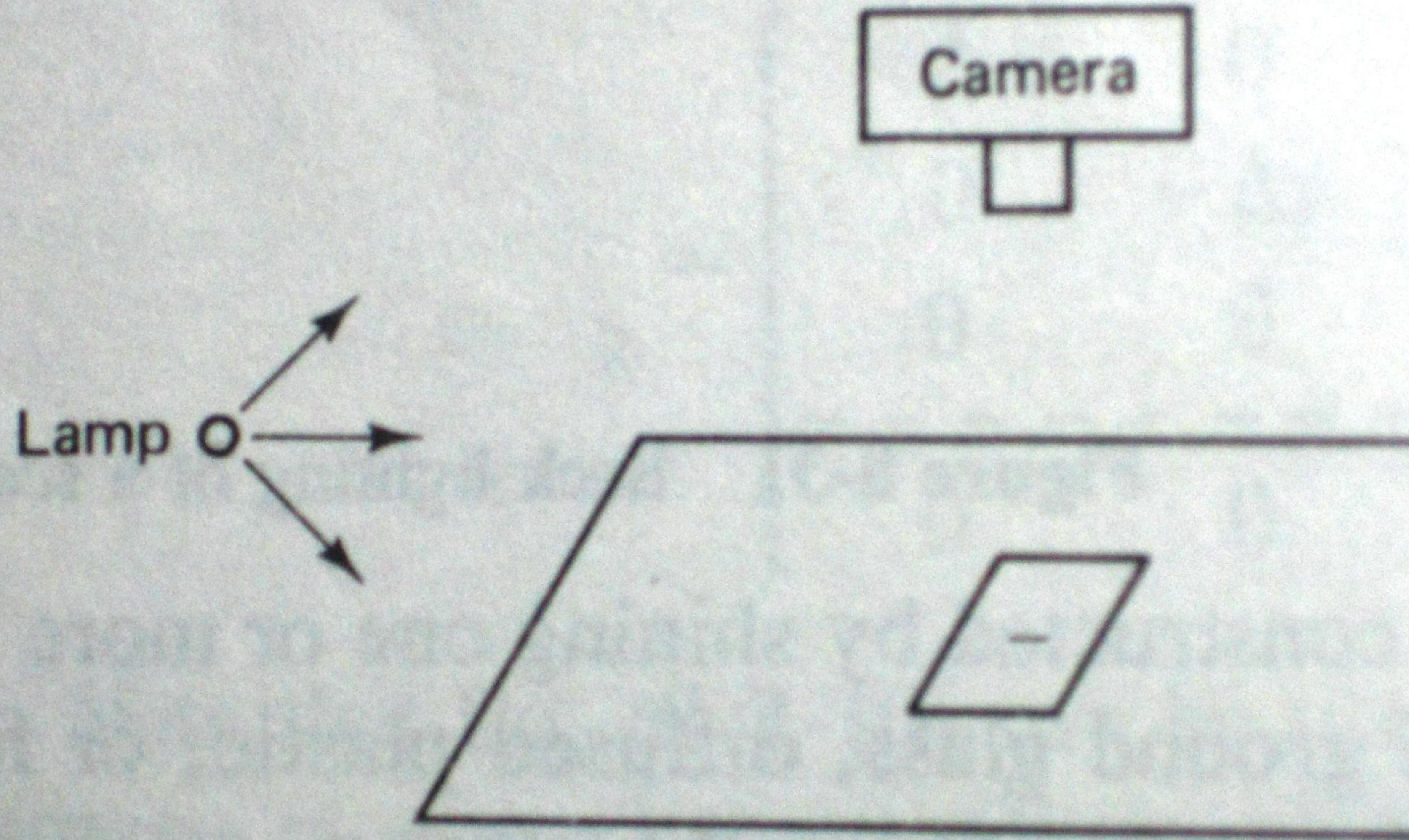
Lighting



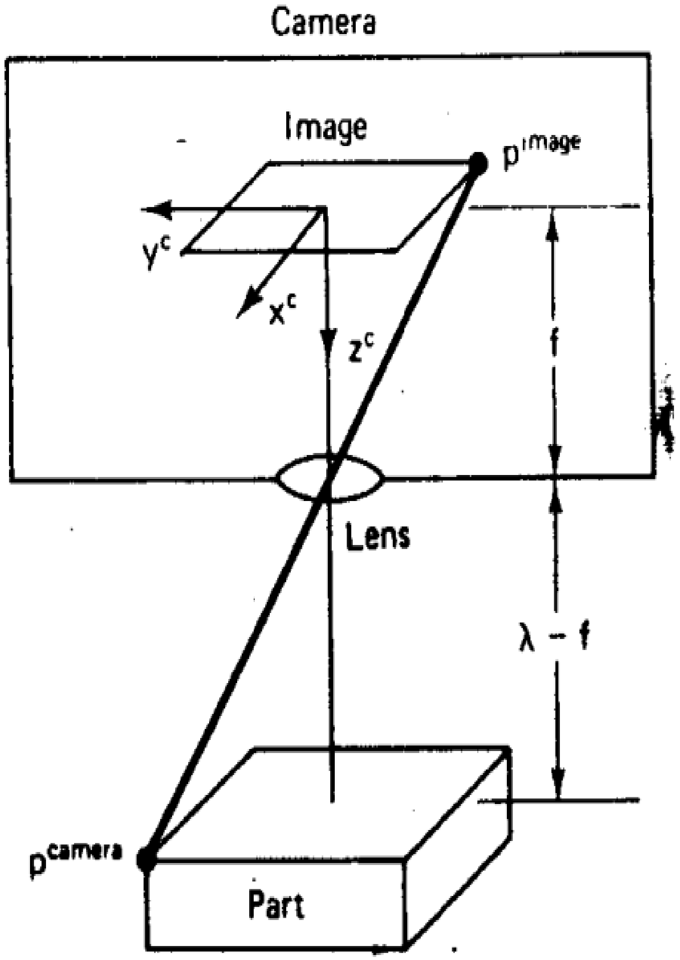
Double lighting



e lighting to highlight surface



Perspective transformation



$$-\frac{p_1^{image}}{f} = \frac{p_1^{camera}}{p_3^{camera} - f}$$

$$-\frac{p_2^{image}}{f} = \frac{p_2^{camera}}{p_3^{camera} - f}$$

$$p_3^{image} = 0$$

Conversion from world coordinate to image coordinate

$$p^{\text{image}} = \begin{bmatrix} -\dfrac{fp_1^{\text{camera}}}{p_3^{\text{camera}} - f} \\ -\dfrac{fp_2^{\text{camera}}}{p_3^{\text{camera}} - f} \\ 0 \end{bmatrix}$$

Inverse transform

$$T_{\text{camera}}^{\text{image}}(\lambda) = \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \mathrm{f} & \dfrac{f\lambda}{f-\lambda} \\ \hdashline 0 & 0 & 1 & \dfrac{f}{f-\lambda} \end{array}\right]$$

Proof. Using Eq. (8-7-4) and the general properties of homogeneous coordinates:

$$T_{\text{camera}}^{\text{image}}(\lambda)p^{\text{image}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & f & \dfrac{f\lambda}{f-\lambda} \\ 0 & 0 & 1 & \dfrac{f}{f-\lambda} \end{bmatrix} \begin{bmatrix} p_1^{\text{image}} \\ p_2^{\text{image}} \\ 0 \\ 1 \end{bmatrix}$$

$$= \left[p_1^{\text{image}}, p_2^{\text{image}}, \frac{f\lambda}{f-\lambda}, \frac{f}{f-\lambda}\right]^T$$

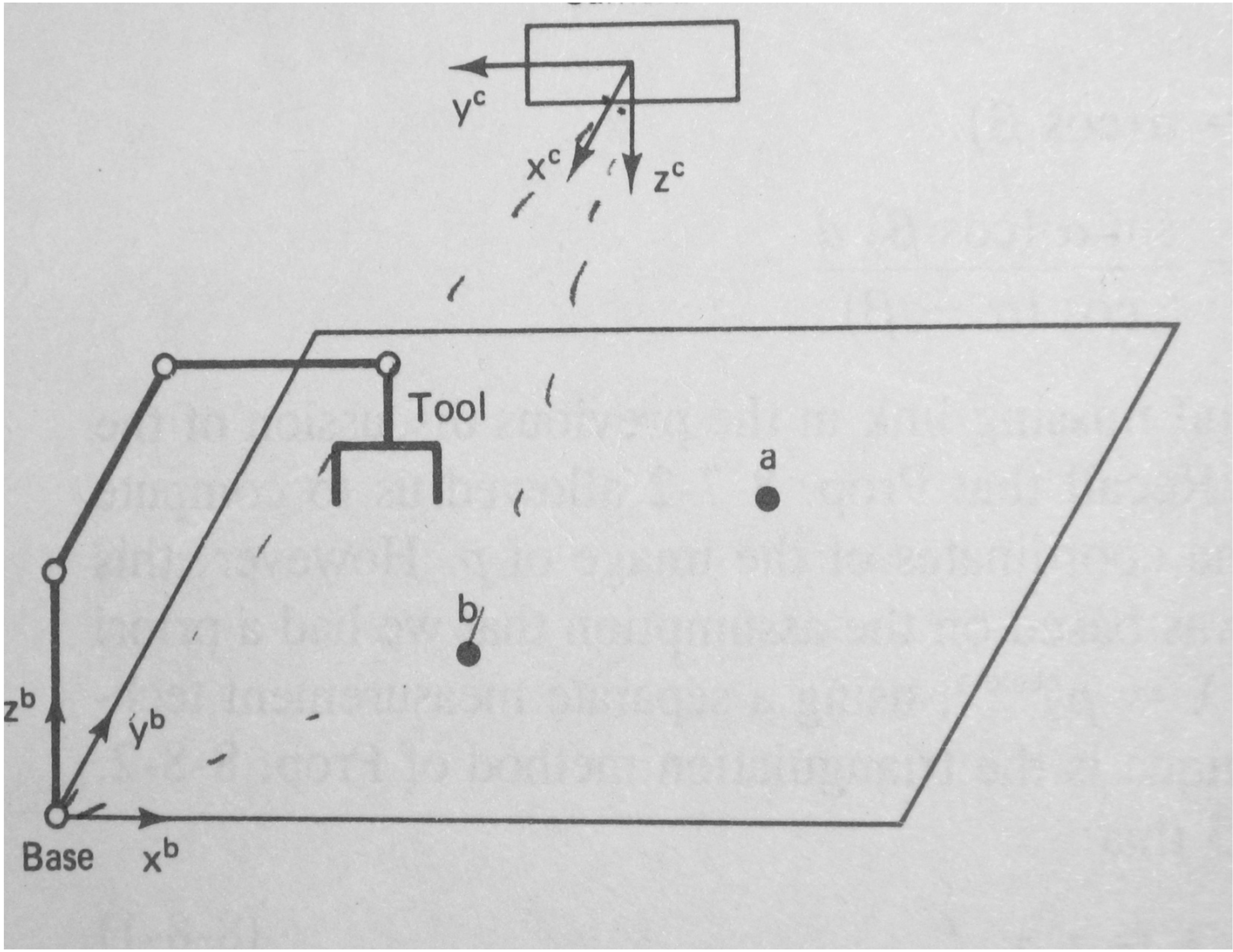
$$= \left[\frac{(f-\lambda)p_1^{\text{image}}}{f}, \frac{(f-\lambda)p_2^{\text{image}}}{f}, \lambda, 1\right]^T$$

$$= [p_1^{\text{camera}}, p_2^{\text{camera}}, p_3^{\text{camera}}, 1]^T$$

$$= p^{\text{camera}}$$

The following example is an illustration of the...

Camera calibration – before using a camera



Unknown Xo, Yo, Zo

$$T_{\text{base}}^{\text{camera}} = \left[\begin{array}{ccc|c} 0 & -1 & 0 & x_0 \\ -1 & 0 & 0 & y_0 \\ 0 & 0 & -1 & z_0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

$$
\begin{aligned}
T_{\text{image}}^{\text{base}} &= T_{\text{image}}^{\text{camera}}\, T_{\text{camera}}^{\text{base}} \\
&= T_{\text{image}}^{\text{camera}} \left[T_{\text{base}}^{\text{camera}} \right]^{-1} \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & x_0 \\ -1 & 0 & 0 & y_0 \\ 0 & 0 & -1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & y_0 \\ -1 & 0 & 0 & x_0 \\ 0 & 0 & -1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & -1 & 0 & y_0 \\ -1 & 0 & 0 & x_0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/f & (f - z_0)/f \end{bmatrix}
\end{aligned}
\qquad
\begin{bmatrix} a_1 \\ a_2 \\ 0 \\ 1 \end{bmatrix}
$$

$$f(y_0 - a_2) = a_1^{\text{image}}(f - z_0)$$

$$f(x_0 - a_1) = a_2^{\text{image}}(f - z_0)$$

$$f(y_0 - b_2) = b_1^{\text{image}}(f - z_0)$$

$$f(x_0 - b_1) = b_2^{\text{image}}(f - z_0)$$

$$z_0 = f\left[1 + \frac{(a_2 - b_2)}{a_1^{\text{image}} - b_1^{\text{image}}}\right]$$

$$y_0 = a_2 + \frac{(f - z_0)a_1^{\text{image}}}{f}$$

$$x_0 = a_1 + \frac{(f - z_0)a_2^{\text{image}}}{f}$$

$$
\begin{aligned}
T_{\text{image}}^{\text{base}} &= T_{\text{image}}^{\text{camera}}\, T_{\text{camera}}^{\text{base}} \\
&= T_{\text{image}}^{\text{camera}} [T_{\text{base}}^{\text{camera}}]^{-1} \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & x_0 \\ -1 & 0 & 0 & y_0 \\ 0 & 0 & -1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & y_0 \\ -1 & 0 & 0 & x_0 \\ 0 & 0 & -1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & -1 & 0 & y_0 \\ -1 & 0 & 0 & x_0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/f & (f - z_0)/f \end{bmatrix} \qquad (8\text{-}9\text{-}2)
\end{aligned}
$$

We can now equate the image coordinates a^{image} with the inferred image coordinates, $T_{\text{image}}^{\text{base}} a$. This can also be done for test point b, and the result, after simplification, is:

$$f(y_0 - a_2) = a_1^{\text{image}}(f - z_0) \qquad (8\text{-}9\text{-}3)$$

$$f(x_0 - a_1) = a_2^{\text{image}}(f - z_0) \qquad (8\text{-}9\text{-}4)$$

$$f(y_0 - b_2) = b_1^{\text{image}}(f - z_0) \qquad (8\text{-}9\text{-}5)$$

$$f(x_0 - b_1) = b_2^{\text{image}}(f - z_0) \qquad (8\text{-}9\text{-}6)$$