Real-time texture boundary detection

Ray Hidayat, Dr Richard Green

University of Canterbury, New Zealand

- Boundary detection is sometimes used as an essential first step to image interpretation

Original image

Global probability of boundary detector

What is a boundary?

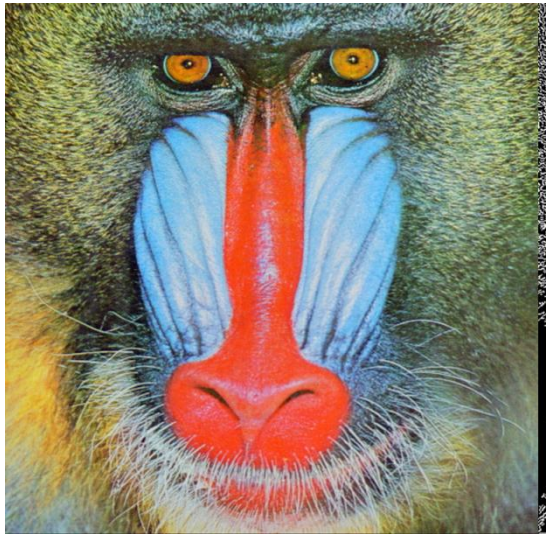
- A boundary separates different areas of an image

An image and its boundaries according to humans

- Boundary detectors find boundaries by finding **significant differences** between neighbouring areas of an image
- But, in the real world, this is not simple, because…

Texture

- Most images in the real world contain texture
- Texture is made up of repeating **variations**
- These variations make boundary detection difficult

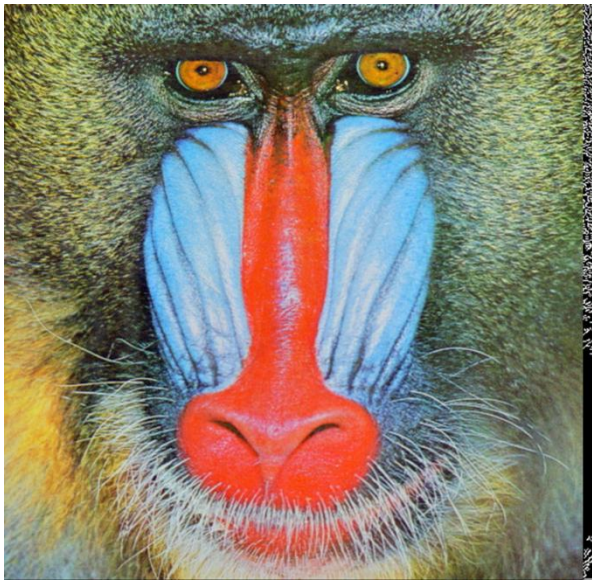


Original image

Canny edge detector
(does not consider texture information)

- A naïve boundary detector can easily confuse these repeating variations for significant differences

- Most images contain texture
- For a boundary detector to be good on most images, it **must** utilise texture information



Original image

Canny edge detector
(does not utilise texture)

Standard deviation ridge detector
(does utilise texture)

- The problem: using texture is slow!

Problem

- Most texture boundary detectors are too slow to be any use in realtime
  - Generally take more than ten seconds per image, some take three minutes or more
- The few texture boundary detectors that work in realtime are low quality
  - Second moment matrix
  - Surround suppression

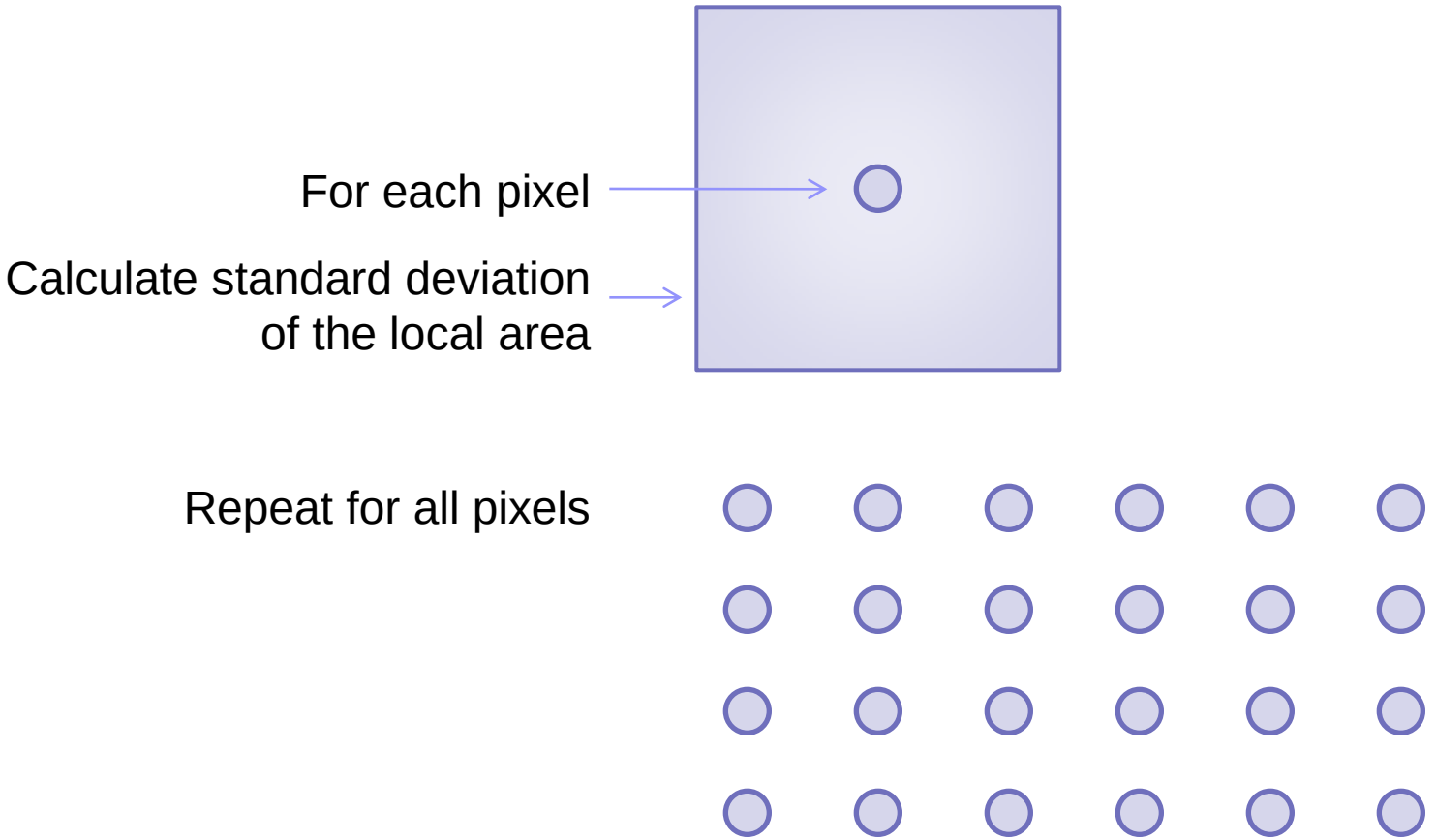
Proposed solution

- Our proposed solution: the standard deviation ridge detector
  - Can detect boundaries
  - Is able to ignore variations in texture
  - Can do this in realtime

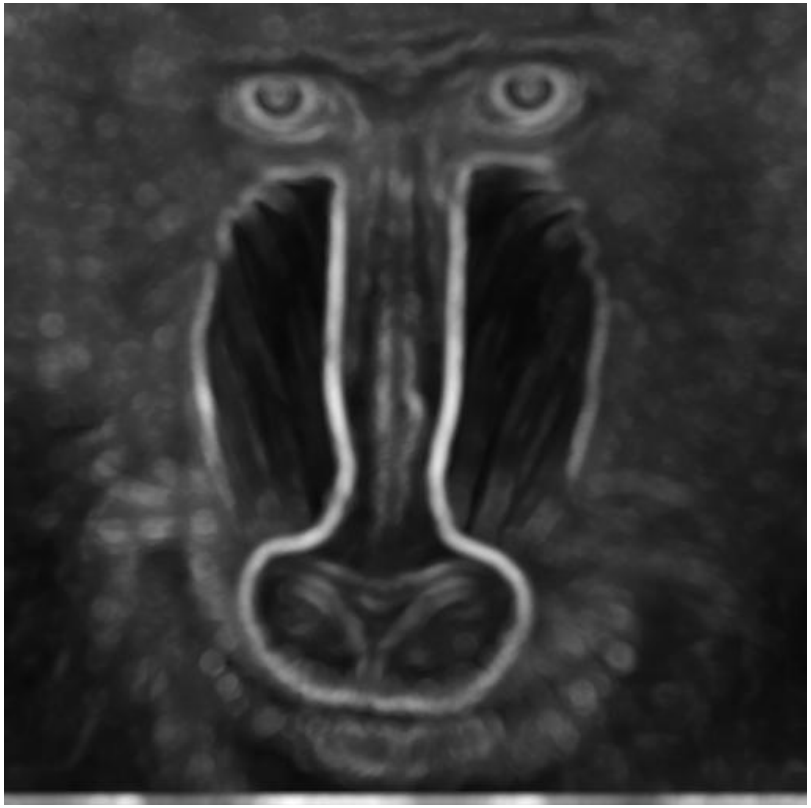
Algorithm

- Works by finding ridges in the standard deviation space
- Four steps:
  1. Standard deviation transform
  2. Gradient transform
  3. Ridge detection
  4. Boundary detection
- Biggest difficulty is, how can you ignore the variations within a texture in realtime?

Step 1: Standard deviation transform

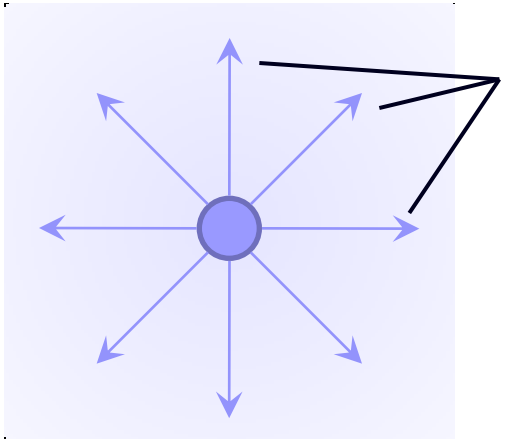


Step 1: Standard deviation transform



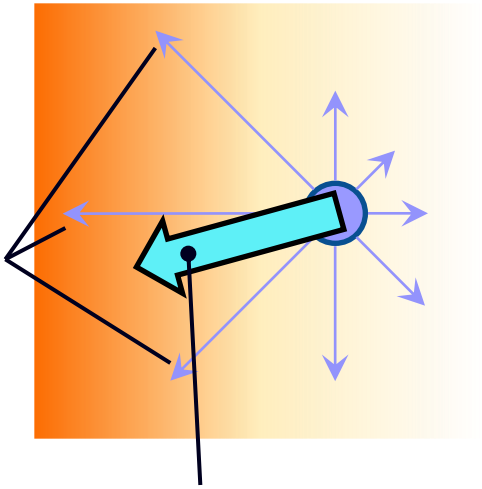
- Why do this?
- Reason 1: Standard deviation is approximately equal for different areas of the same texture
- Therefore, if the algorithm can detect this, it could suppress the variations in texture

Step 2: Gradient transform



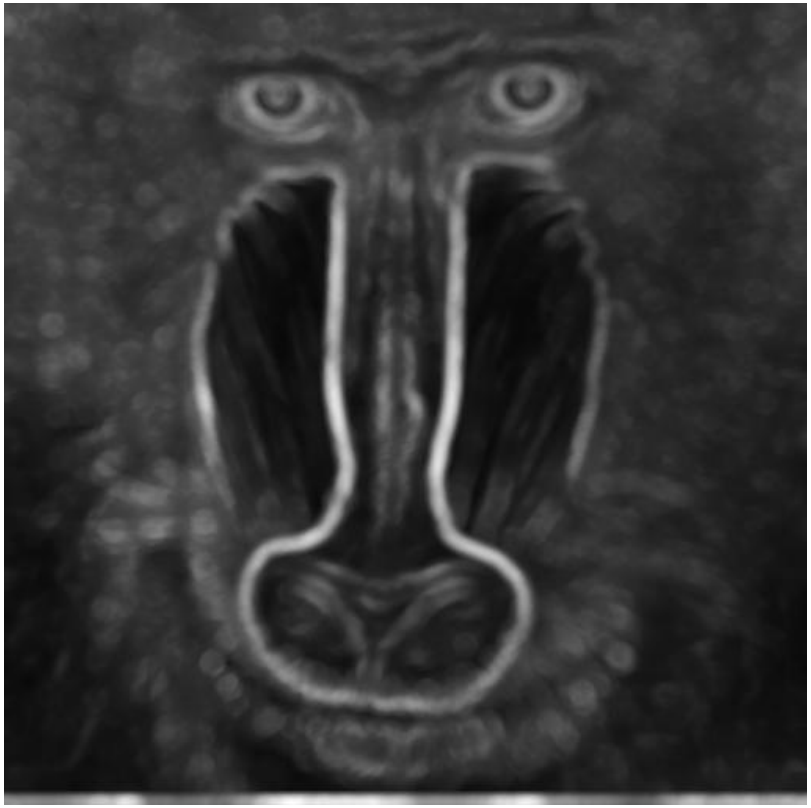
The surrounding pixels ‘pull’ the point of interest in their direction

Pixels with higher standard deviation have a stronger ‘pull’



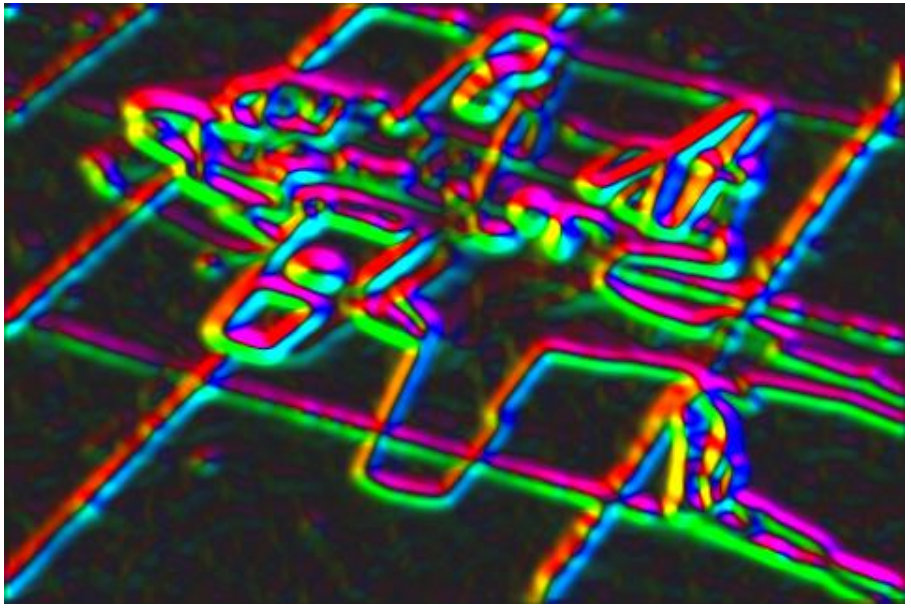
The overall gradient is calculated for each pixel

- Within the same texture, standard deviation tends to be equal
  - Therefore, within a texture, the forces will cancel out
  - This way, texture is eliminated from the algorithm



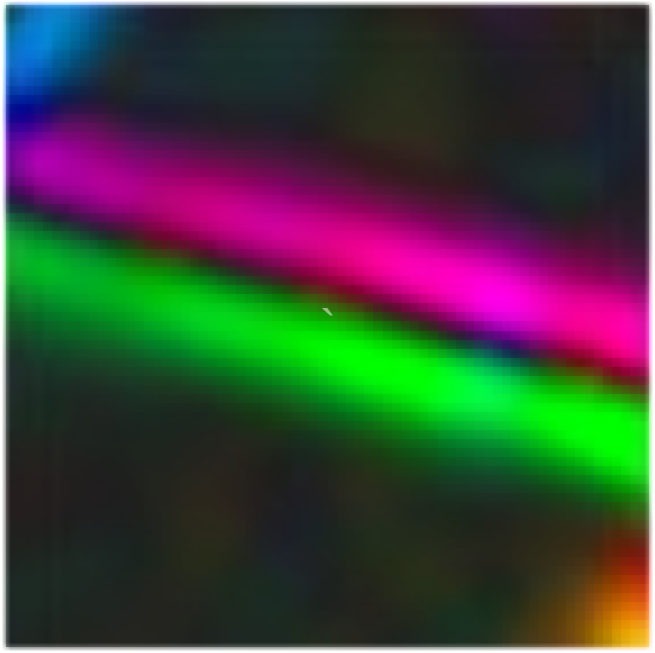
- Reason 2: Standard deviation peaks on boundaries
- Within texture:
  - SD = intra-class variation
- On boundaries:
  - SD = intra-class variation + inter-class variation
- Therefore, if the algorithm can detect this, it could detect boundaries

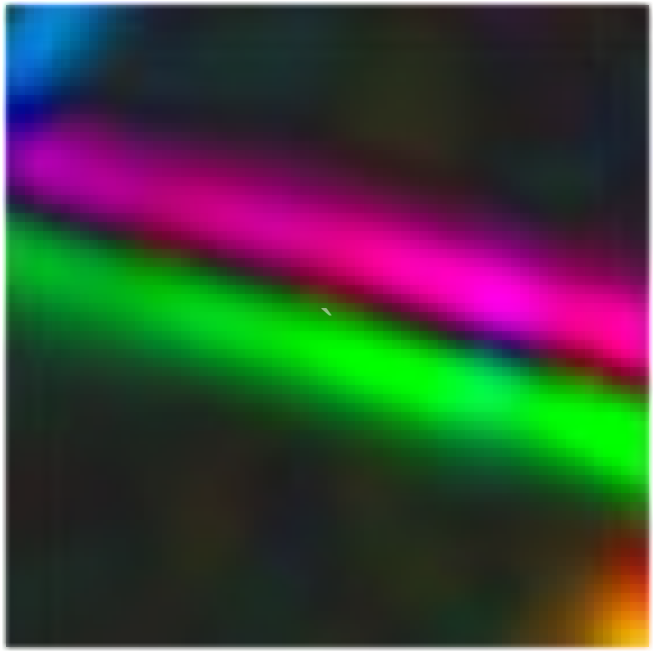
Step 2: Gradient transform



- In a one-dimensional function, a peak will have a positive gradient on one side, and a negative gradient on the other
- In two dimensions, gradients point towards the peaks in the standard deviation transform
- Must detect this pattern to detect the boundaries

Step 3: Ridge transform





- Use dot product to calculate how strongly the gradients point towards each other
- Compare gradients at particular predetermined offsets

Step 3: Ridge transform

- Do this with 0°, 45°, 90° and 135° offsets
- Take the strongest response from all of the offsets
- The result is the ridge transform

Original image

Ridge transform

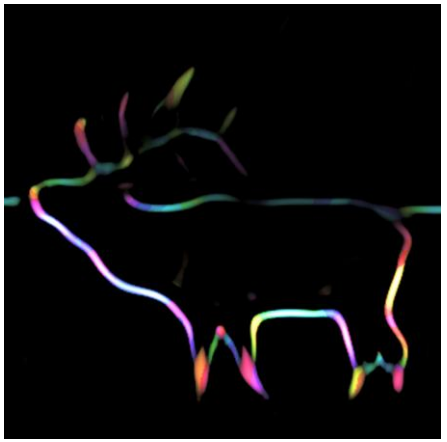
Step 4: Boundary detection

Ridge transform

-

Gradient transform

=



Final result

- Finding ridges by comparing offset gradient images will mislocate boundaries because it only compares two points at a time
- A peak must have a negative gradient on one side, and a positive gradient on the other, so this is a good way to improve the boundary detection

Results

- Is able to execute at 43.29 frames per second
  - This is orders of magnitude faster than most non-realtime texture boundary detectors, which take more than 10 seconds per image
  - Can run up to 125 frames per second for a tradeoff in quality
  - This supports our claim that this is a realtime algorithm
- Achieves 0.62 on the Berkeley segmentation benchmark
  - The Berkeley benchmark is a publicly available system which objectively ranks the world's best boundary detectors
  - Our proposed algorithm outperforms the best realtime texture boundary detector, the second moment matrix (0.57)
  - This supports our claim that the algorithm is a texture boundary detector

Conclusion

- The standard deviation ridge detector is able to detect boundaries using texture information.
- It is orders of magnitude **faster** than non-realtime texture boundary detectors
- It produces **higher quality results** compared to all existing realtime texture boundary detectors.
- Our objective scientific measurements fully support these claims
- Because boundary detection is so important, these positive results could allow improvements to many realtime computer vision applications