



**Computer Science @ RIT**  
B. Thomas Golisano College of Computing & Information Sciences

# Intro to Computer Vision: Automatic Image Resizing

Kurt A. Kluever (kurt@klover.com)  
Department of Computer Science  
Rochester Institute of Technology

- Devices with limited screen real estate
  - Mobile phones
  - PDAs
  - Hand-held PCs
- Devices with limited bandwidth
  - Want to only transfer important parts of the image
- Resizing web pages
  - When shrinking a web page, images remain static and fixed, while text is forced to wrap
  - When enlarging a web page, it might be beneficial to enlarge the image as well as unwrapping the text



- **Definition:**
  - the removal of unwanted parts of an image to reduce the overall size of the image
- **Reason for doing so:**
  - Improve framing (rule of thirds)
  - Focus the users attention to a particular part of the image
  - Change the aspect ratio
  - Make the image “fit” onto a display
  - Improve overall composition of image (photo)
  - Remove irrelevant/unwanted image data (example on next slide)

Original image (with questionable content):



Cropped image (sans beer bottles):



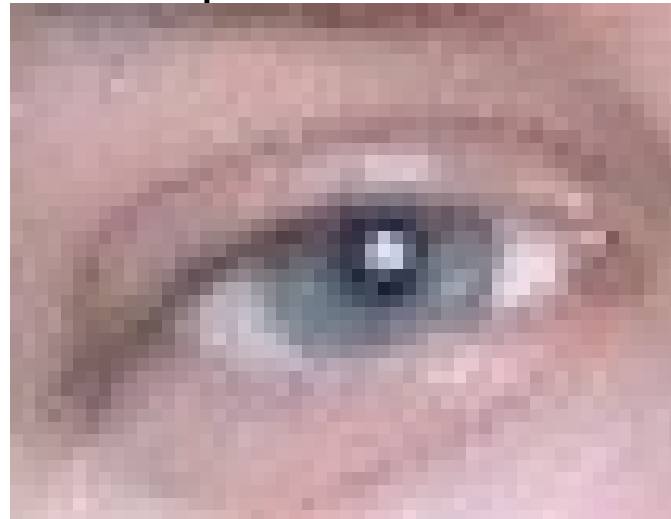
- Scaling is the process of uniformly resizing an image without adding or removing parts of the image
- Methods:
  - Nearest Neighbor Interpolation
  - Bilinear Interpolation
  - Bicubic Interpolation

- Example:

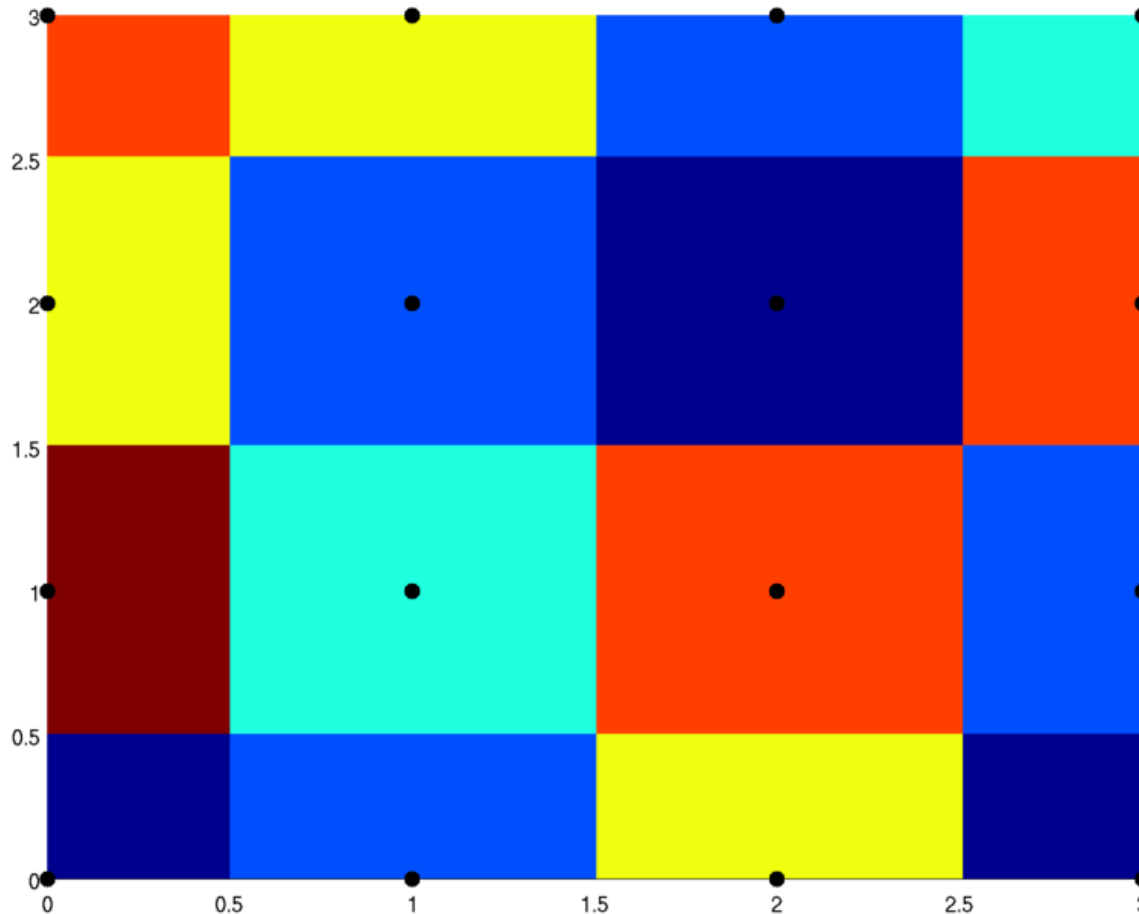
- Original



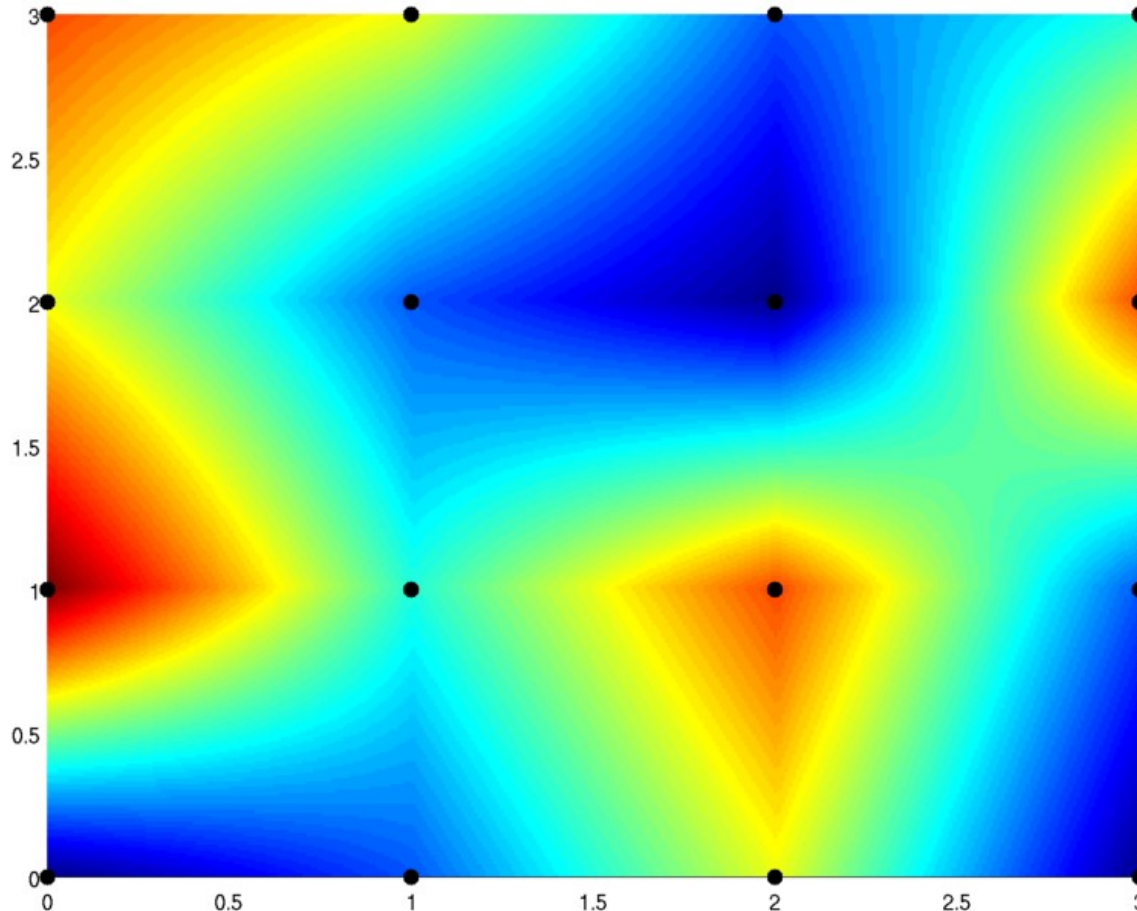
- Scaled Up:



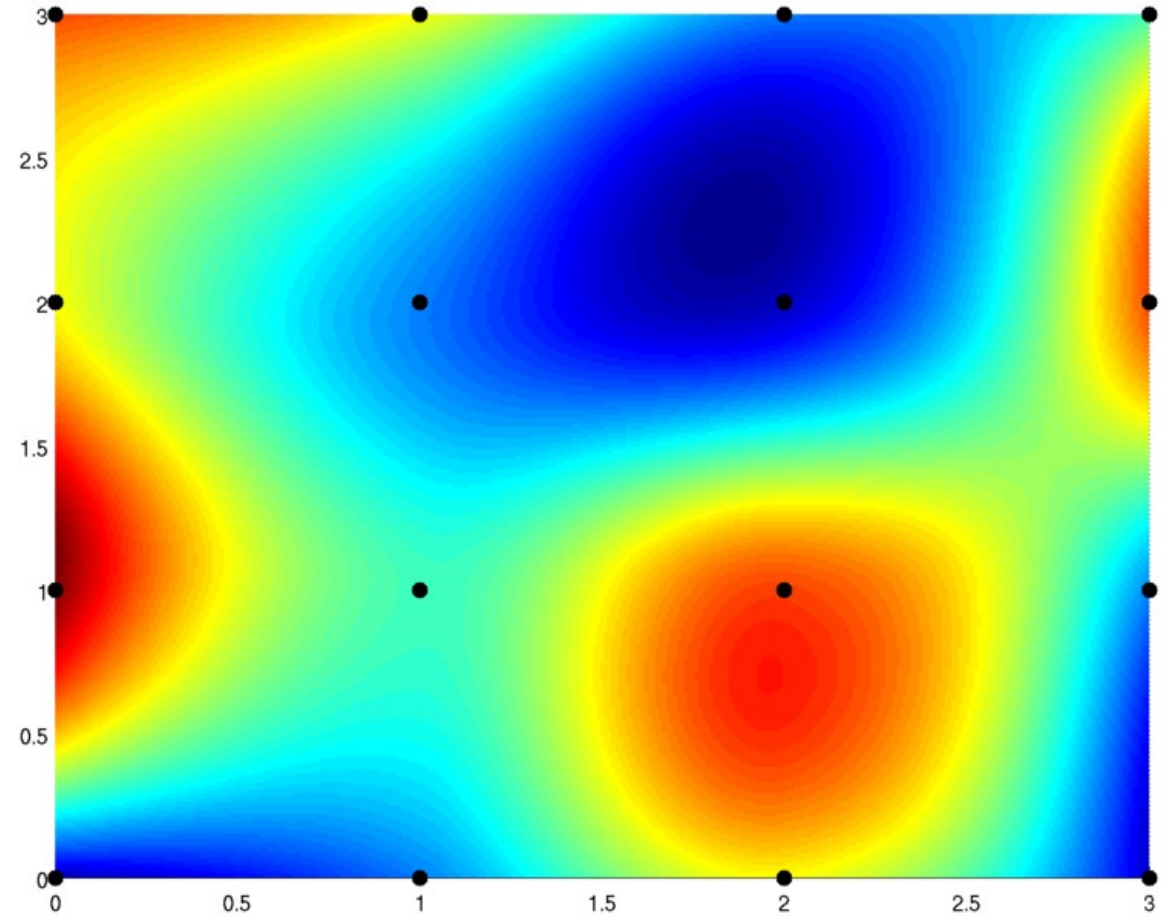
- Idea:
  - replace every pixel with four pixels of the same value
- Disadvantages:
  - image becomes “jaggy” (especially on diag. lines)
- Example:



- Idea:
  - perform linear interpolation in both x + y directions
- Disadvantages:
  - details become softened; still a bit jaggy
- Example:



- Idea:
  - perform interpolation; result and first derivative must be continuous
- Disadvantages:
  - still a uniform image resizing method
- Example:



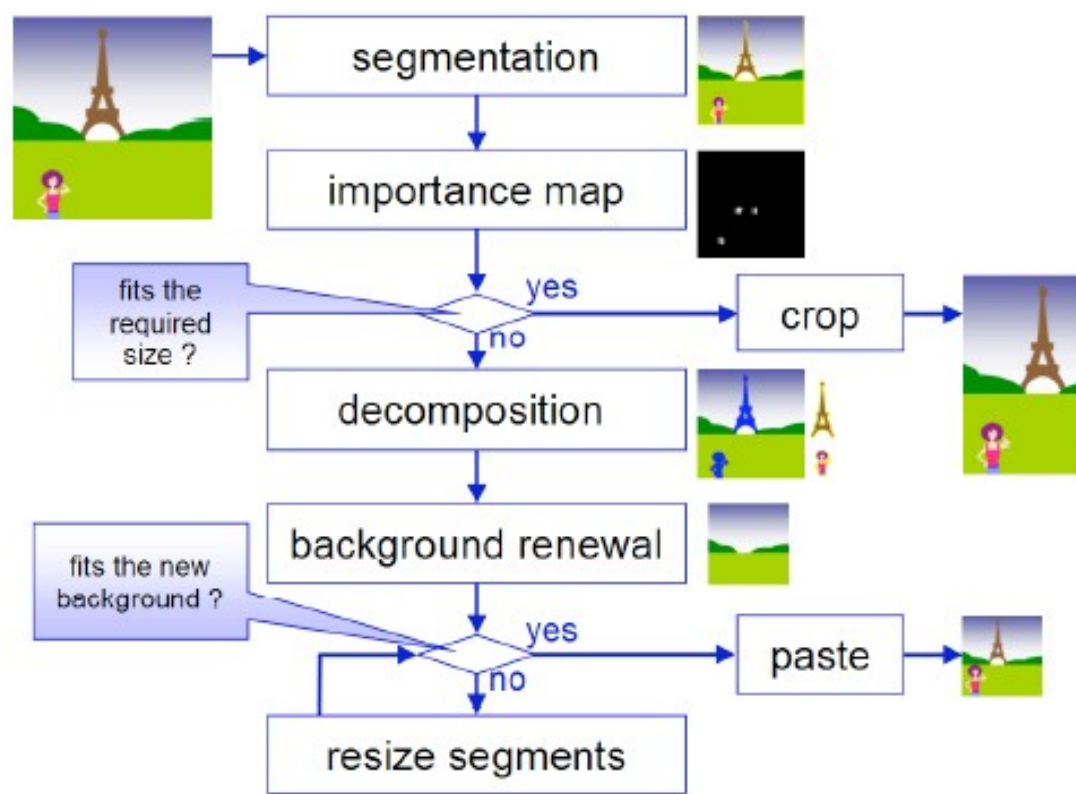
- Scaling and cropping seem to work, right? Yep, but...
- How do we automate this process?
  - Cropping only ROIs [4] (2003)
    - Bongwon Suh, Haibin Ling, Benjamin Bederson, and David Jacobs of the University of Maryland
  - Exaggerating ROIs [3] (2005)
    - Feng Liu and Michael Gleicher of University of Wisconsin, Madison
  - Cut and paste ROIs [2] (2005)
    - Vidya Setlur and Bruce Gooch of Northwestern University
    - Ramesh Raskar of Mitsubishi Electronic Research Labs
    - Saeko Takagi of Wakayama University
    - Michael Gleicher of University of Wisconsin, Madison
  - Seam carving [1] (2007)
    - Shai Avidan of Mitsubishi Electronic Research Labs
    - Ariel Shamir of The Interdisciplinary Center & MERL

- Idea:
  - Identify key regions of importance (ROIs) based on either saliency map or face detection and crop those
- Saliency Map:
  - Detect salient portions via saliency map (originally developed by Itti and Koch) that is based on the human visual system
  - Based on low-level features (independent of semantic info)
  - Find the smallest rectangle containing a fixed fraction of saliency
    - Threshold is dynamically calculated (point of diminishing returns)
  - First attempt: brute force (exhaust search of sub-rectangles)
  - Final solution: greedy algorithm (only consider only rectangles that include the peaks of saliency)
- Automatic face detection
  - Use CMU's face detection algorithm (<http://demo.pittpatt.com/>)
  - Draw the smallest bounding rectangle possible around faces

- Idea:
  - Use a non-linear fisheye-view warp that emphasizes ROIs while shrinking the rest of the image
  - Identify ROIs via low-level salience and high-level object recognition (see last slide)
- Advantages:
  - Does not discard any data from the original image (only de-emphasizes them)
  - Draws the users attention to the correct part of the image (ROI)
- Disadvantages:
  - Results look warped (by design)!!! (fat head)



- Idea:
  - Segment image into regions, identify the ROIs, cut the ROIs, fill in the holes created by the ROIs, resize the new image, paste the ROIs back onto the image



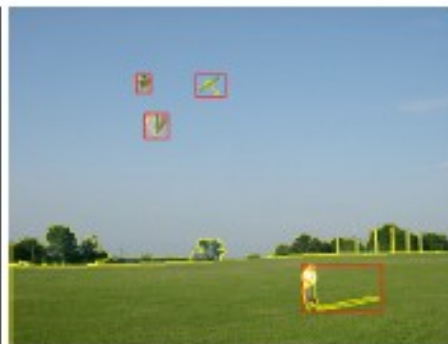
- How do we identify regions of importance (ROIs)?
  - Compute an importance map based on an attention model
  - Attention model:
    - Recognizable objects (faces)
    - Regions that attract the low-level visual system
      - Use measures of visual salience
      - Based on color, intensity, and edge orientation
      - Features towards center of image get preference
    - Easily extendible if desired
    - Can accommodate user specified areas of importance
  - Use a greedy algorithm to find best possible ROI
    - Identify a candidate ROI
    - Grow the ROI using a clustering algorithm
- Background growing
  - Replicate surrounding textures to fill in the gaps
- Paste ROIs onto background (shrinking if necessary)

Original

Saliency Map

Locate ROIs

Segment ROIs



Background

Retargeted

Cropped

Scaled

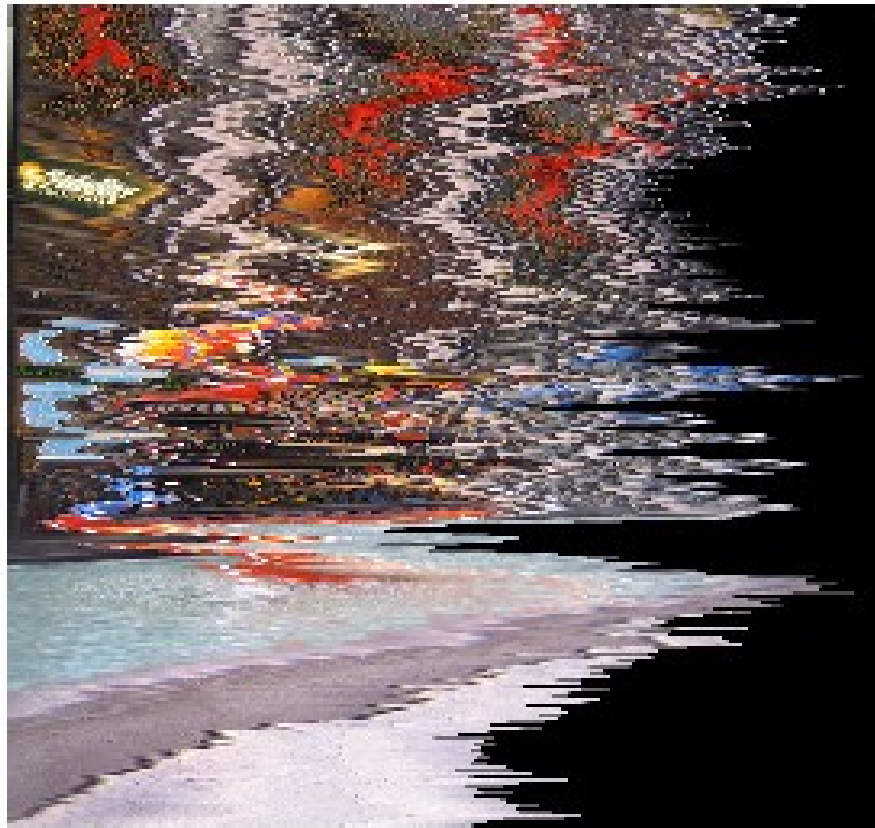
Which one looks the best? Retargeted, Cropped, or Scaled?

- Idea:
  - Define an energy function for computing areas of low energy
  - Remove unnoticeable pixels that blend in with their surroundings
  - Side effect: by removing pixels of low energy, the overall energy of the image will go up (not necessarily a bad thing)
- How do we figure out which pixels to remove?
  - Energy function



- Examined several possibilities
  - $L_1$  and  $L_2$ -norm of the gradient
  - Saliency measure [Itti et al. 1999] (similar to Crop Only ROIs)
  - Harris-corners [Harris and Stephens 1998]
  - Eye gaze measurement [DeCarlo and Santella 2002]
  - Face detectors (CMU's example used in [1])
  - $e_1$  error
  - Entropy over a  $9 \times 9$  window (then add  $e_1$  error to it)
  - Segment (then apply  $e_1$  error norm) [Christoudias et al. 2002]
  - Histogram of Gradients ( $e_{\text{HOG}}$ ) [Dalal and Triggs 2005]
- No single energy function works best in all cases
  - All perform OK on most images
  - $e_1$  or  $e_{\text{HOG}}$  seem to perform best

- Remove pixels with minimum energy in ascending order
  - BAD: destroys the rectangular size of the image (minimum energy pixels are not uniformly distributed among rows)



- Remove an equal number of low energy pixels from every row
  - BAD: maintains rectangular size, but creates a zigzag warp effect on content in the image



- Remove the column of pixels with the lowest energy
  - BAD: maintains rectangular size, but creates a jagged edges in the resulting image



- Remove a seam of pixels with the lowest energy
  - Seam: an 8-connected path of pixels in the image from the top to the bottom, containing one pixel in each row of the image



**BINGO!**

- Now that we have defined our energy function and how to apply it, how can we effectively compute which seam to remove?
  - Dynamic programming:
    - Overlapping subproblem: Can the problem be broken down into subproblems which are reused several times?
    - Optimal substructure: Can the optimal solution can be constructed efficiently from optimal solutions to its subproblems?
  - $M(i,j) = e(i,j) + \min(M(i-1, j-1), M(i-1, j), M(i, j-1))$ 
    - The minimum value of the last row in M will be the end of the seam
    - Start at this value and traverse upward, moving to the cell with the lowest value (either up & left, up, or up & right)
- Remove the pixels located in this seam and shift all other pixels to the left if necessary
- Rinse and repeat!

- Can insert artificial seams into an image to enlarge it
- Process
  - Compute the optimal vertical seam (of low energy)
  - Duplicate the pixels of the seam by averaging them with their left and right neighbors
  - Rinse and repeat
- Can create artifacts if enlarging the image significantly
  - Problem: We don't want to keep selecting the same seam to duplicate
  - Solution: When enlarging by  $k$  pixels, select the  $k$  seams with the lowest energy and duplicate these
  - Problem: If we have an image that is  $k$  pixels wide and we want to enlarge it by  $k$  pixels, all seams will be selected by the step above and we will in effect scale the image by 50%
  - Solution: Perform the enlarging in 2 steps

- Mark an area of pixels to remove (set energy to zero)
- Mark an area of pixels to protect (set energy to infinity)
- Perform seam removal until all of the marked pixels have been removed from the image



- Limitations
  - Automatic process does not work on all images
    - Can manually define areas of high or low importance
  - Images with too much content are impossible to retarget
    - Images with no “un-important” areas are impossible
    - If the amount of content is condensed, you must scale the image or crop something out to resize the image
  - Images with content that is laid out in certain ways are hard
    - If you are trying to reduce the width of an image (want to remove vertical seams), and image contains an object which spans the entire image (horizontally), the object will get distorted
    - Impossible to bypass important parts
- Future Work
  - Resizing of video (combine scaling + retargeting for robust multi-sized images)
  - Find a way to combine horizontal and vertical seams

- [1] Avidan, S. and Shamir, A. 2007. Seam Carving for Content-Aware Image Resizing. In *ACM Transactions on Graphics, Vol 26, No 3, SIGGRAPH 2007*
- [2] Setlur, V., Takagi, S., Raskar, R., Gleicher, M., and Gooch, B. 2005. Automatic Image Retargeting. In *In the Mobile and Ubiquitous Multimedia (MUM)*, ACM Press.
- [3] Liu, F., and Gleicher, M. 2005. Automatic Image Retargeting with Fisheye-View Warping. In *ACM UIST*, 153-162.
- [4] Suh, B., Ling, H., Bederson, B., and Jacobs, D. 2003. Automatic Thumbnail Cropping and its Effectiveness. In *ACM UIST*, 95-104.



**Computer Science @ RIT**  
B.Thomas Golisano College of Computing & Information Sciences

# Questions?

[kurt@klover.com](mailto:kurt@klover.com)