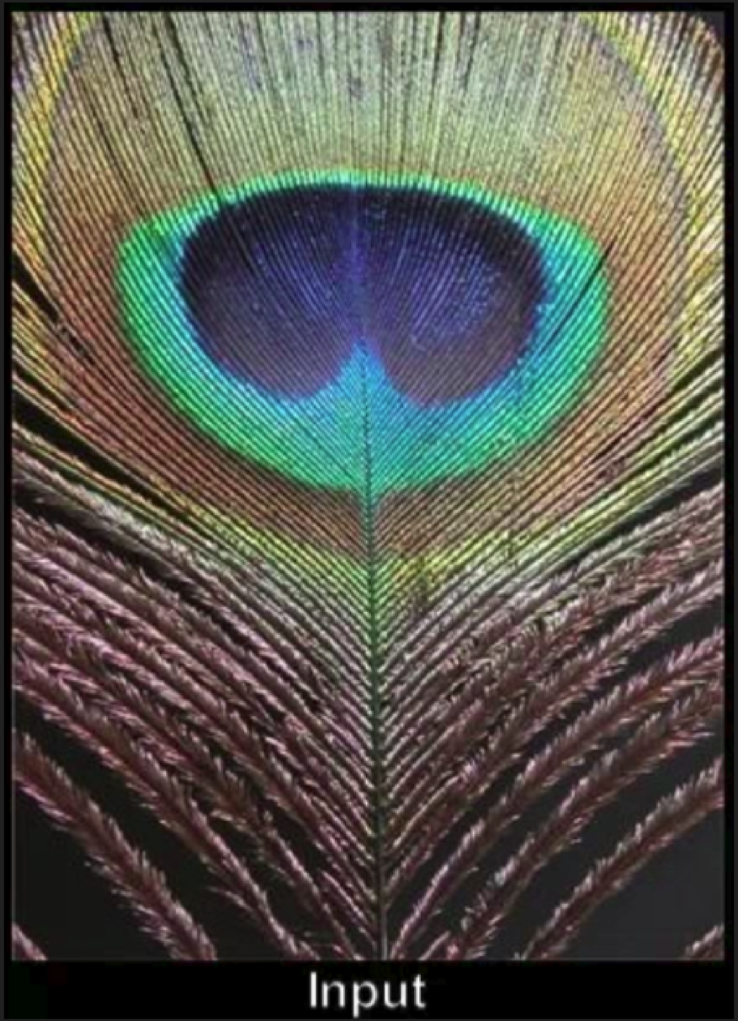
Content Adaptive Image Downscaling

CSE – 478 Digital Image Processing
Project 16
TA Mentor :- Joyneel Misra

1. Harshit Mahajan (201501231)
2. Akshay Vyas (201530032)
3. Ayush Rai (201501119)



Our Final Output

Downscaling Using Content Adaptive Kernel

Downscaled Image

Introduction to Downscaling

- Downscaling is one of the most commonly used image operation in image processing.
- In downscaling image, we aim to reduce the size of the image to required smaller size maintaining a proper trade off between sharpness (information content) and aliasing.

Previous Ideas For Downscaling

- Subsampling
- Various Windowing Filter (like Sinc)
- Bilinear
- Bicubic
- Lanczos

Problems with Existing Ideas

- Simply subsampling without prefiltering introduces strong aliasing artifacts.
- Methods like bicubic smoothens out fine details and edges.
- Lanczos method uses sinc filter but it produces ringing artifacts near strong image edges.
- Method developed by Nehab and Hoppe 2011 added a correction stage on discrete signal before reconstruction which resulted reduced ringing at a new computational cost.

Our Project Idea

- Key idea is to optimize the shape and locations of the downsampling kernels to better align with local image features.
- These content adaptive kernels are formed as a bilateral combination of two Gaussian kernels defined over space and colors.
- Technically we will be using maximum likelihood optimization using a constrained variation of Expectation-Maximization algorithm.

Major steps in Code

```
Function downscaling
        Initialization Step
        Loop
                Expectation Step        ( E-step )
                Maximization Step       ( M-step )
                Correction Step         ( C-Step )
                If no change in last M-step or C-step
                        return
                                End loop
                        End function
```

Notations

(w_i, h_i)	input (*high* resolution) image dimensions
(w_o, h_o)	output (*low* resolution) image dimensions
(x_k, y_k)	2D kernel indices; $(x_k, y_k) = (k \bmod w_o + \frac{1}{2}, \lfloor k/w_o \rfloor + \frac{1}{2})$
(r_x, r_y)	input/output dimension ratios; $(r_x, r_y) = (w_i/w_o, h_i/h_o)$
k, n	kernel indices; $k, n \in [0, w_o \cdot h_o - 1]$
$\mathbf{\mu}_k$	spatial mean
Σ_k	spatial covariance
$\mathbf{\nu}_k$	color mean
σ_k	color variance
R_k	set of pixel indices where kernel k can become non-zero; $R_k = \{x + yw_i \mid 0 \leq x < w_i, 0 \leq y < h_i, \lvert x - x_k \rvert < 2r_x, \lvert y - y_k \rvert < 2r_y\}$
N_k^4	Set of indices of the 4-neighbors of kernel k
N_k^8	Set of indices of the 8-neighbors of kernel k
i	pixel index; $i \in [0, w_i \cdot h_i - 1]$
$\mathbf{p}_i$	pixel location; $p_i = (i \bmod w_i, \lfloor i/w_i \rfloor)$
$\mathbf{c}_i$	CIELAB color of pixel i
$w_k(i)$	Value of kernel k at pixel i
$\gamma_k(i)$	Value of *normalized* kernel k at pixel i

Major equations in Code

- E-Step :- we compute assignment probabilities of each pixel to each kernel, assuming the current estimate of the parameters is correct.

  $\gamma_k(i) \leftarrow Pr(k \mid x_i ; \theta) = w_k(i) / \sum_n w_n(i)$

  -> $\gamma_k(i)$ quantifies contribution of input pixel i relatively to the final color of the output pixel k.

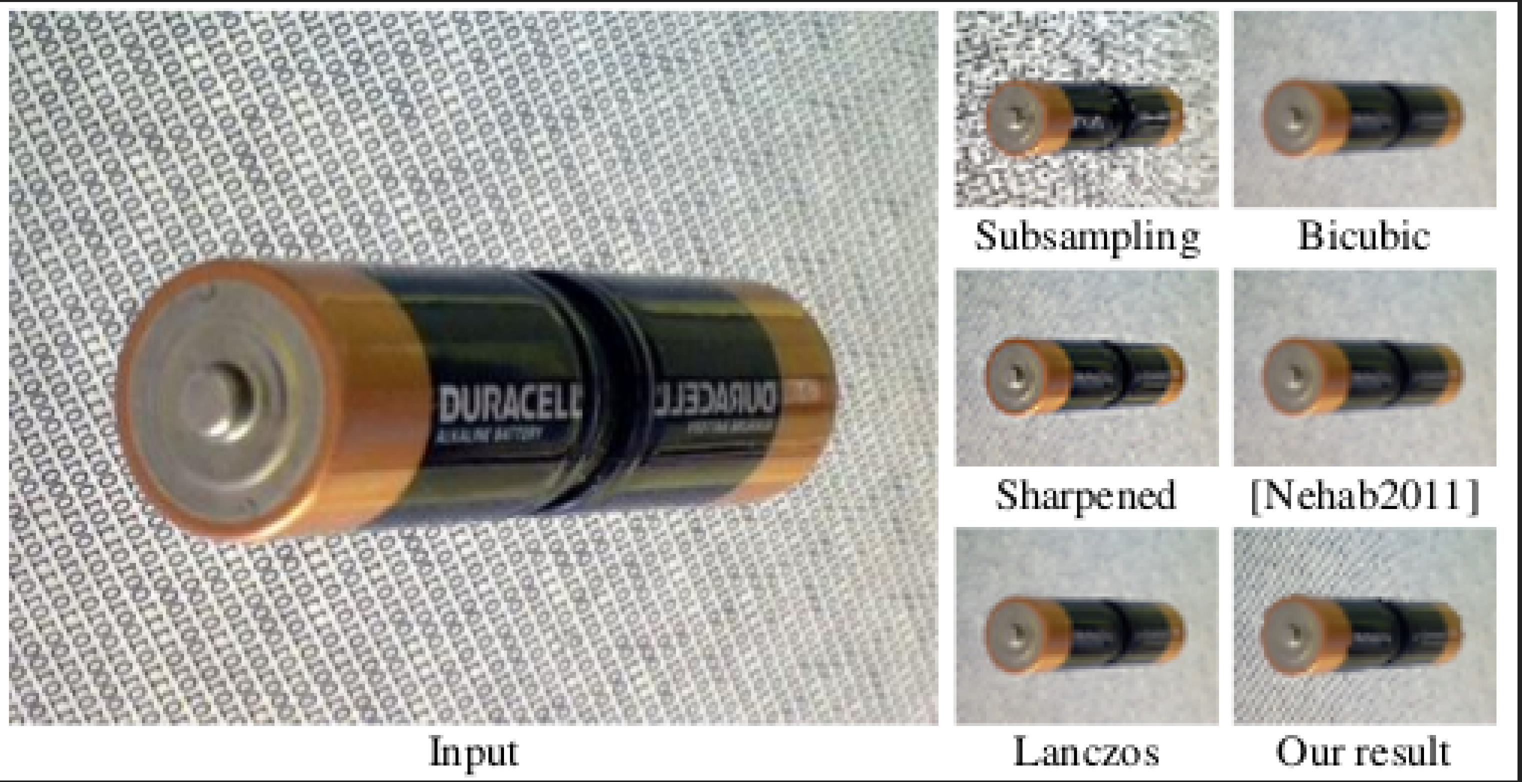
- M-Step :- we use these assignments in a weighted maximum-likelihood fit to update the estimate of the parameters θ :

  $\mu_k \leftarrow (\sum_i \gamma_k(i)\, p_i) / \sum_i \gamma_k(i);$

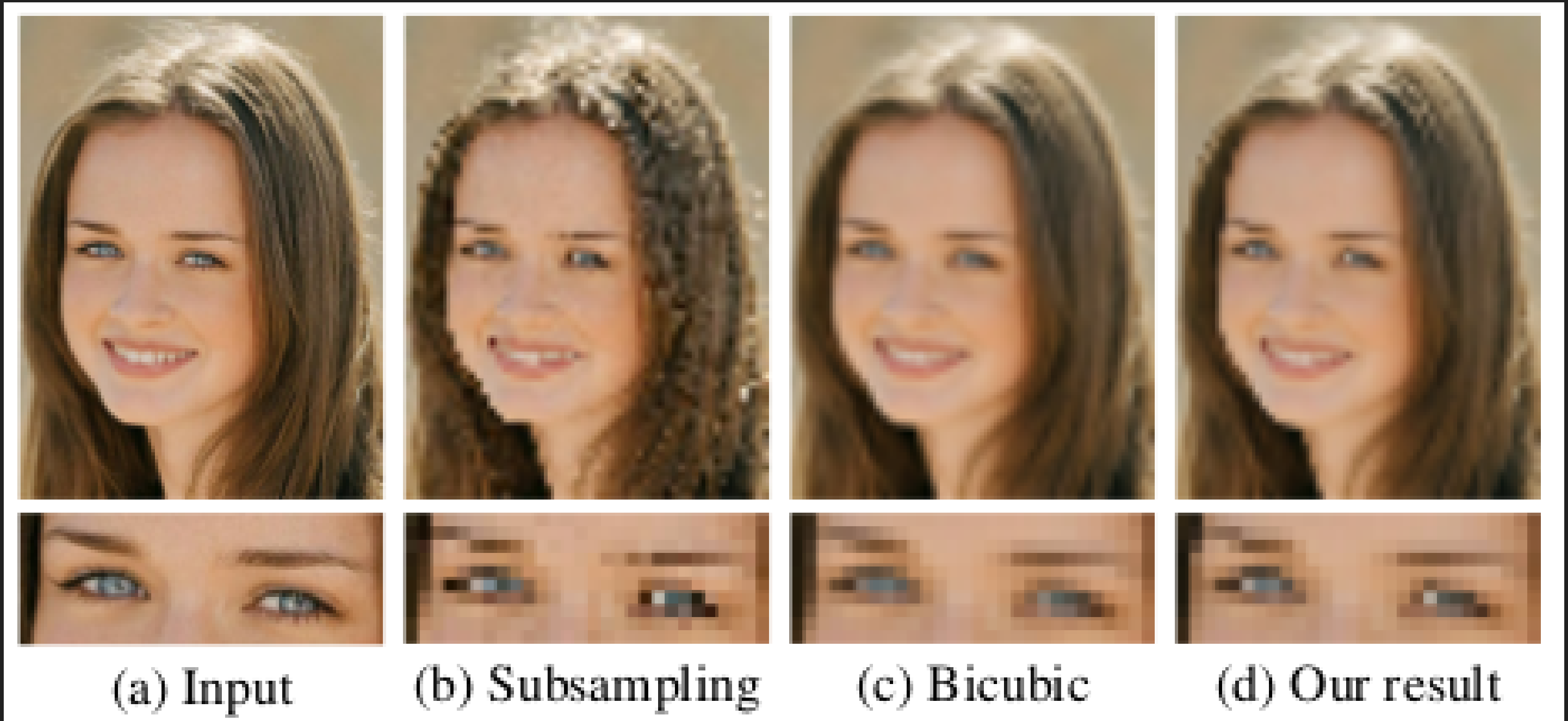
  $\Sigma_k \leftarrow \sum_i \gamma_k(i)(p_i - \mu_k)(p_i - \mu_k)^T / \sum_i \gamma_k(i);$

  $\nu_k \leftarrow \sum_i \gamma(i) c_i / \sum_i \gamma(i)$

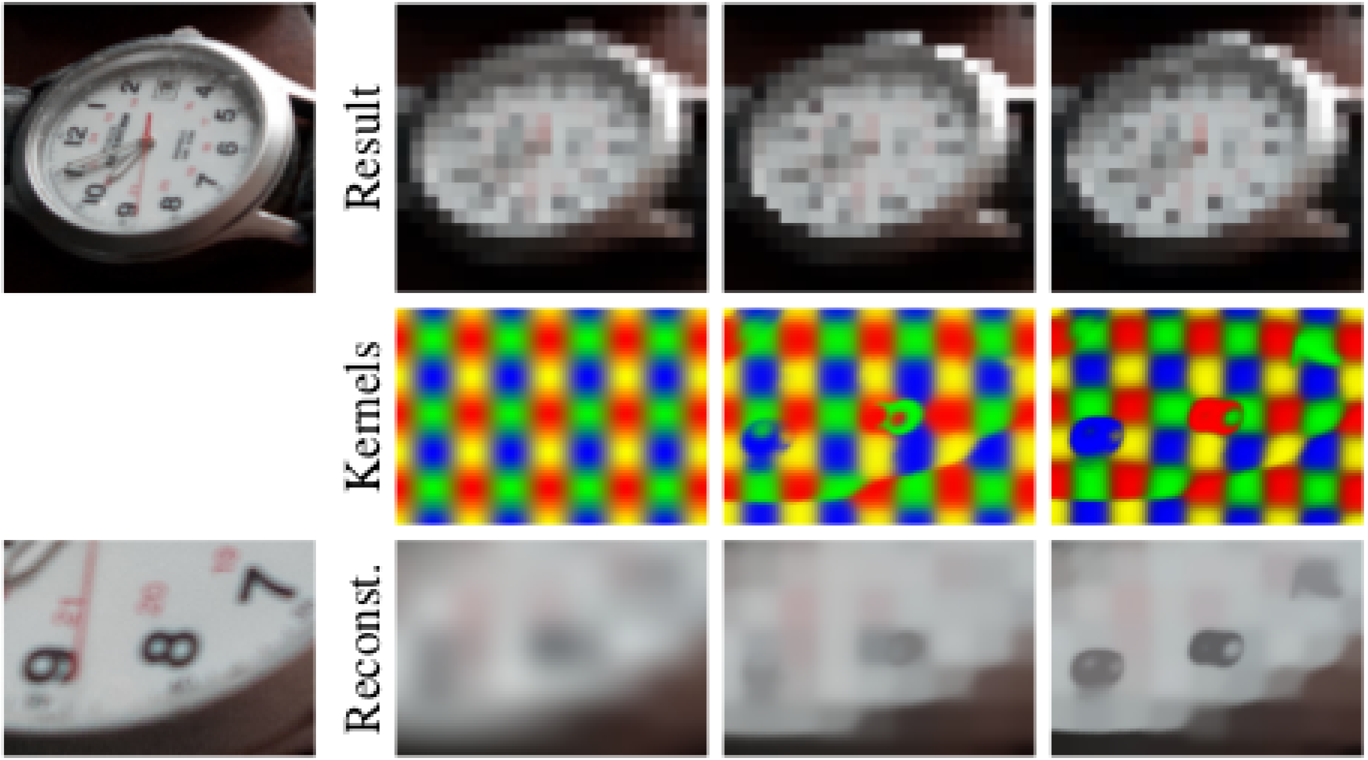
Comparison With Other Downscaling Method



Comparison With Other Downscaling Method



Comparison With Other Downscaling Method



(a) Input (b) Bicubic (c) Bilateral (d) Our result

Advantage Of using The Idea

- The new algorithm is designed in such a way that it provides a more balanced result between preserving sharpness of image and prevention of aliasing artifact.
- It outputs more sharp downscaled image and neither contains noise nor ringing, and mostly avoids aliasing artifacts.
- It is also useful for downscaling cartoon and vector art, and can also be combined with palette reduction to create pixel art imagery.

Limitations Of the Idea

- Running time of this algorithm is around 100 times that of other naive algorithms for pixels in the range of ~10^6 as it uses EM algorithm which may take time to converge.
- The algorithm does not always produce better results on images with blurred features, or images that contain structured textures.
- It uses EM-algorithm in which we may have multiple local maxima and we may reach slightly different solutions depending on our initialization.

Our Work Flow

- Our aim was to implement the content adaptive downscaling algorithm and test the algorithm on various images.
- We also compared output of this algorithm with some other algorithms
- We also tried to improve the performance and quality of algorithm mentioned in Koph paper.
- We compared it with other advanced algorithms like Perceptually Based Downscaling Algorithm.

Continued……

- We divided the project into smaller milestones and handed each individual separate chunk of work in order to achieve those milestones.

  The milestones we decided were:-

1. Implementing the expectation-maximization algorithm for single iteration.
2. Applying correction step on the output of the image obtained from expectation-maximization step.
3. Now, implementing the iterative version of the above two steps.
4. Testing and comparing with other algorithms.

Challenges

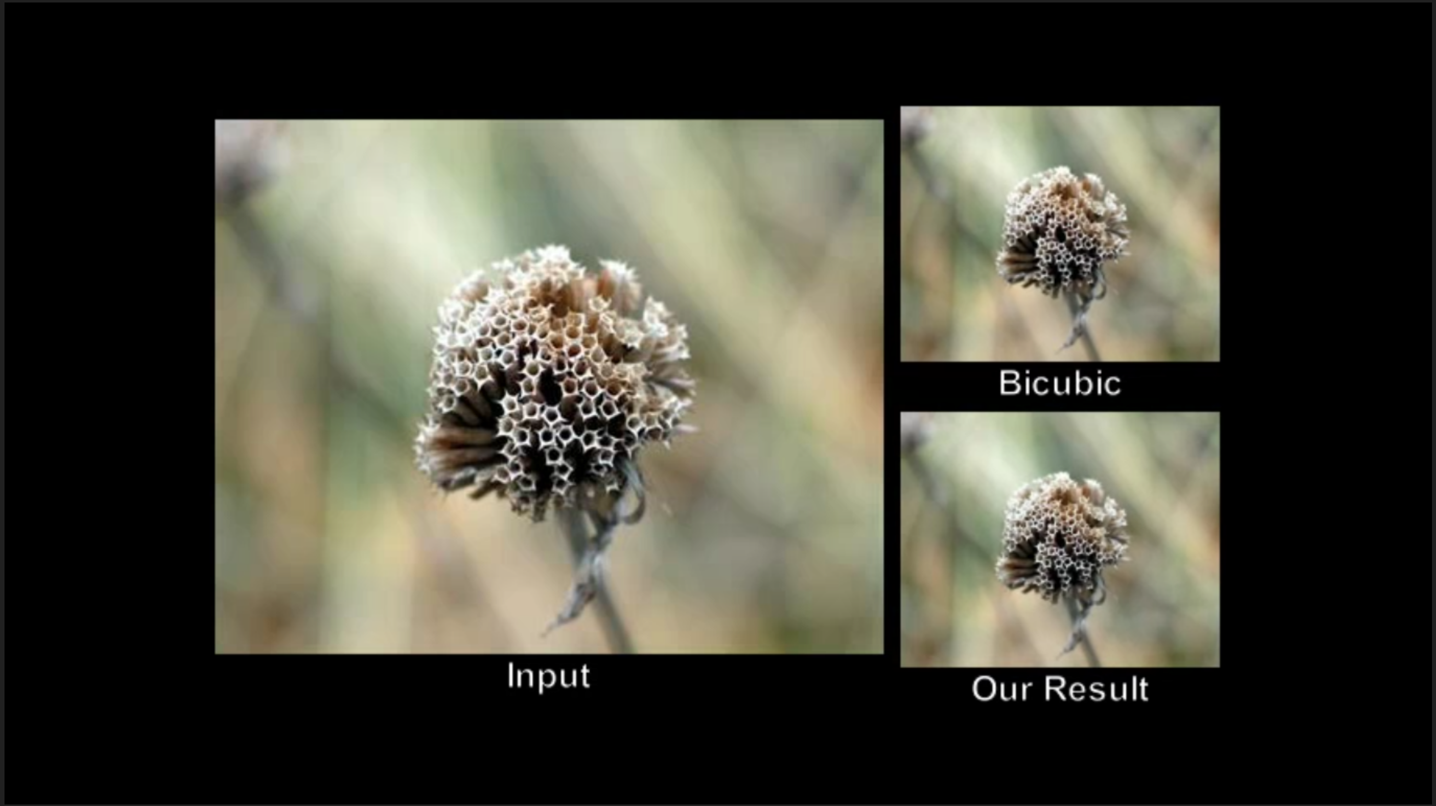
1. Required lots of time to test and run the code even on small images.

2. For larger images, we were to test the code as the memory limit got exceeded.

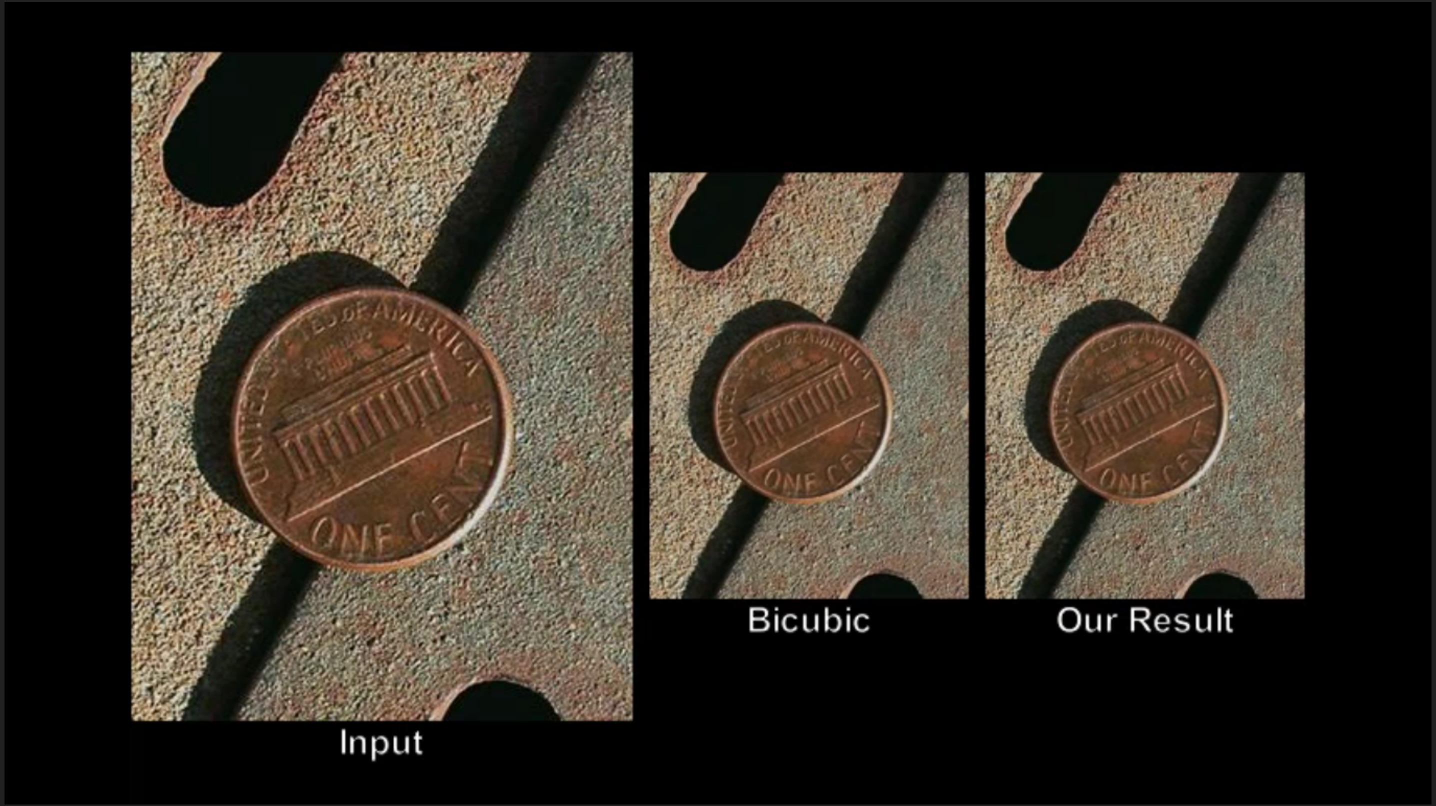
3. We were unable to find the angle between distance vector and derivative of gamma vector(probability density function) which was used in constraint step.

4. Scaling down CIELAB color space values between 0 and 1.

Sample Outputs



Sample Outputs



Resources And Tools Required

- Resources : “Content Adaptive Image Downscaling” by Johannes Kopf, Ariel Shamir and Pieter Peers.
- Tools: Matlab, Python

Thank You!