Vetter's Method

Given an input image of a face

1. Compute correspondence with a "reference face" in the same pose

   (Uses coarse-to-fine optical flow algorithm using a Laplacian pyramid)

   Result is a "shape vector"

   $s = (x_1, y_1, \ldots, x_n, y_n)$

2. Compute "texture vector"

   Difference map between corresponding pixels' image intensities

   $t = (i_1, \ldots, i_n)$

3. Linear Shape Model of Faces

   Given a database of m 3D models of m faces, any new face can be described by

   $$S = \sum_{i=1}^{m} \beta_i S_i$$

   where $S = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)$ is the new face's 3D shape and S_i is the 3D shape of the i^{th} face in the DB.

   * Arbitrarily rotating 3D shape and projecting into an image does not change the β_i's!

   $$s_r = \sum_{i=1}^{m} \beta_i s_i^r$$

   where $s_r = PS^r$, P = perspective projection

4. Compute linear shape approximation

→ Find best β_i's (min error) so that

$$s^r = \sum_i \beta_i s_i^r$$

where s^r = input image face

s_i^r = i^{th} reference face image at same pose

5. Compute new view's shape vector

$$s^f = \sum_i \beta_i s_i^f$$

6. Compute linear texture approximation

$$t^r = \sum_i \alpha_i t_i^r$$

7. Compute new view's texture vector

$$t^f = \sum_i \alpha_i t_i^f$$

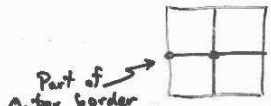
Note: This texture mapping is not correct except for Lambertian surfaces. Alternatively, use 3D head model to remap texture

8. Warp texture onto shape

$$s^f + t^f$$

2D Boundary Representation

- Any connected component is completely determined if we know its **borders**
- **4-border** of S = set of points of S that have at least 1 4-neighbor in $\bar{S}$
- A point may be on more than 1 border:



- Outer border = set of border points adjacent to **background** (closed curve)
- Hole border = closed curve of border pts adjacent to a **hole**.

Edge-Crawl Algorithm

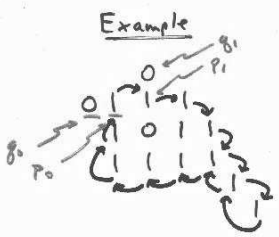
- Trace the 4-border of an 8-connected component C given border point p_0 and 4-adjacent neighbor $q_0 \in \bar{C}$

Step 1: $i = 0$

Step 2: /* Let neighbors of p_i in clockwise order starting at q_i be $q_i = r_1, r_2, \ldots, r_8$ */
for ($j = 2$ until 8) do
if $r_j \in C$ then goto Step 3
endfor
halt; // C = 1 pixel only

Step 3: $p_{i+1} = r_j$ // $r_j \in C$
$q_{i+1} = r_{j-1}$ // $r_{j-1} \notin C$
if ($p_0 == p_{i+1}$ and $q_0 \in \{r_1, \ldots, r_j\}$)
then halt
else goto Step 2

Example



- Consecutive p_i's are 8-neighbors
- Instead of representing coords of p_i's, represent direction to next border point

3	2	1	i^{th} neighbor in direction $45i^\circ$ rel. to positive x-axis
4	p_i	0	
5	6	7	

$\Rightarrow$ 0076773344422

Chain Code

- Why termination test includes g_0

8 $\rightsquigarrow$ A border point may occur multiple times on a given border

- Perimeter $= (n_0 + n_2 + n_4 + n_6) + \sqrt{2}(n_1 + n_3 + n_5 + n_7)$

where n_i = # occurrences of link i

- Chain code is translation invariant but not rotation invariant