Edge Contour Extraction

(Pitas, section 5.5, Sonka et al., sections 5.2.4-5.2.5)

- Edge detectors typically produce short, disjoint edge segments.

- These segments are generally of little use until they are aggregated into extended edges.

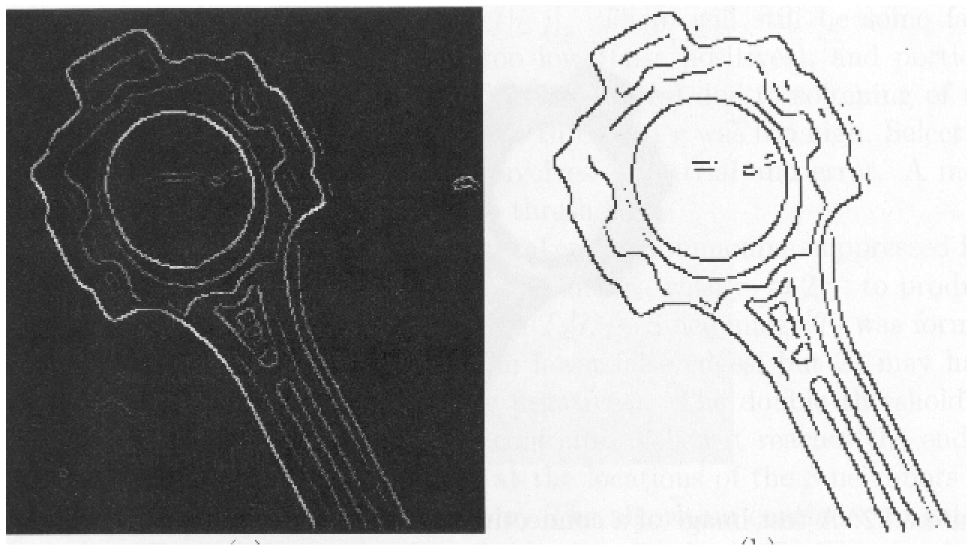
- We assume that edge thinning has already be done (e.g., non-maxima suppression).

- Two main categories of methods:

  (1) local methods (extend edges by seeking the most "compatible" candidate edge in a neighborhood).

  (2) global methods (more computationally expensive - domain knowledge can be incorporated in their cost function).

Note: local methods can not handle big gaps.



Local processing methods

- Some important observations useful for contour extraction:

  * The output of edge detectors tend to have approximately constant intensity along object boundaries.

  * Image edges and lines are smooth and tend to have low curvature.

  * Small local edge direction differences ensure smooth object boundaries.

(1) At each edge pixel, a neighborhood (e.g., 3x3) is examined.

(2) The center edge pixel can be linked with its neighbors if the magnitude and direction differences are below certain thresholds and their magnitudes are relatively large:

$$|e(p_i) - e(p_j)| \leq T_1$$

$$|\phi(p_i) - \phi(p_j)| \leq T_2$$

$$|e(p_i)| \geq T \qquad |e(p_j)| \geq T$$

• Contour extraction using heuristic search

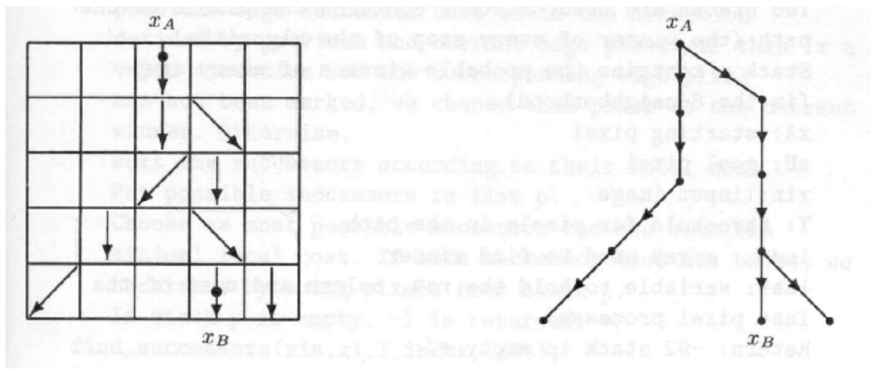
- A more comprehensive approach to contour extraction is based on graph searching.

- Graph representation of edge points:

  (1) Edge points at position p_i correspond to graph nodes.

  (2) The nodes are connected to each other if local edge linking rules (e.g., like the ones given previously), are satisfied.

  (the edge linking rules may be modified to suit the requirements of a particular problem).



- The generation of a contour (if any) from the pixel p_A to the pixel p_B is equivalent to the generation of a minimum-cost path in the directed path.

- A cost function for a path connecting nodes $p_1 = p_A$ to $p_N = p_B$ could be defined as follows:

$$C(p_1, p_2, \ldots, p_N) = -\sum_{k=1}^{N} |e(p_k)| + a \sum_{k=2}^{N} |\phi(p_k) - \phi(p_{k-1})| + b \sum_{k=2}^{N} |e(p_k) - e(p_{k-1})|$$

- Finding a minimum-cost path is not trivial in terms of computation (typically, the approach is to sacrifice optimality for the sake of speed).

• Contour extraction using dynamic programming

Dynamic programming

- It is an optimization method that searches for optima of functions in which not all the variables are simultaneously interrelated.

- It subdivides a problem recursively into smaller subproblems that may need to be solved in the future, solving each subproblem - proceeding from the smaller ones to the larger ones - and storing the solutions in a table that can be looked up as and when need arises.

Principle of optimality (applied to the case of graph searching): the optimal path between two nodes p_A, p_B, can be split into two optimal sub-paths $p_A p_i$ and $p_i P_B$ for any p_i lying on the optimal path $p_A p_B$.

Express the problem in recursive form

- Consider the following objective function to be maximized:

$$F(p_1, p_2, \ldots, p_N) = \sum_{k=1}^{N} |e(p_k)| - a \sum_{k=2}^{N} |\phi(p_k) - \phi(p_{k-1})|$$

- The objective function can be written in a recursive form as follows:

$$F(p_1, p_2, \ldots, p_k) = F(p_1, p_2, \ldots, p_{k-1}) + f(p_{k-1}, p_k)$$

where $f(p_{k-1}, p_k) = |e(p_k)| - a|\phi(p_k) - \phi(p_{k-1})|$

Complete optimization problem

$$\hat{F}(\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_k) = \max_{p_i \; i=1,2,\ldots,k} [F(p_1, p_2, \ldots, p_k)]$$

Solution using dynamic programming

- The optimal path $\hat{p}_1 \hat{p}_k$ can be split into two optimal sub-paths $\hat{p}_1 \hat{p}_{k-1}$ and $\hat{p}_{k-1} \hat{p}_k$ satisfying the following recursive relation:

$$\hat{F}(\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_k) = \max_{p_i \; i=1,2,\ldots,k} [F(p_1, p_2, \ldots, p_{k-1}) + f(p_{k-1}, p_k)] =$$

$$\max_{p_k} [\hat{F}(\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_{k-1}) + f(\hat{p}_{k-1}, p_k)]$$

- The initial value of $\hat{F}(\hat{p}_1)$ is given by:

$$\hat{F}(\hat{p}_1) = |e(p_1)|$$

- Dynamic programming has reduced the global opimization problem to N stages of two variable optimization.

A-algorithm (by Nilsson, 1980)

(1) Expand the starting node x_A and put all its successors into an OPEN list with pointers back to the starting node x_A. Evaluate the cost $f()$ for each expanded node.

(2) If the OPEN list is empty, fail. Determine the node x_i from the OPEN list with the lowest associated cost $f(x_i)$ and remove it, If $x_i = x_B$, then trace back through the pointers to find the optimum path and stop.

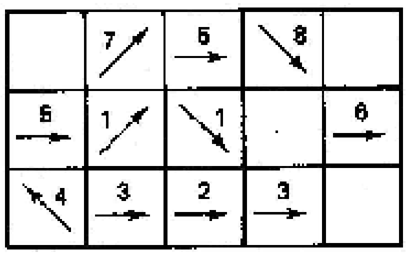
(3) If the option to stop was not taken in step 2, expand the specified node x_i, and put its successors on the OPEN list with pointers back to x_i. Compute their costs $f()$. Go to step 2.

- Note that this algorithm does not guarantee the globally optimum path.

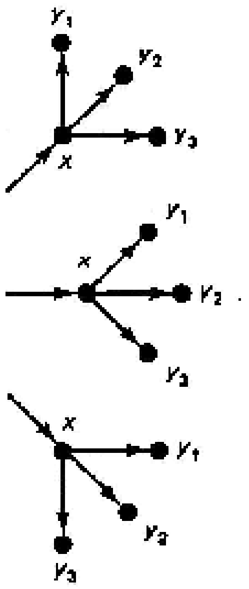
- One disadvantage of the heuristic search algorithm is that short paths may have smaller cost than longer paths that are more likely to be the final winners.

- Pruning certain paths (e.g., short ones, paths having high cost per unit length) can help alleviate some of these problems.

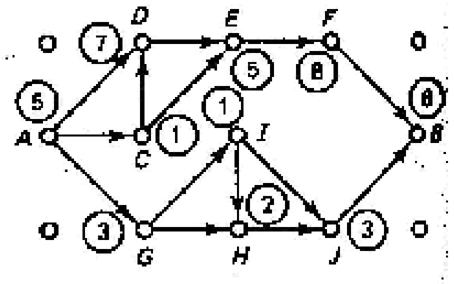
• An example



(a) Gradient magnitudes contour directions



(b) Linkage rules



(c) Graph interpretation

- A pixel is linked to one of its three 8-connected neighbors in front of the contour direction.

- $f(x_k)$ is the sum of edge gradient magnitudes along the path from x_A to x_k (i.e., max cost is used in this example).

- The algorithm finds the path ADEFB -- cost 31 (the optimum path, however, is the path ACDEFB -- cost 32)

Global processing methods

- If the gaps between pixels are very large, local processing methods are not effective.

- Global methods are more effective in this case !!

  Hough Transform can be used to determine whether points lie on a curv ed of a specified shape (model-based method).

  Deformable Models (Snakes) can be used to extract the boundaries of objects having arbitrary shapes.

  Grouping can be used to decide which groups of features are likely to be part of the same object.