Corridor Scissors: A Semi-Automatic Segmentation Tool Employing Minimum-Cost Circular Paths

Dirk Farin, Magnus Pfeffer,
Peter H. N. de With,
Wolfgang Effelsberg

Contact address:
Dirk Farin
Eindhoven University of Technology (TU/e)
Design Technology Institute
5600 MB, Eindhoven, The Netherlands
d.s.farin@tue.nl

Introduction

- Image segmentation is a requirement for
  - Image/video editing – placing objects into new context
  - Annotation of image objects / Hyperlinking
  - Image-based metrology (especially for medical images)
- Automatic segmentation
  - Only works for limited application areas.
  - Clear object-model description is required.
  - Many problem cases (shadows, occlusions, fuzzy boundaries).
  - Fails for many situations or gives inaccurate results.
- Manual segmentation (without computer assistance)
  - Very time consuming, not practical for most applications.
- Semi-automatic segmentation (manual with computer assistance)
  - User marks coarse object boundary.
  - Computer refines the drawn boundary to a pixel-exact segmentation.

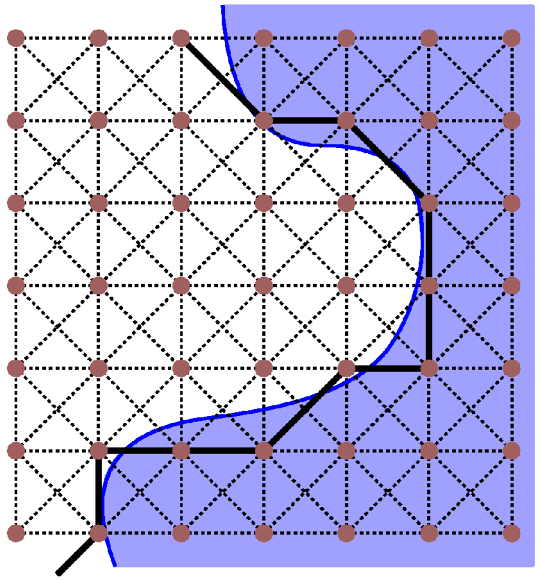
Presentation Outline

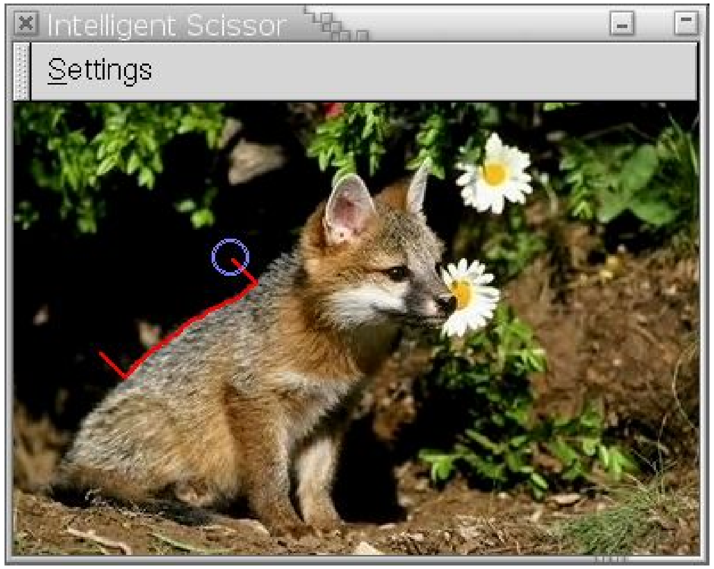
- Introduction to *Intelligent Scissors* algorithm.
  - Semi-automatic segmentation algorithm.
  - Based on computing shortest paths between two points.
- Problems with the user interface.
  - Difficult to make corrections to previously made segmentations.
  - High computational complexity since graph search works on complete image.

- Our new segmentation tool: ***Corridor Scissors***.
  - More intuitive user interface, allowing gradual improvements of the result.
  - Reduced computational complexity.
  - Based on computing shortest circular paths in a ring-shaped graph.
- Main new contribution: algorithm for computing min.-cost circular paths.

The Intelligent-Scissors Tool

- Tool to assist manual segmentation
  - User traces coarsely along the object border
  - The algorithm snaps the contour to the accurate object border.





- Basic idea: define a graph on the image
  - Image pixels are the graph nodes
  - Edges between neighboring pixels are weighted with the gradient strength
  - Contour is determined by computing lowest cost path (Dijkstra algo.) between starting point and current mouse position.

Definition of Graph-Edge Costs

- Edges in the graph are attributed with weights *f*
  - Edge weight *f* is composed of two components

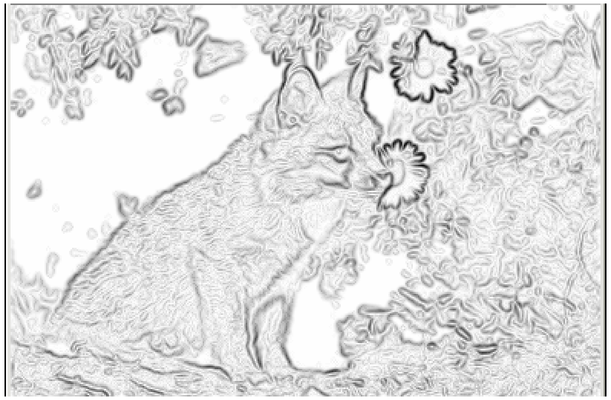
$$f = f_G + \alpha f_Z$$

  - f_G is based on the luminance gradient in the image (low costs along strong image gradients).

$$f_G = 1 - \frac{||\nabla I||}{\max ||\nabla I||}$$

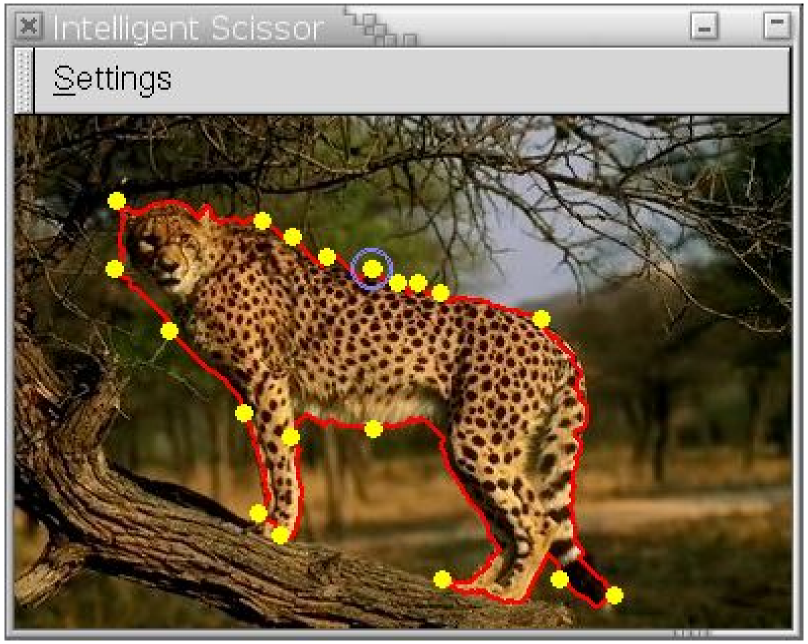
  - f_Z is 0 at zero-crossings of the Laplacian
    - Provides better localization along unsharp boundaries.

$$f_Z = \begin{cases} 0 \text{ at zero crossings of } \nabla^2 I, \\ 1 \text{ otherwise} \end{cases}$$



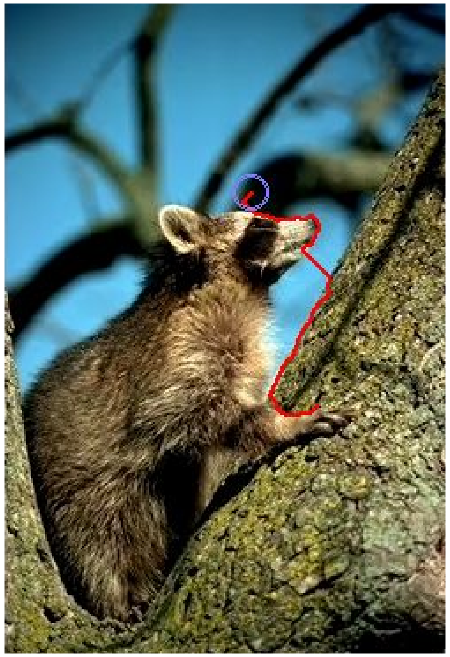
The User-Interface / Setting Control-Points

- User marks the starting point of the graph search.
- Algorithm determines shortest path to the current mouse position.
- When the path is distracted from the correct object boundary,
  - the user re-initiates the search by placing a control point,
  - a new search is started from this control point.
- Once a control point is placed, the previous contour is fixed.
- Unintuitive user-interface since control-points have no semantic meaning.



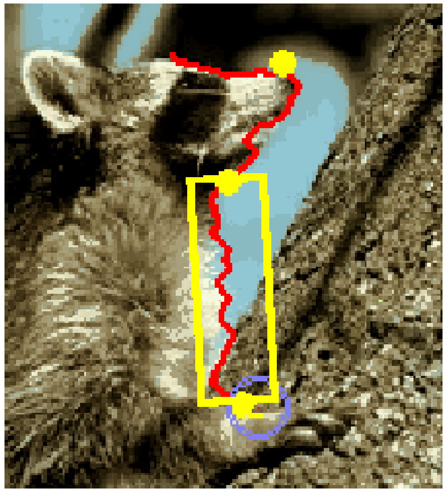
The *Snap-to-High-Contrast* Problem

- For longer distances, the path often snaps to wrong boundaries, because
  - the cost for this boundary is especially low,
  - so that even high-cost areas are traversed.



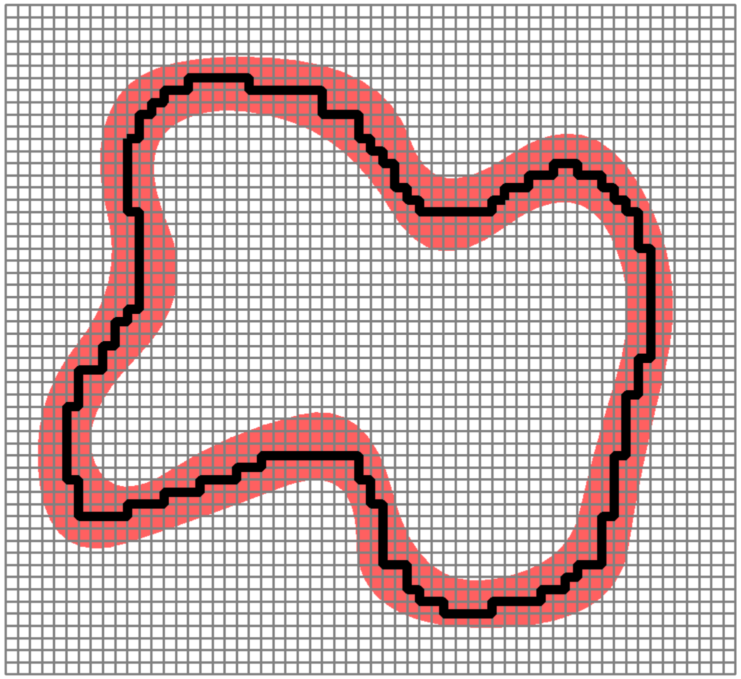
Rubberband Extension [Luo:ICIP'02]

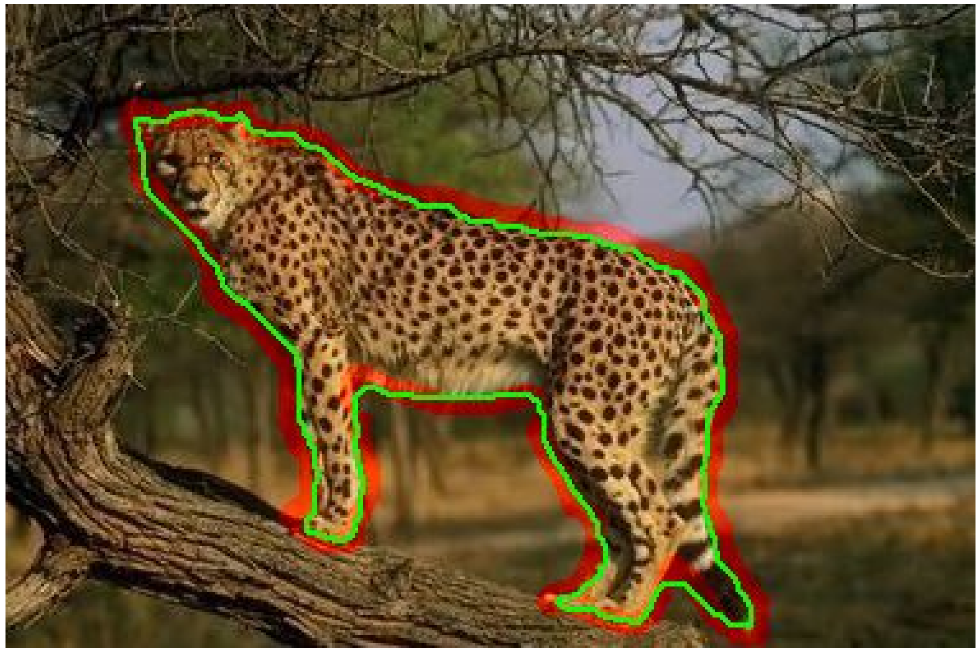
- Limit the search-area to a small rectangle between last control point and current position.
- The width of the rectangle is adjusted by the user.
- Snapping to wrong edges is prevented if they are outside of the search-area.
- Problems:
  - More parameters to adjust by the user.
  - Limiting the search to the rectangle area requires to set control points frequently.



Principle of *Corridor Scissors*

- User draws a broad circular corridor along the object boundary.
- Algorithm searches for minimum-cost circular path within the corridor.
- Segmentation can be gradually improved by adapting the corridor
  - Narrow it to force path to correct boundary,
  - widen it to include missing detail.

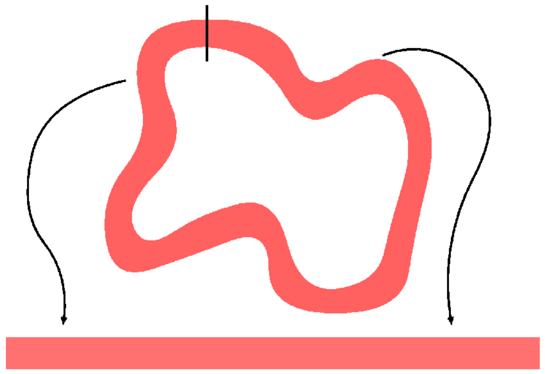




Circular Graph Search (I)

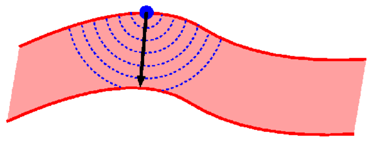
- Main problem:
  - **Algorithm to compute minimum-cost circular path is required.**

- Basic solution approach:
  - Cut circular corridor into linear lane.
  - Find shortest path from left side to right side of lane.
  - Starting point and ending point is not known !
  - Starting point and ending point must be neighbors !



- Preprocessing: Cut the corridor at an arbitrary position into a lane.
  - Start at top-most pixel
  - search for the shortest path

    to the inside of the corridor.



Properties of Shortest Paths

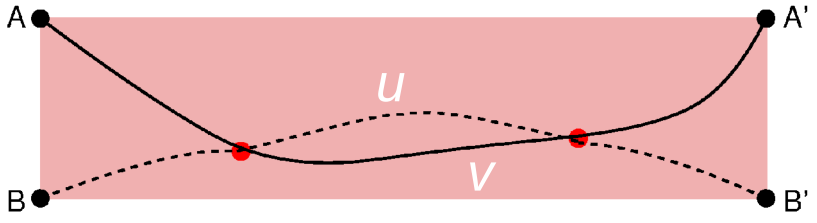
- We make use of this easy theorem:

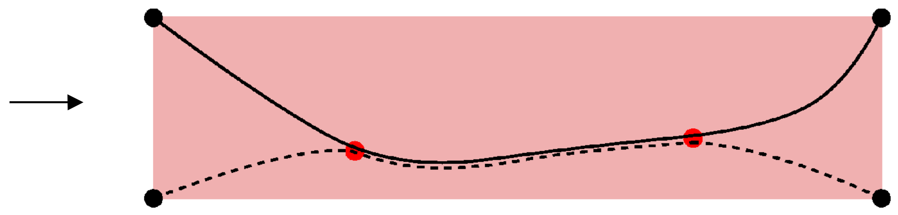
Shortest paths don’t cross twice.

  - Prove:

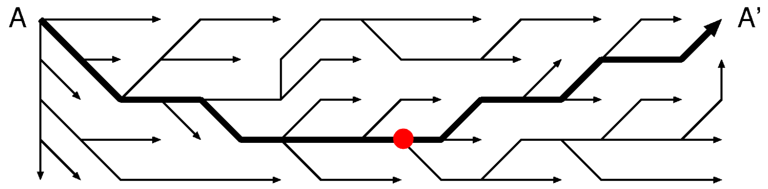
    The more costly sub-path *u* could be replaced with the cheaper sub-path *v*.

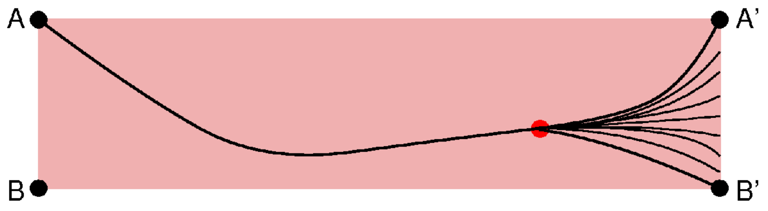
    ↙ Contradiction to assumption of shortest path.





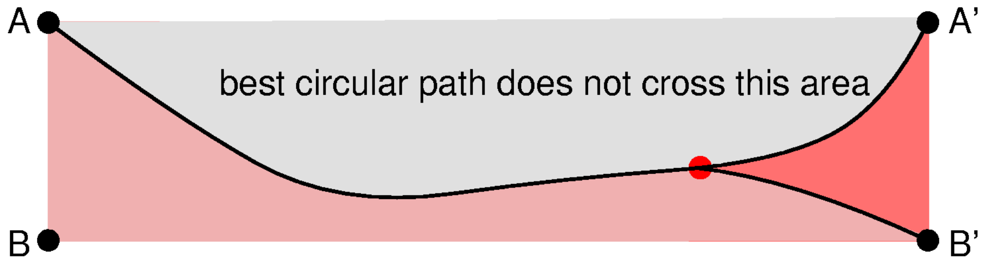
- One run of the Dijkstra algorithm computes
  - complete tree of shortest paths to all nodes, and
  - nodes are computed in the order of increasing total cost.



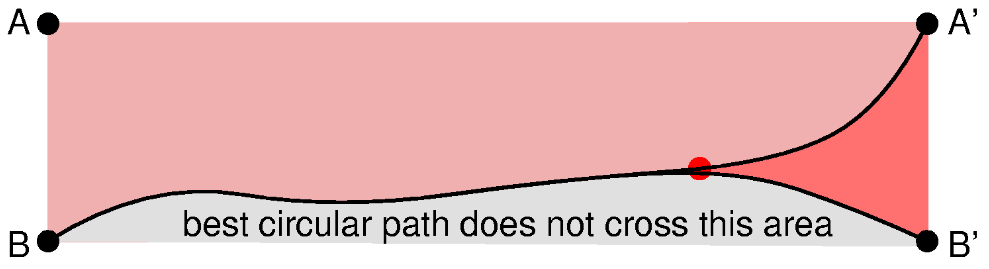


Circular Graph Search (II)

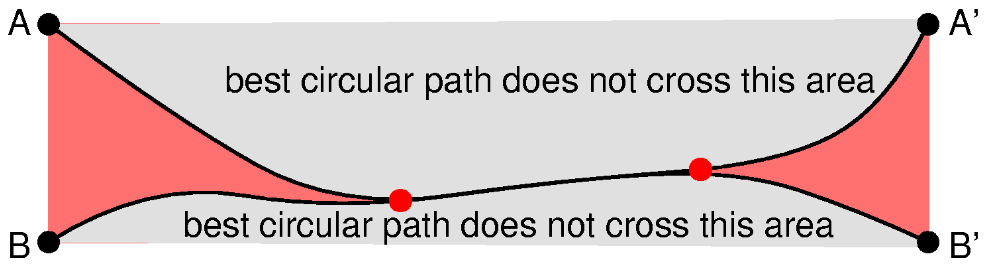
- Start a shortest path search from point A (top-left position).
  - We get shortest paths to every destination node on right side.
  - Shortest paths cannot run through the grey area (otherwise A—A` would be crossed twice).



- Start shortest path search from
  point B (bottom-left position).
  - Shortest paths cannot run through the grey area below B—B'.

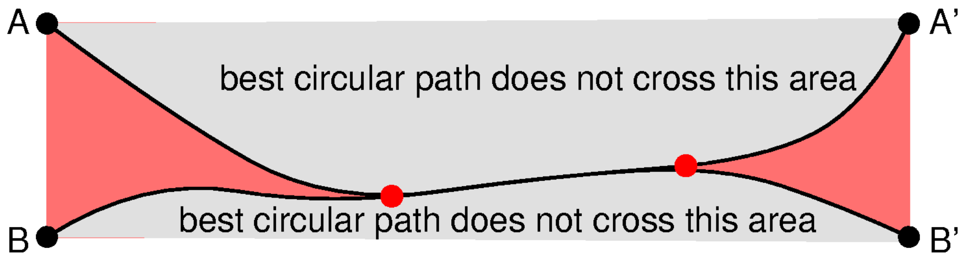


- Since the lane is much longer than wide, both paths will join into a common sub-path.
  - This sub-path is part of every shortest path between left and right side.
  - Hence, this sub-path is also part of the shortest circular path.

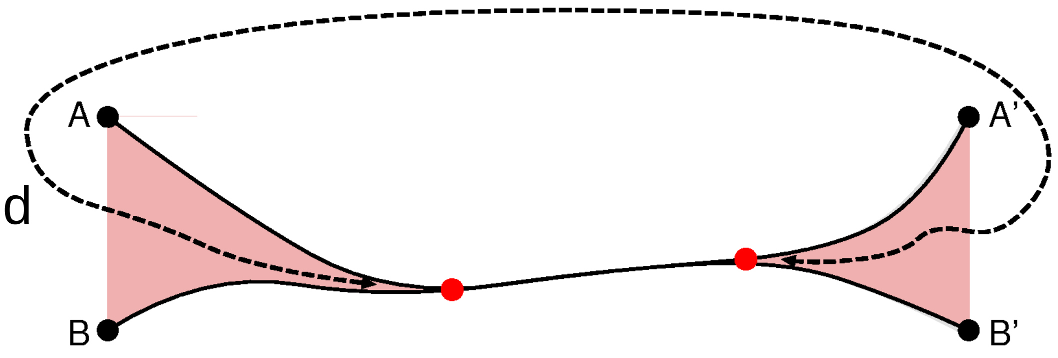

A
A'
best circular path does not cross this area
best circular path does not cross this area
B
B'

Circular Graph Search (III)

- Up to now,
  - we know a sub-path that is also part of the shortest circular path.
  - We still need the best path crossing the corridor cut.

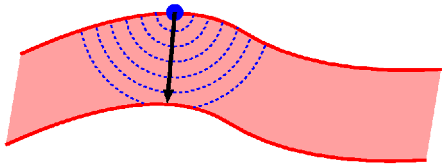

A
A'
best circular path does not cross this area
best circular path does not cross this area
B
B'

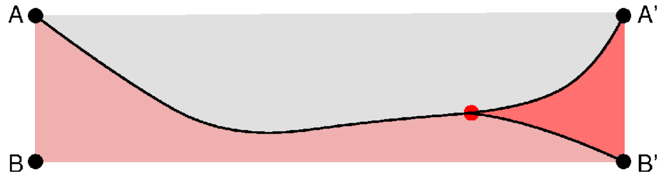
- Do a final shortest-path search between the left joining-node and the right joining-node.

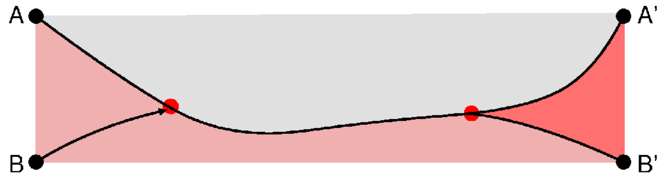


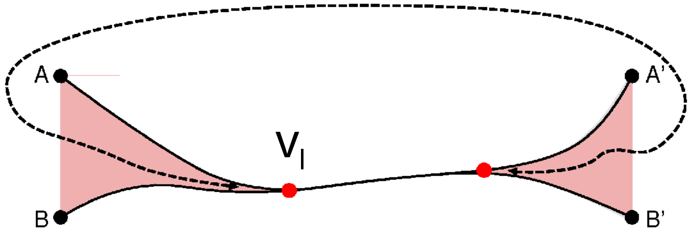
Circular Path Search – Efficient Implementation

- Step 1: cut corridor into linear lane
  - Heuristic, or shortest-path search.
  - Search can be stopped when other side of corridor is reached.
- Step 2: shortest path A—A'
  - One full Dijkstra algorithm run in the corridor area.
- Step 3: shortest path B—B'
  - Approximation: stop as soon as path A—A' is reached.
- Step 4: connect joining-nodes
  - Graph search in the red area (usually very small), or even
  - Searching only in the left part (beginning at v_l and reusing the costs from Step 2).



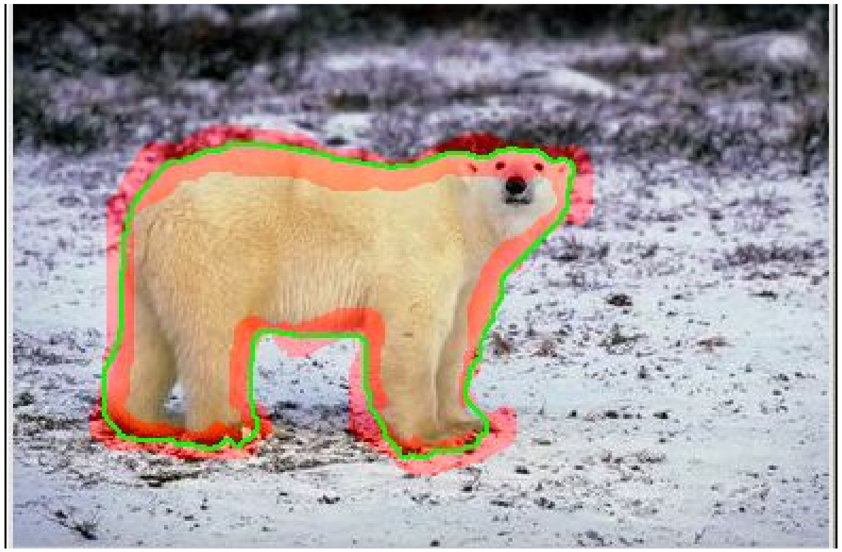
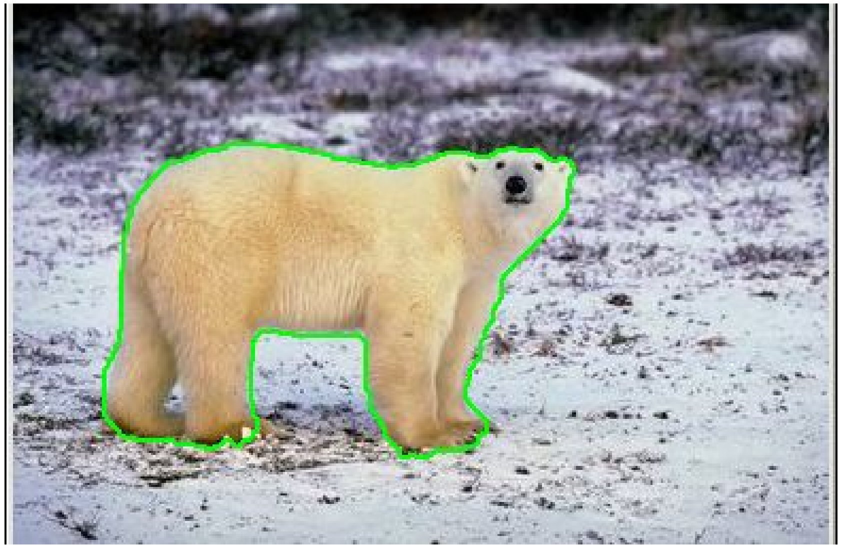
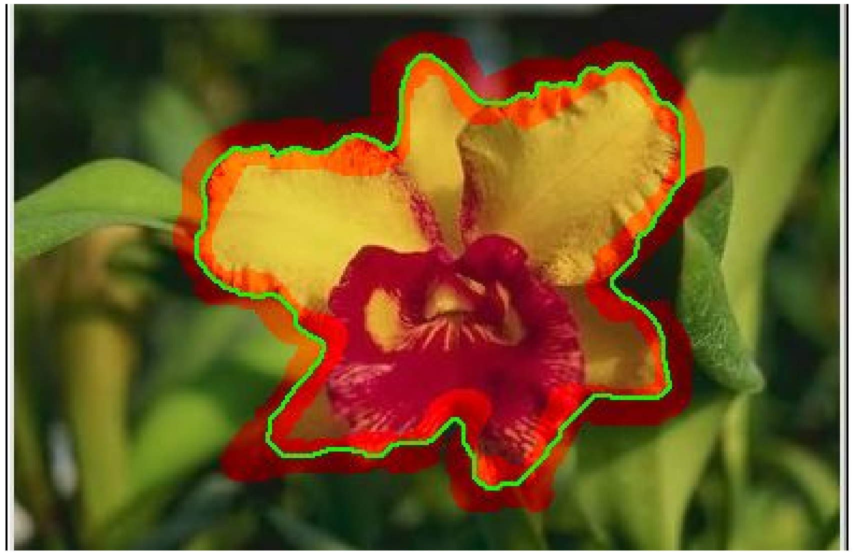
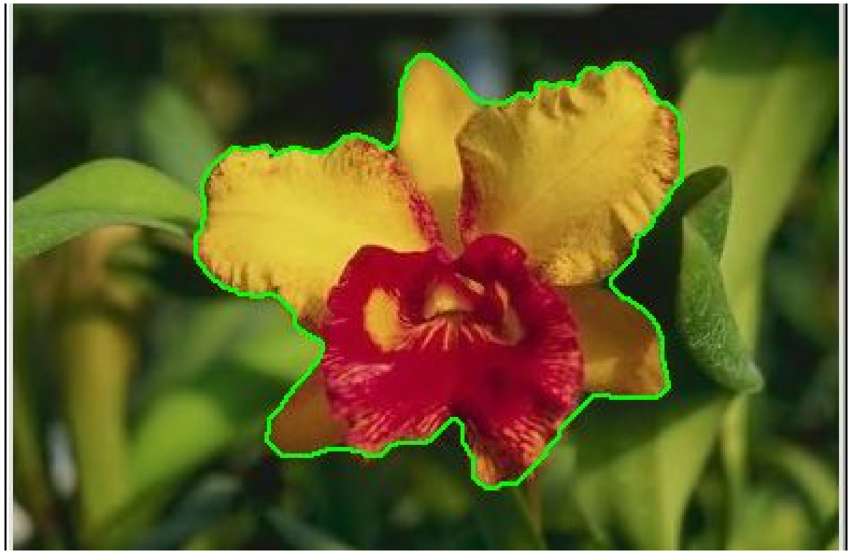






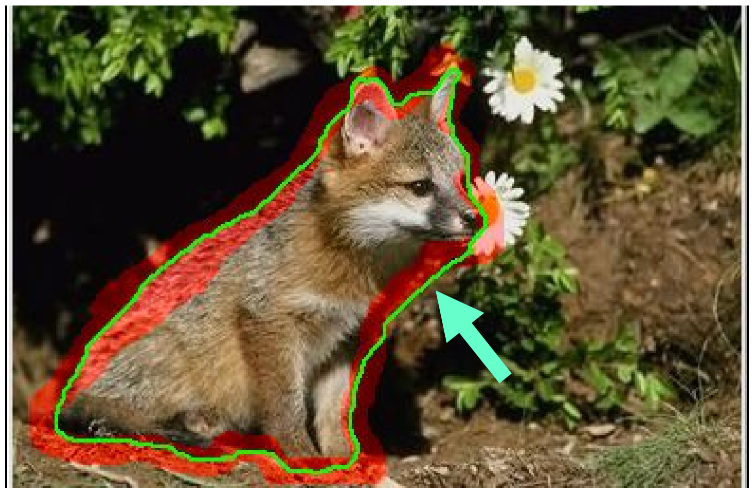
- Total computation time: not much more than a single search in the corridor area.

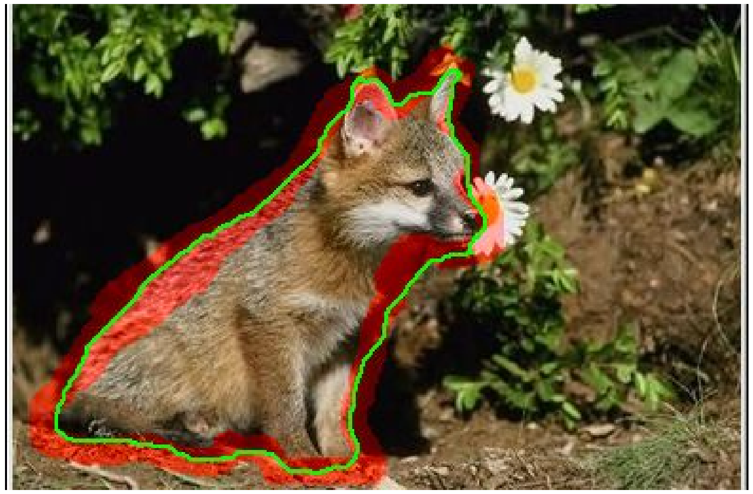
Results

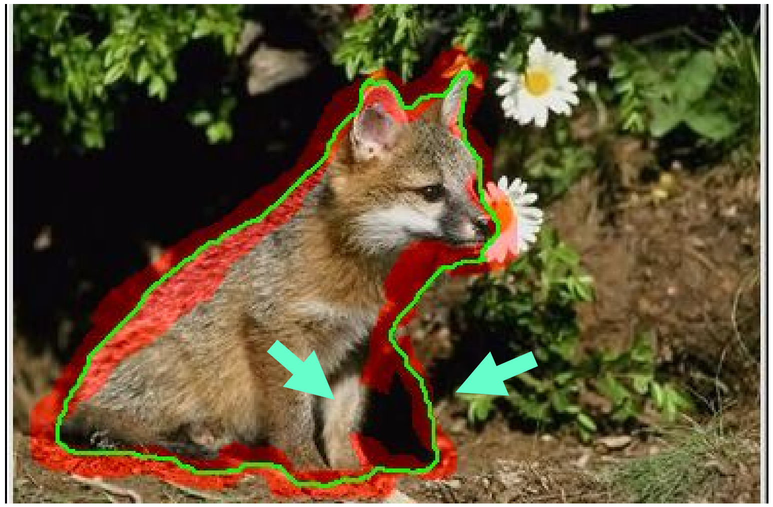
Editing for Gradual Improvements

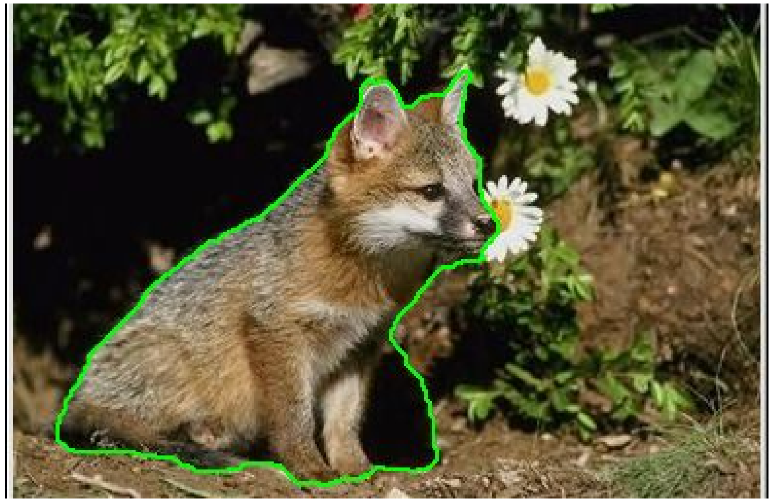
Improve Segmentation / Change Object



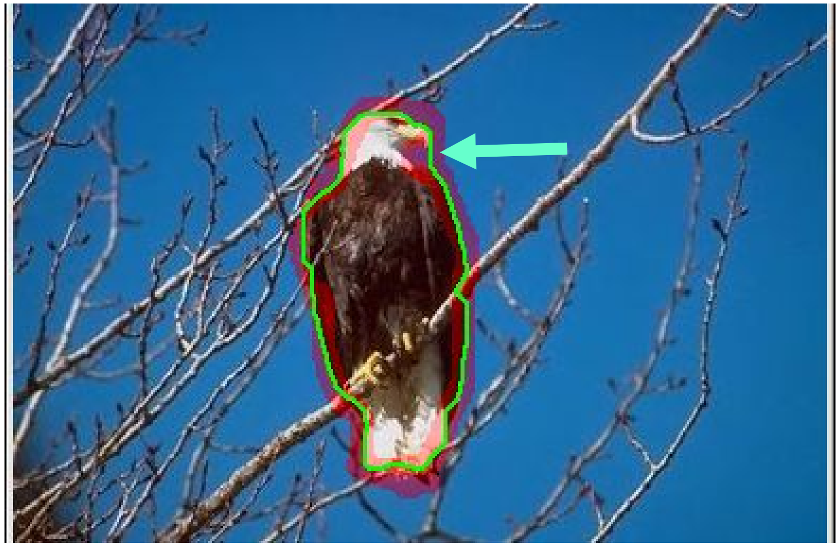
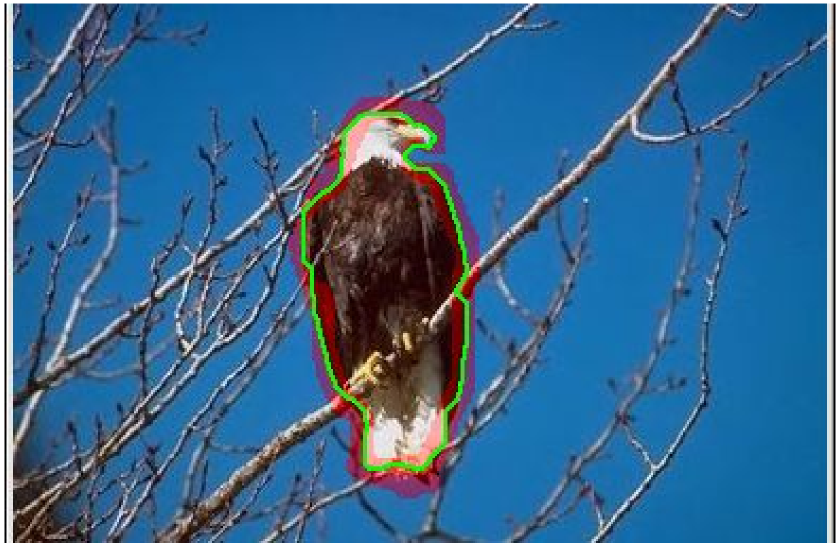
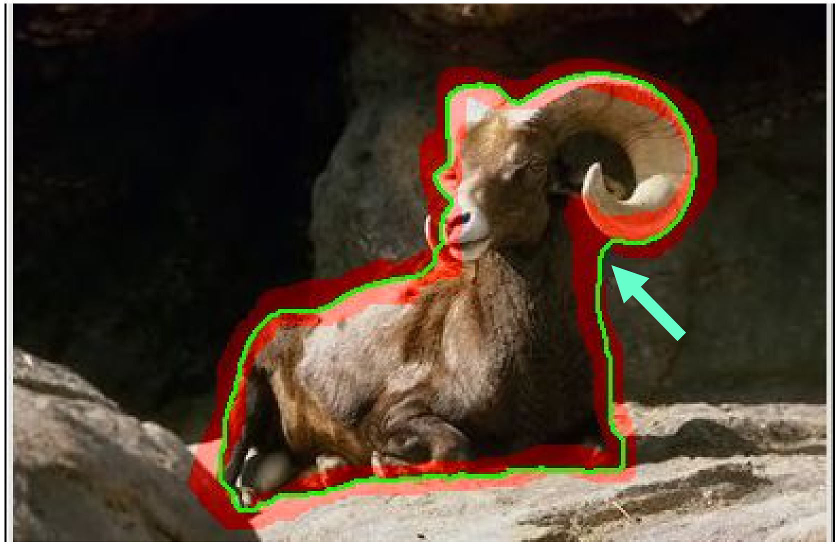
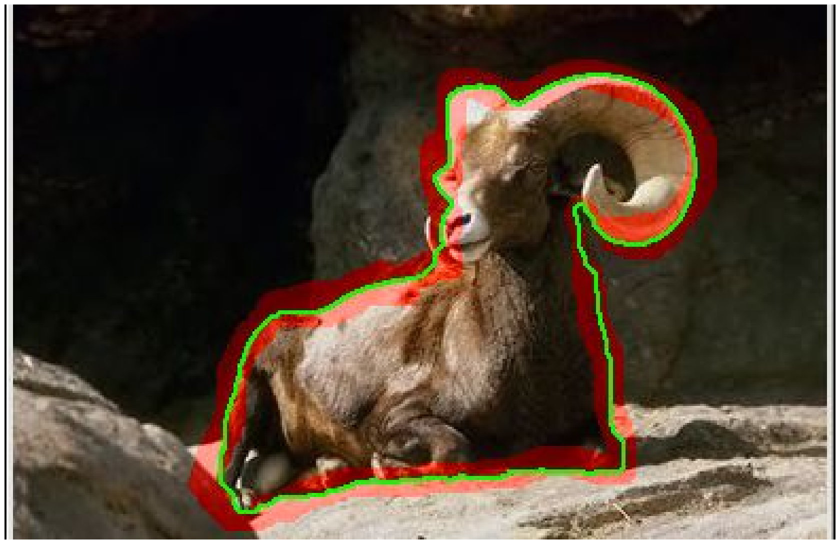


Change Object (include shadow)

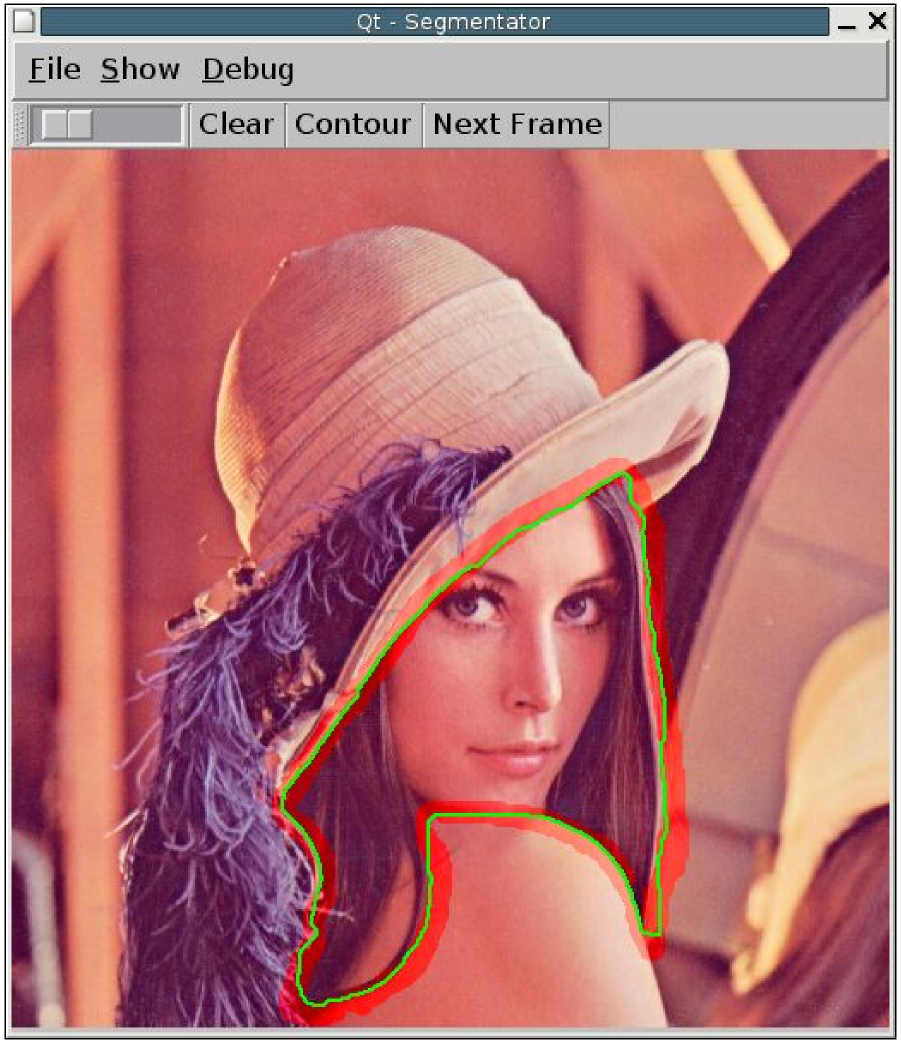


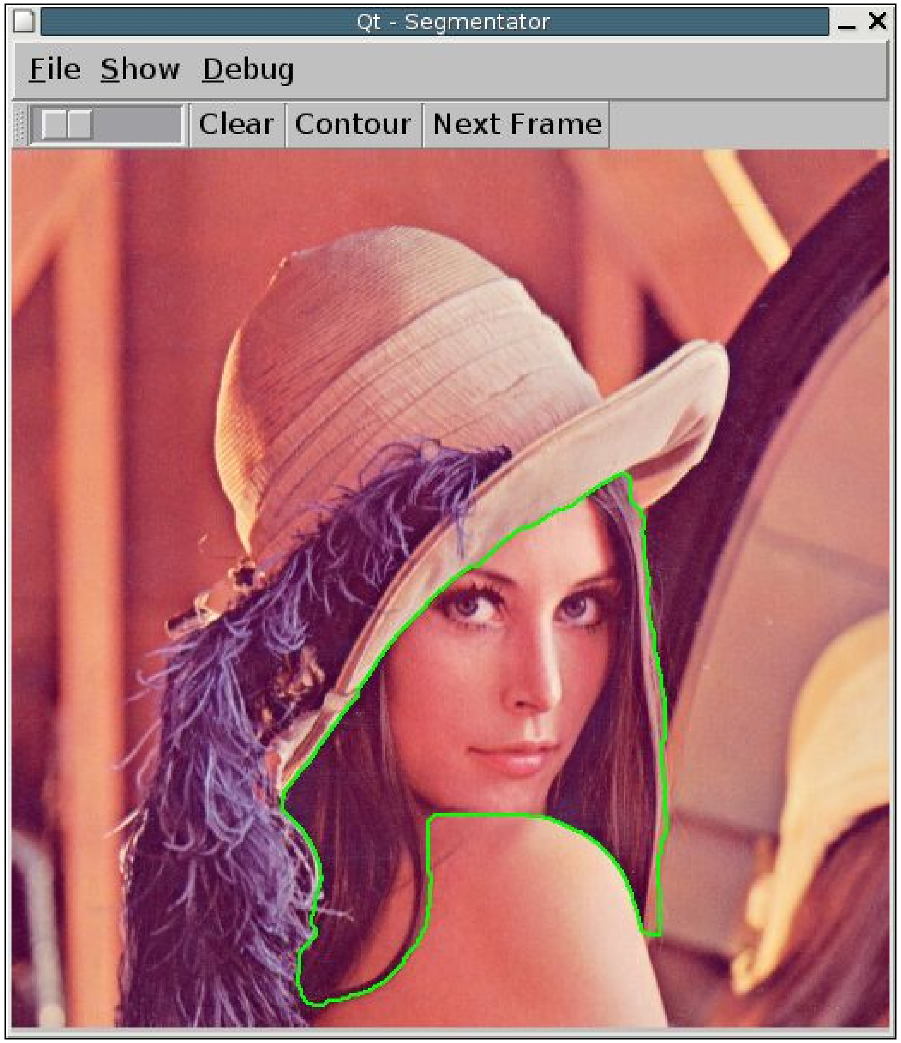


Typical Segmentation Errors: Shortcuts

Results: Lena





Conclusions

- We presented
  - Corridor Scissors as a new semi-automatic segmentation tool
    - New concept of user interaction.
    - Allows for easy gradual improvement of the segmentation.
  - A general algorithm to compute minimum-cost circular paths.
    - High computation speed, allowing interactive use on high-resolution images.
- The minimum-cost circular path algorithm
  - Computation time is approx. the same as a single Dijkstra algorithm run.
  - Easy implementation (combination of a few Dijkstra runs).
- Extensions
  - Straightforward extension to video sequences
    - Generate corridor for next image by dilating the contour of the last frame.