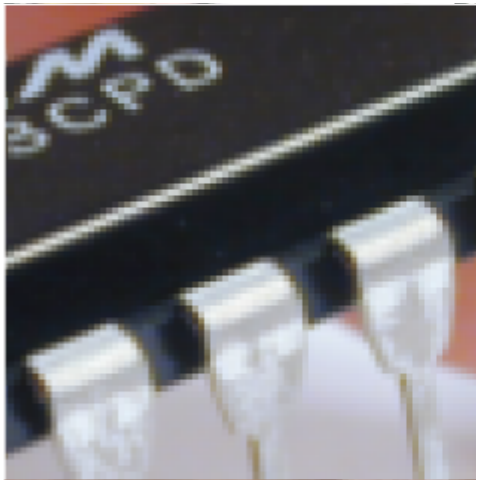
Superresolution and their Fundamental Limitations

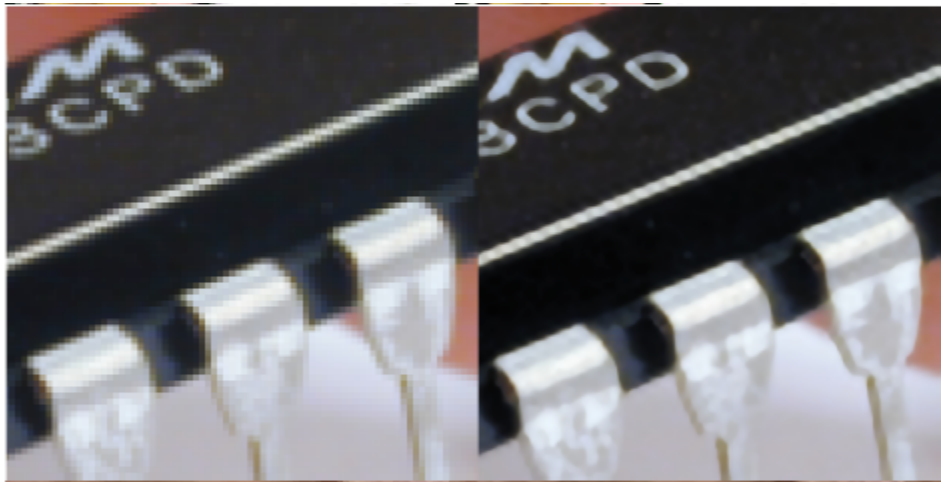
Neil Del Gallego
Oliver Bob Urag
Edward Patrick Tighe

Topic Outline

1. What is superresolution?
2. Multiple-image SR
3. Single-image SR
4. Applied SR in Android platform
5. Results and future work

Superresolution?


3CPO



Super Resolution

Superresolution

- The enhancement of a low resolution image to a high or “super” resolution image
- Two types of SR techniques
  - Multiple Image SR
  - Single Image SR

Multiple Image SR

- Involves using multiple low resolution images to create a single high resolution image

More sample of a scene -> More chance to increase resolution

Multiple Image SR

Image Registration

- The captured image sequence is combined and aligned using transformations that considers multiple factors to yield a common reference frame

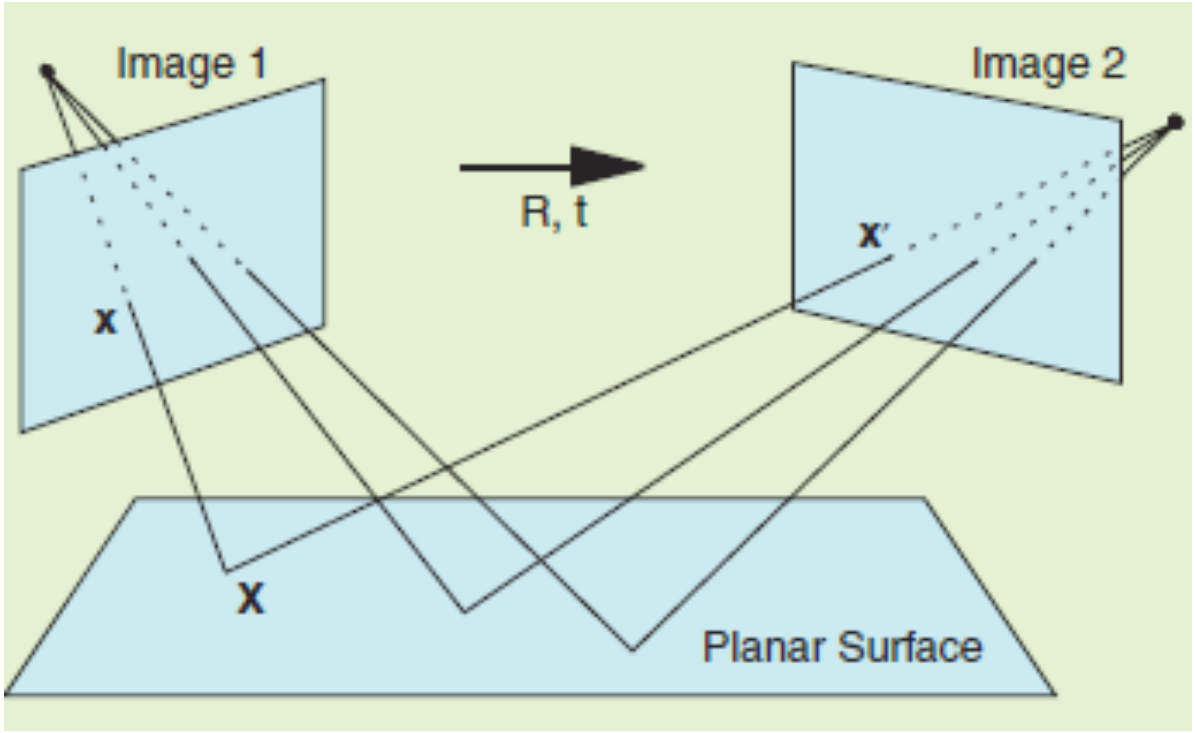
Image Registration

Given ***N*** different images of a scene,
for each image point in ***k*** view (***k*** $\in$ ***N)***,
find the image point in ***k + 1*** view
which has the same properties of ***k***
(or same point in the scene)

Image Registration

- Three kinds of registration process:
  - Geometric
    - considers displacements of images in its coordinates
  - Feature-based
    - considers interest points (corners)
  - Photometric
    - Considers change in illumination, intensity variations, color composition, RGB channel, etc.

Image Registration



Multiple Image SR

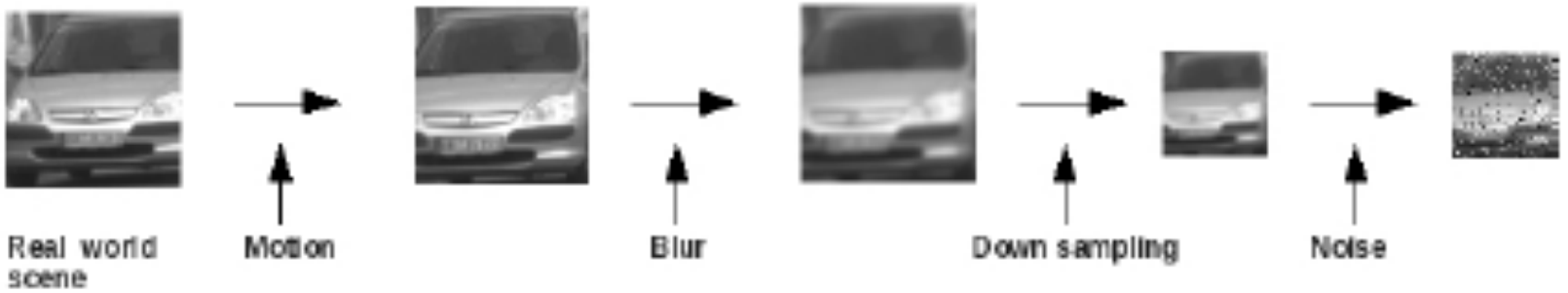
SR Estimation

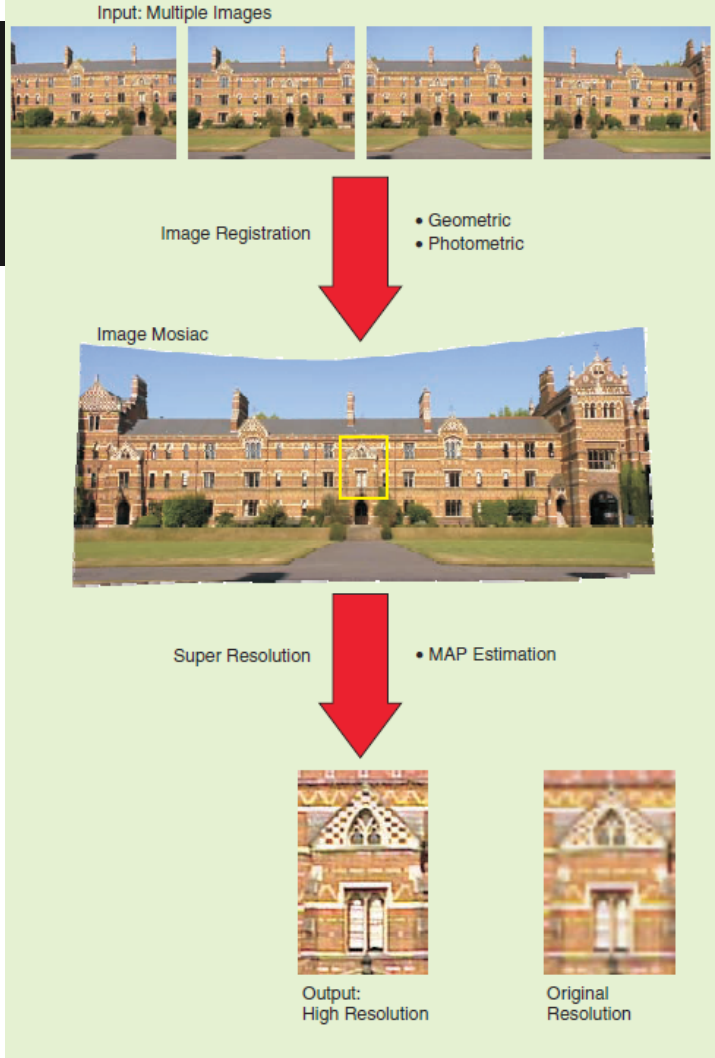
- SR estimation is then performed to model the image degradation caused in the formation process. Important details are then propagated to the higher resolution image.

SR Estimation

- There are 4 factors that affect an image:
  - Down-sampling
  - Blurring
  - Warping
  - Noise

SR Estimation




Input: Multiple Images
Image Registration
• Geometric
• Photometric
Image Mosiac
Super Resolution
• MAP Estimation
Output:
High Resolution
Original
Resolution

SR Estimation in Detail

$$e^{(n)} = \sqrt{\sum_{k} \sum_{(x,y)} (g_k(x,y) - g_k^{(n)}(x,y))^2}.$$

- From a candidate *H* image, attempt to “guess” its possible set of *L* images.
- Compute difference Gk(original *L* images) vs Gk^n (*L* images generated from SR). This is *e* or the error function.
- If *e* has reached threshold, output *H* image. Otherwise, perform refinements to *H* such that *e* is reduced on next iteration.
- In a perfect world, *e* = 0. Original *L* images are exactly the same as the generated *L* images. *H* is the perfect superres image from *L set.*

Example

Oppo’s Super Zoom feature

- Take 4 of the 10 required input images
- These 4 images are considered for ML estimation
- Given a 13 MP image, output would be a 50 MP image




Limitations

- Although the edges are smoothened and the noise is reduced significantly, the finer details are lost


SHARE
your
JOURNEY
SHARE
your
JOURNEY

Limitations

- According to Baker, K. (2000), a set of low resolution images to reconstruct an SR image is no longer applicable beyond a point wherein the magnification factor is great as it is now affected by a wide range of factors

Single Image SR

Use of a single image as input to produce the high resolution image

Single Image SR

- From a High Resolution image a Low Resolution image is produced by different factors (downscaling, compression, blurring and noise). Shan, Li, Jia and Tang (2008)

- Natural images tend to contain repetitive visual content. *Bagon, Glasner and Irani (2009)*

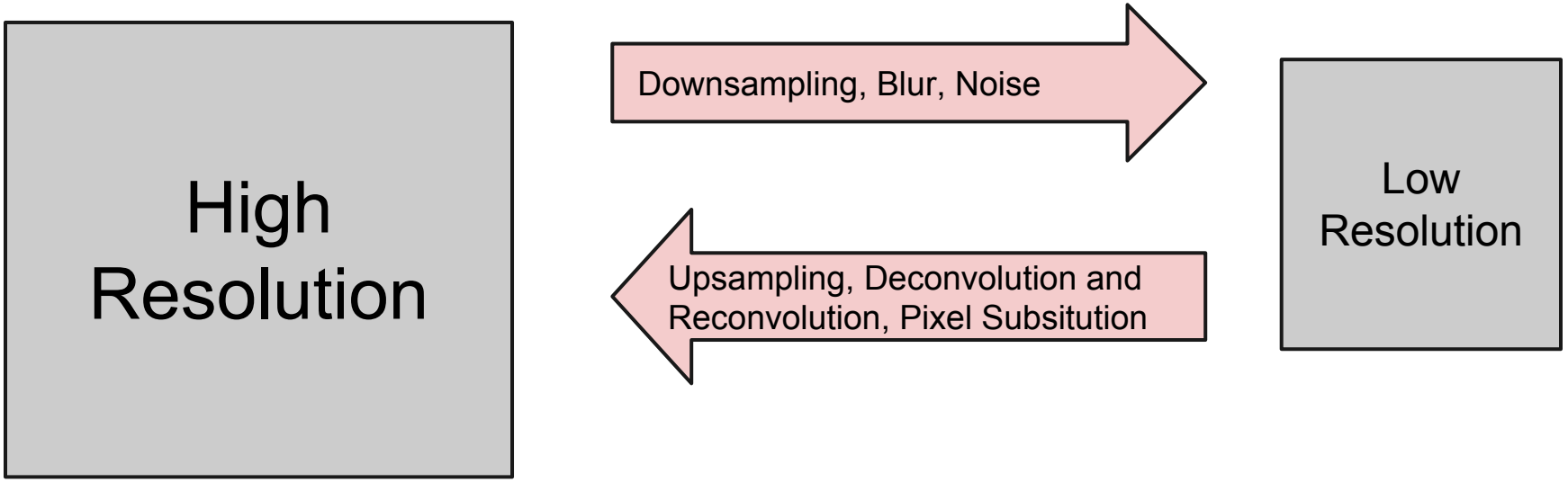
Single Image SR

- **Feedback Control** - Given a low resolution image, model the factors that produced the low resolution image and perform image processing to reverse the factors revealing the original high resolution image.

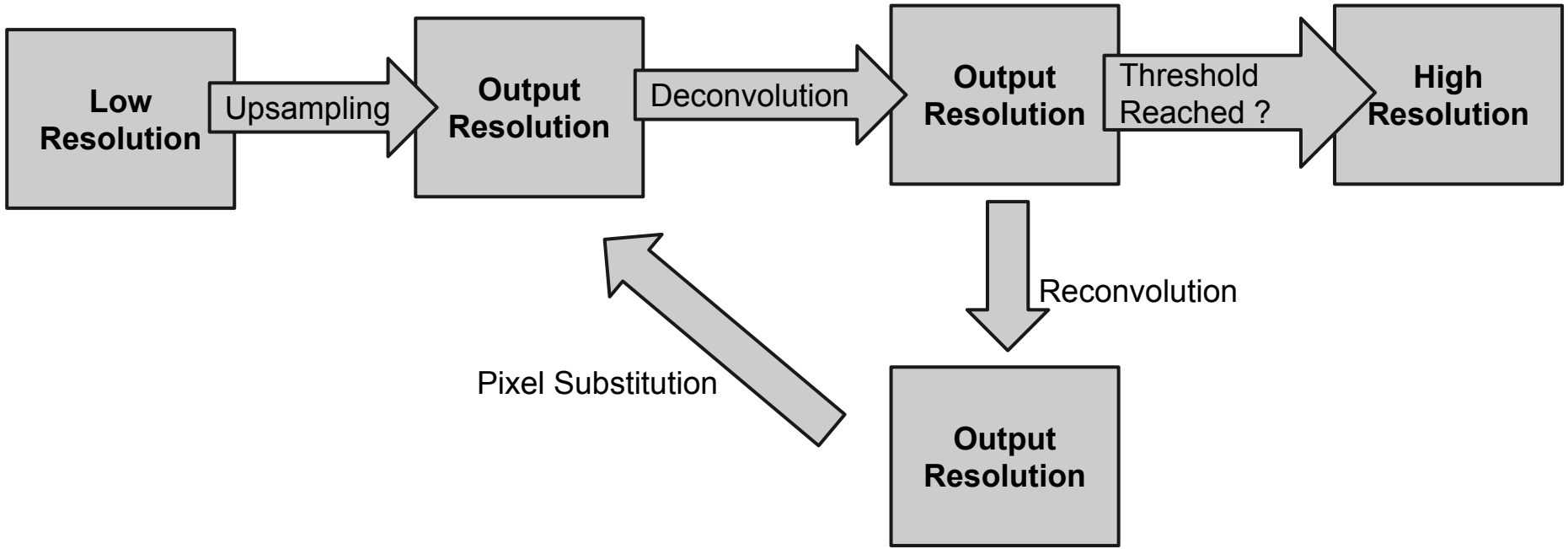
- **Patch Redundancy** - exploit the repetitive visual content of a low resolution image “patch” to produce the high resolution image.

- **Edge Statistics** - Utilize observation that real-world images exhibit pixel differences that depend on their distance from an edge. Highlight these edges so contours are preserved.

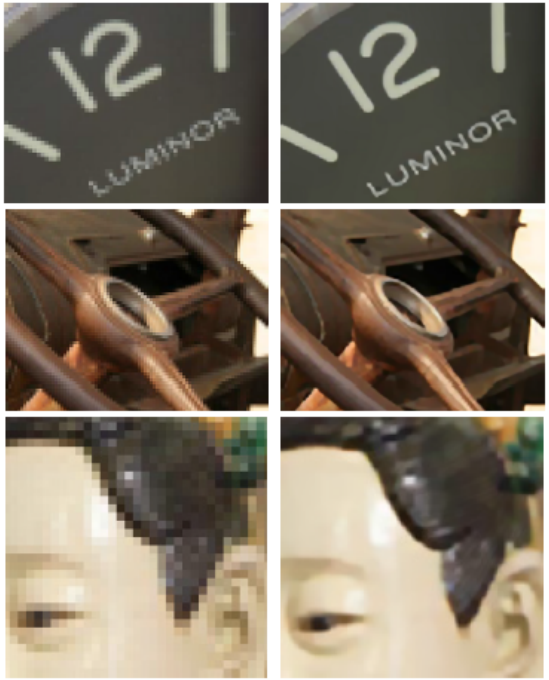
Single Image SR



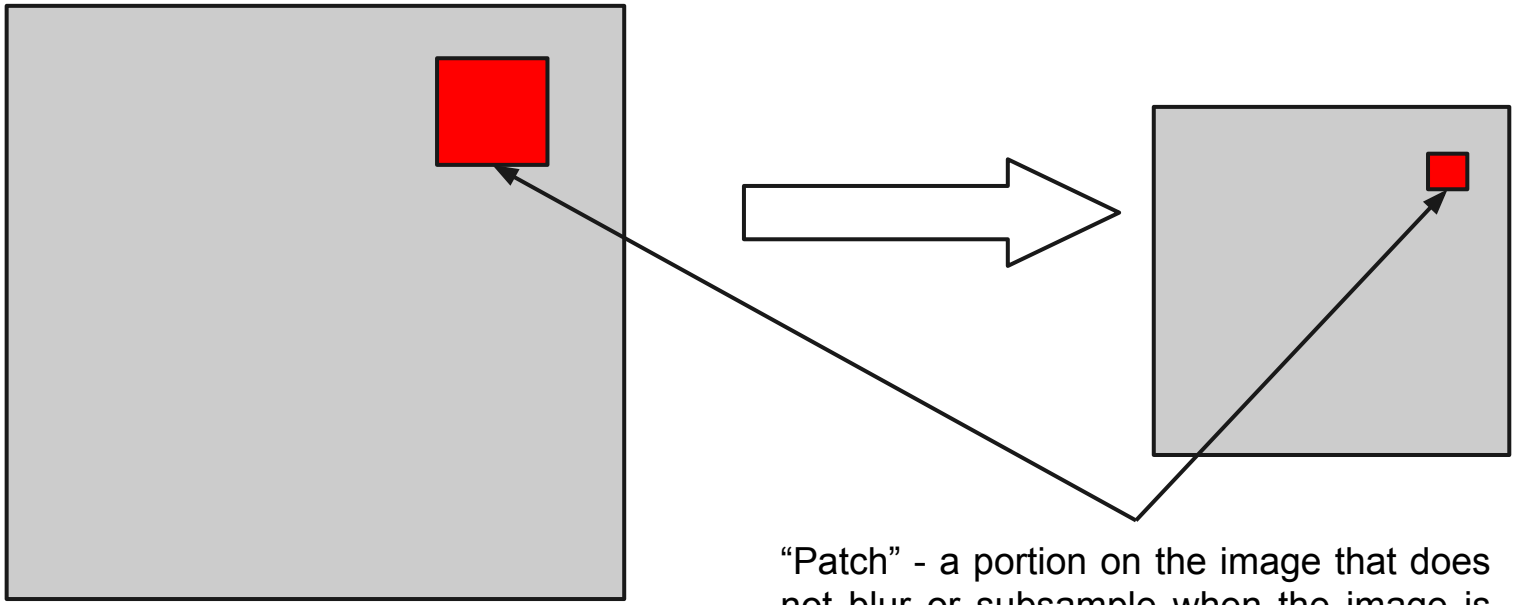
Single Image SR



Single Image SR

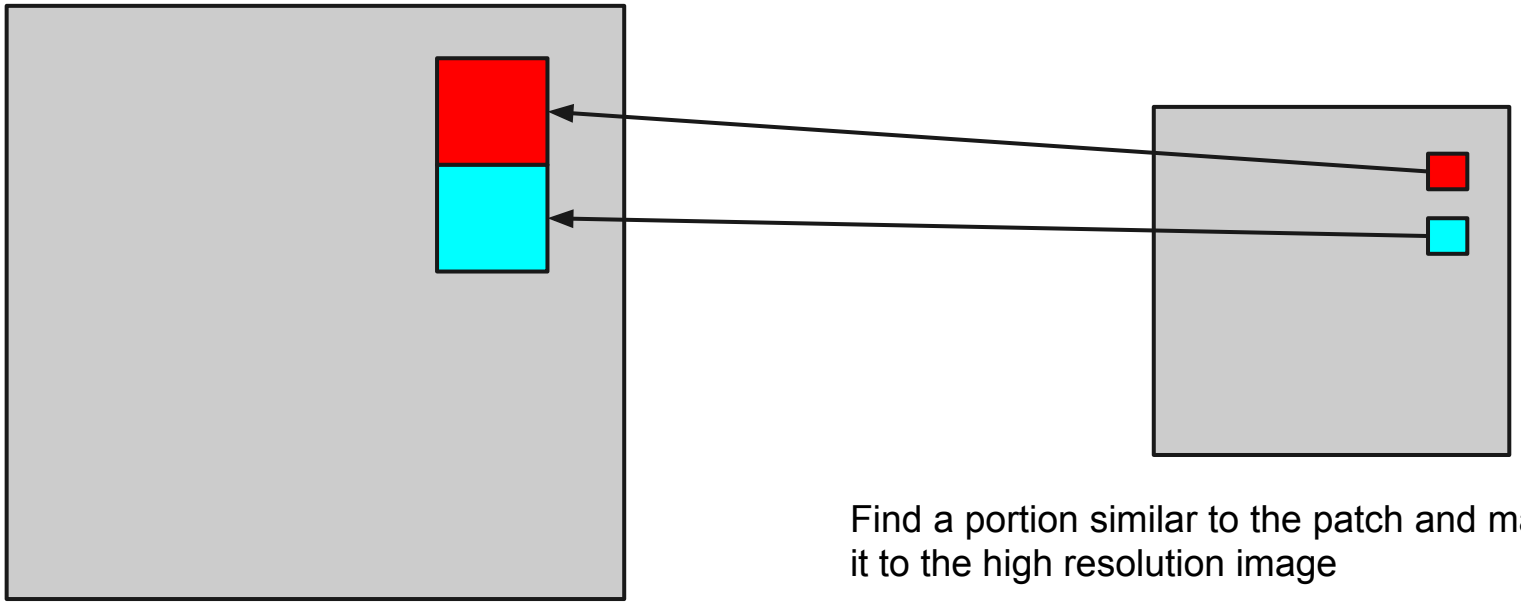


Single Image SR



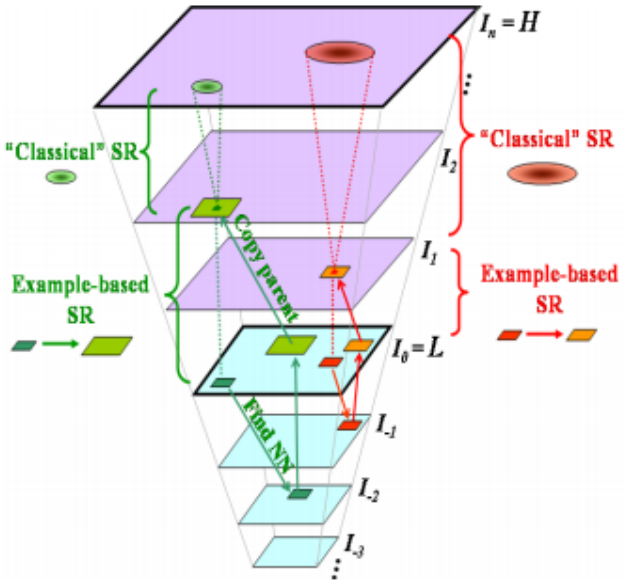
“Patch” - a portion on the image that does not blur or subsample when the image is scaled down

Single Image SR



Find a portion similar to the patch and map it to the high resolution image

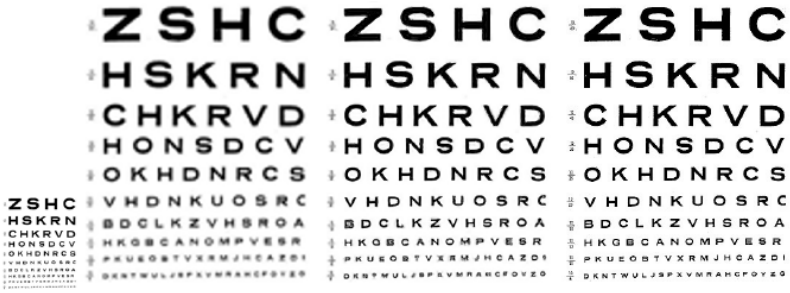
Single Image SR



The patches are discovered (4 green boxes) and is similar portions to the patches are identified producing the input low resolution image I_0.

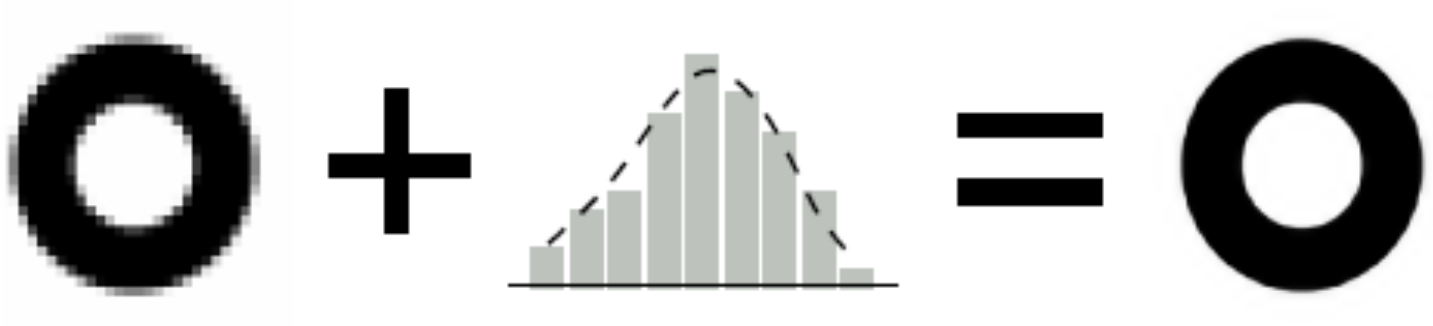
When the patches are mapped from the input low resolution (green) to the high resolution (violet). Then the high resolution image becomes an input to “learn” a new higher resolution image.

Single Image SR



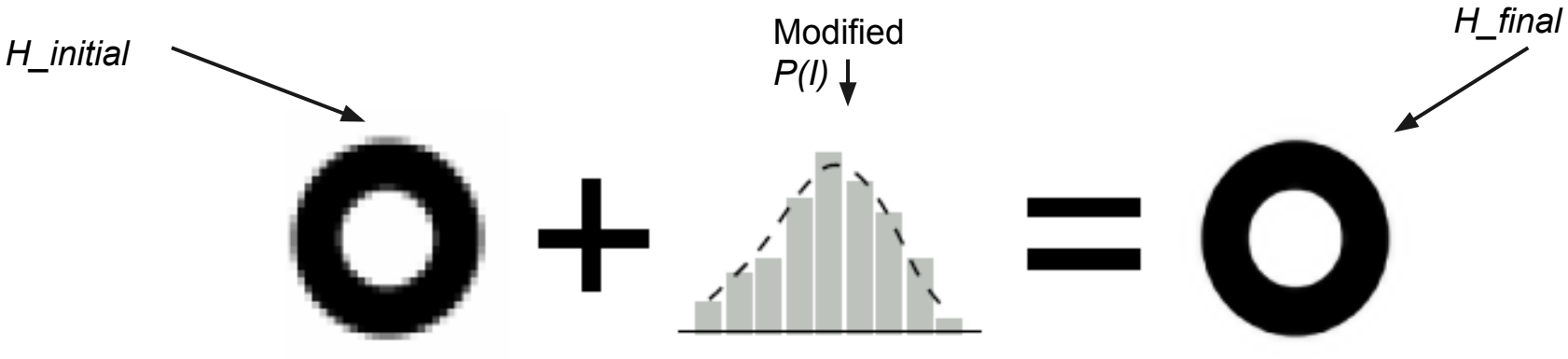
Single Image SR

- Highlighting edges



Single Image SR

1. Let H_initial be an upsampled input image L using bicubic interpolation
2. Extract edge features from H_initial.
3. Compute Gibbs distribution *P(I)*.
4. Extract image *L* intensity values and compare to *P(I)* to reduce false predictions of edges.
5. Use modified *P(I)* to highlight the edges (such as Laplacian sharpening) in *H_initial.*
6. Let *H_final* = *H_initial. H_final* be the output *H* image.



Single Image SR

Single Image SR

Limitations :

- The Feedback Control method is computationally expensive and takes a long time to produce the output. The repeated deconvolution and convolution also removes the details on the edges reducing the sharpness of the high resolution image.
- The Patch Redundancy method rely on the number of “patches” in the image. The number of patches determines how far could you get in improving the resolution of the image.
- In Edge Statistics method, while edges are preserved, we lose finer details inside the edges.

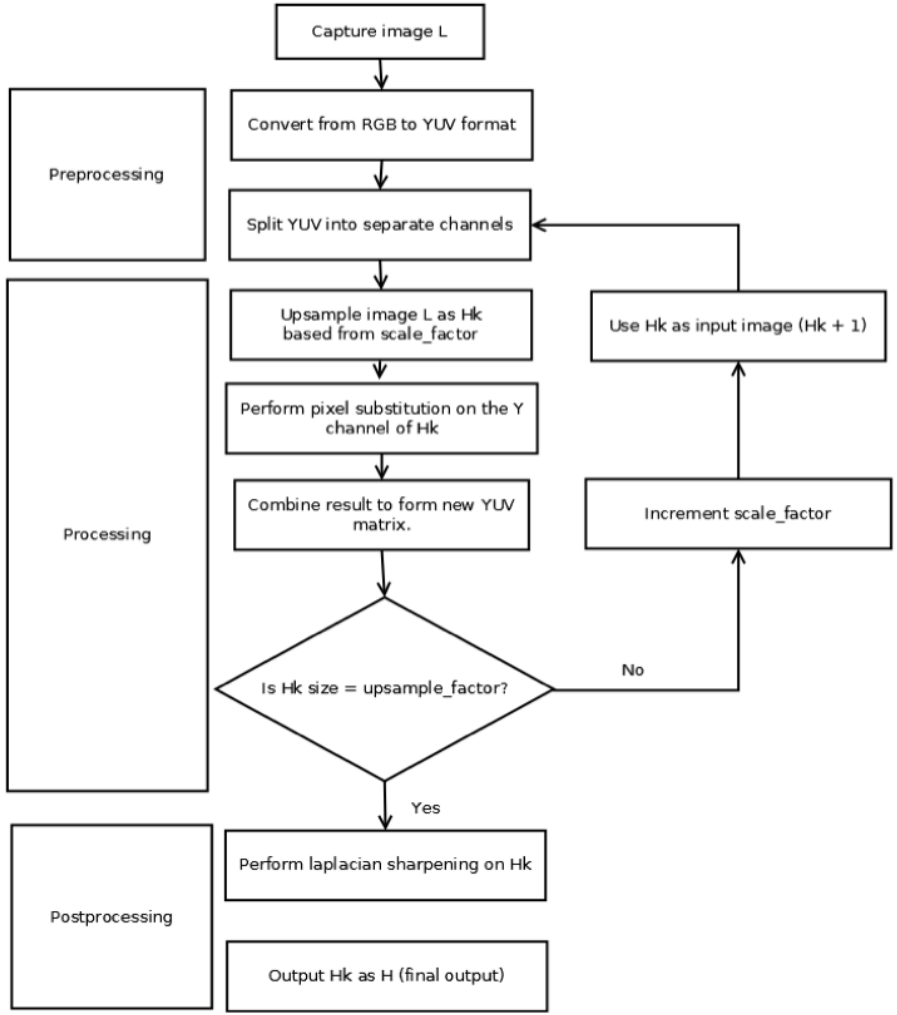
Applied SR in Android platform

- Using Android device camera
- Using openCV for image processing
- Input image - 1600x1200
- Output image - 6400 x 4800 (upsample of 4)
- Minimum OS: 4.0 (Ice Cream Sandwich)

The system framework

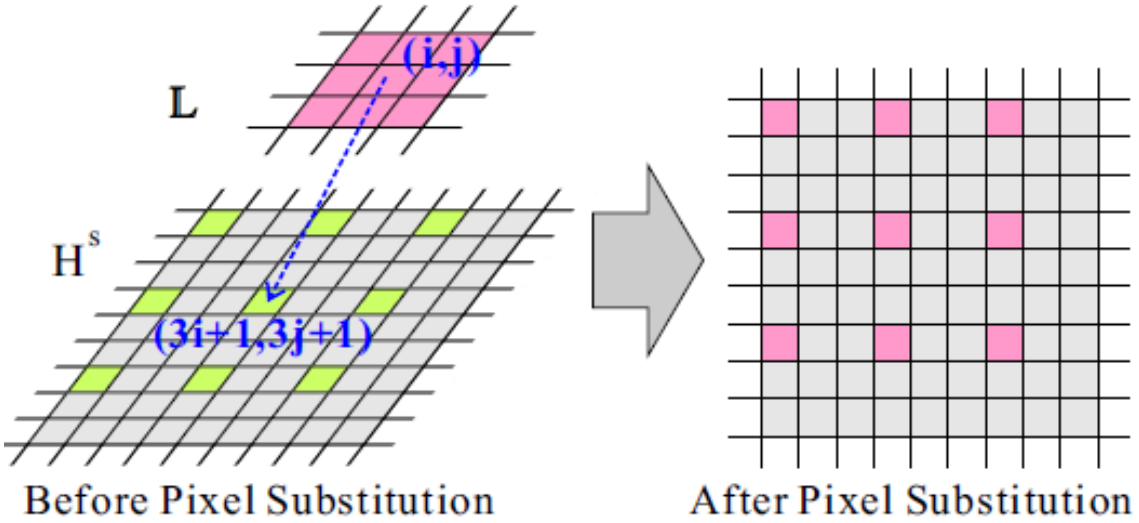
- Gradual scaling method from patch observation technique.
- Pixel substitution method to preserve as much information as we can from image *L.*
- Weighted edge sharpening using Laplacian operator.

- Divided into three stages.
- Pre-processing stage
  - Handles conversion of RGB to YUV
  - Splitting of YUV to separate channels
- Processing stage
  - Handles iterative upsampling and pixel substitution
  - Upsample using bicubic interpolation
  - Only process the Y channel for pixel substitution. Why? Performance consideration.
  - upsample_factor set to 4. scale_factor set to 0.25.
- Postprocessing stage
  - Laplacian sharpening
  - Weighted addition. RGB matrix bias of 1.0. Laplacian filter bias of -0.25. Why? Let's see the results.

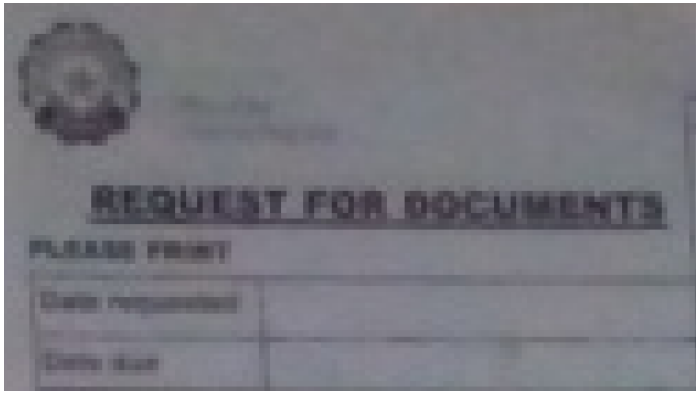


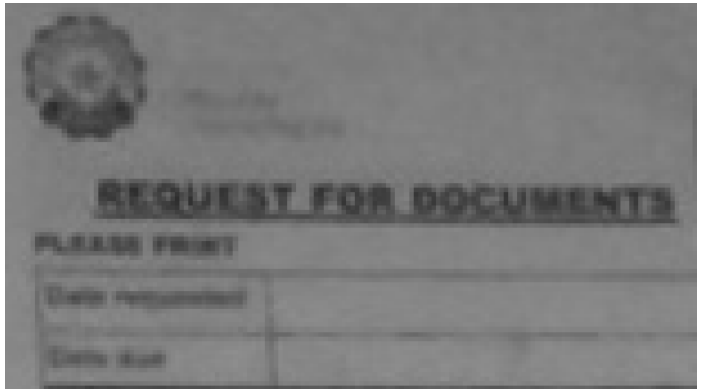
Pixel substitution example

Given upsample factor of 3. Replace corresponding pixels in image *H* from pixels in image *L*.

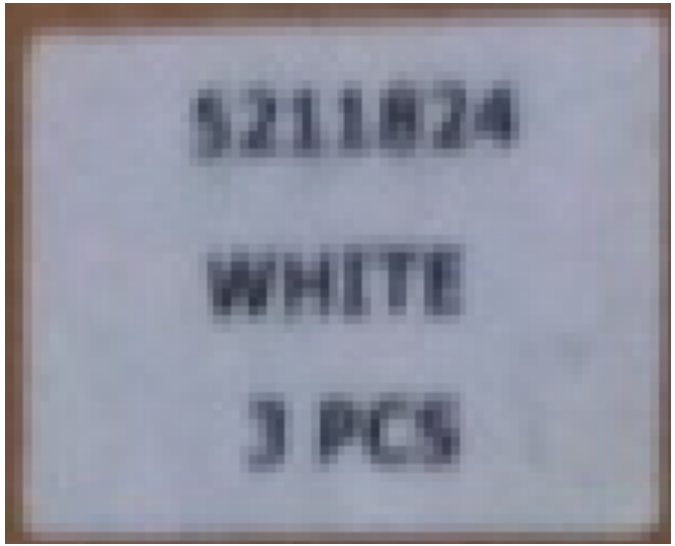


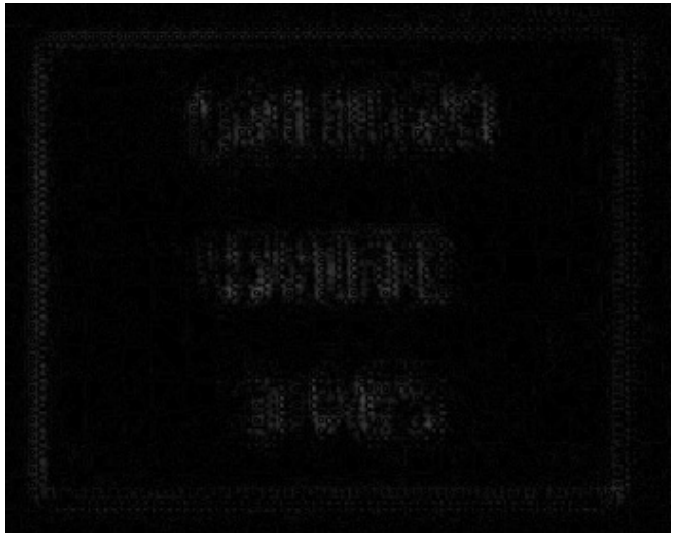
Pixel substitution example





Laplacian sharpening example





Results and Observations

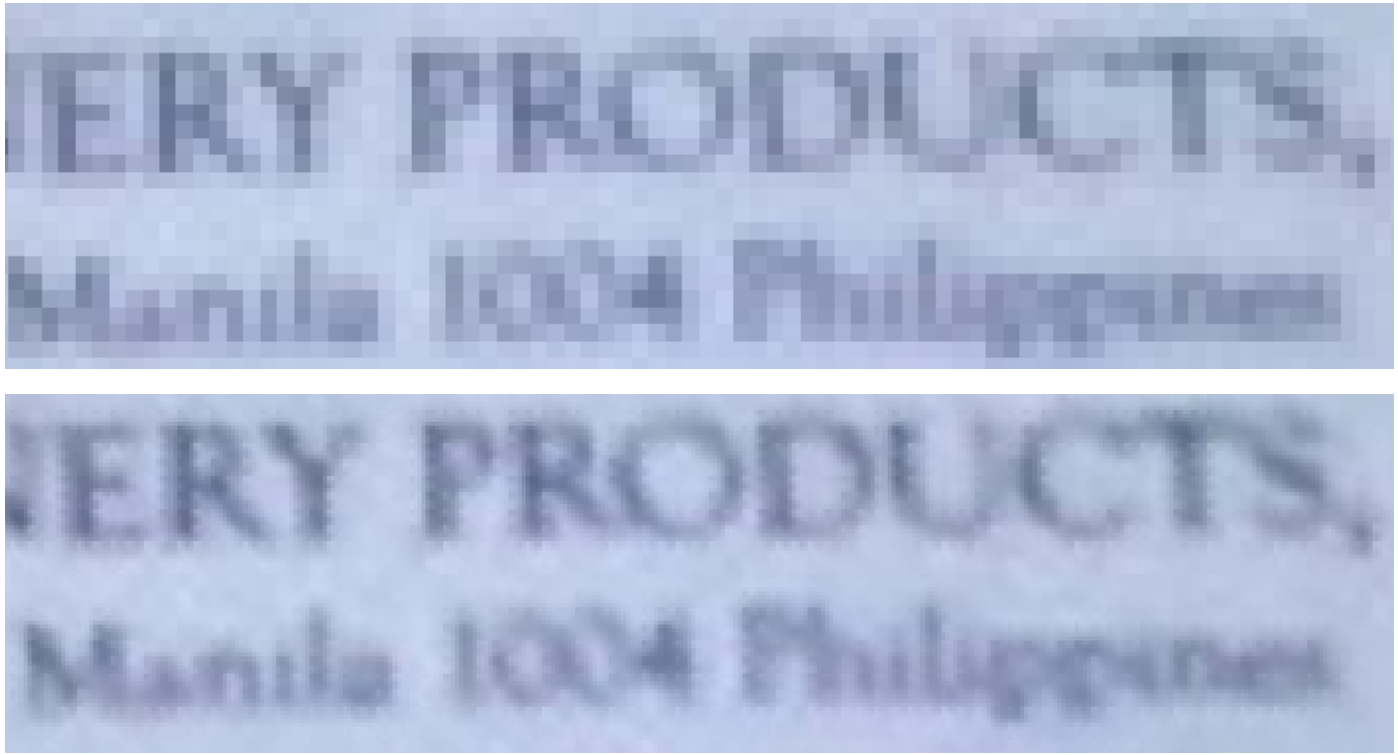
Results and Observations

- Unweighted addition.
- Very sharp edges resulted from laplacian filter.
- Solution? Reduce impact of the laplacian filtered image.

Results and Observations

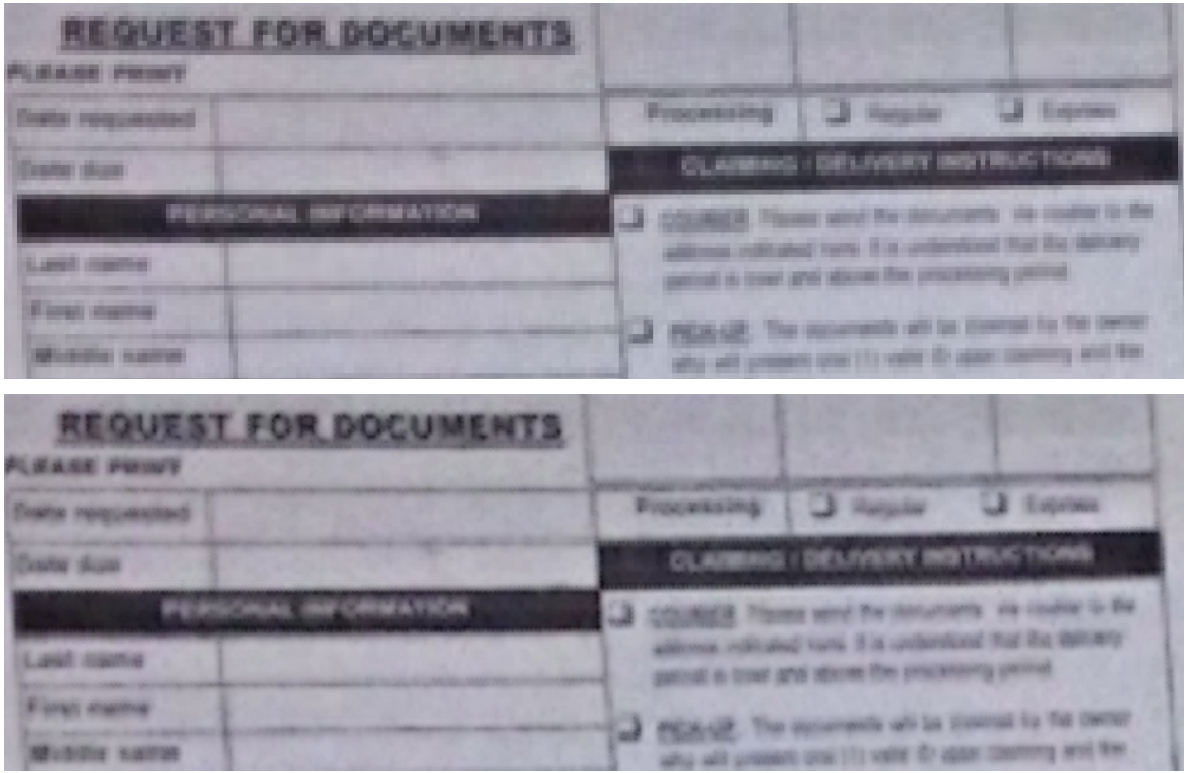
Using weighted addition.
RGB matrix = 1.0
Laplacian filtered matrix = -0.25

Results and Observations



Results and Observations

Results and Observations



Results and Observations

Results and Observations

Results and Observations

Other Details

- Given upsample_factor of 4 and scale_factor of 0.25, it takes 11 iterations to process the image.
- Running time: approximately 3-4 minutes. Oppo's product takes 10 seconds. :(
  - Possible reason? We process every pixel in the image. Oppo processes an average of 400 pixels only!
- Design issue. Cannot perform parallel execution. Result is always carried on the next iteration.
- Does not handle low lighting too well.

Other Details

- Why scale_factor is 0.25? Just purely observation.
  - 0.5 - we felt image was no different from a fast bicubic interpolation.
  - 0.15 - takes REALLY long to process!
  - 0.25 - the sweet spot.
- Why upsample_factor of 4?
  - Image input size is already huge. 1600x1200. More information was preserved.
  - Deterioration of images might not be seen and proven immediately (which is covered in our report).

Recommendations

- Superres in a nutshell.
  - Preserve as much information as you can! Processing becomes easier.
  - Compression and downsampling of image accounts for more than 60% loss of information. (i.e. compression of images for web use)
- Going beyond superresolution techniques here.
  - Recognition-based.
  - Suppose you have a blurry text image but you know the font style and you recognize the characters. Just attempt to redraw the image.
  - Hallucination algorithm is based from this principle.

Thank you! Q&A

- Thank you for listening!
- Q&A