INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

Digital Image Processing - CS 663

Toonification of Image

Avyakta Wrat - 180070010 Nayan Barhate - 180070037

Meeta Malviya - 180070034 Varad Mane - 18D070034

December 6, 2020

Algorithm

1. The image is smoothen out and then passed through a median filter to remove any salt and pepper noise in the image.
2. Then edges are identified using Canny Edge Detection.
3. Then the original image is passed through a bilateral filter.
4. The detected edges are added back to the original image. This ensures that the edges in the picture are black.
5. The image's pixel intensities are quantized to bring out the solid colour texture in the cartoonified image.

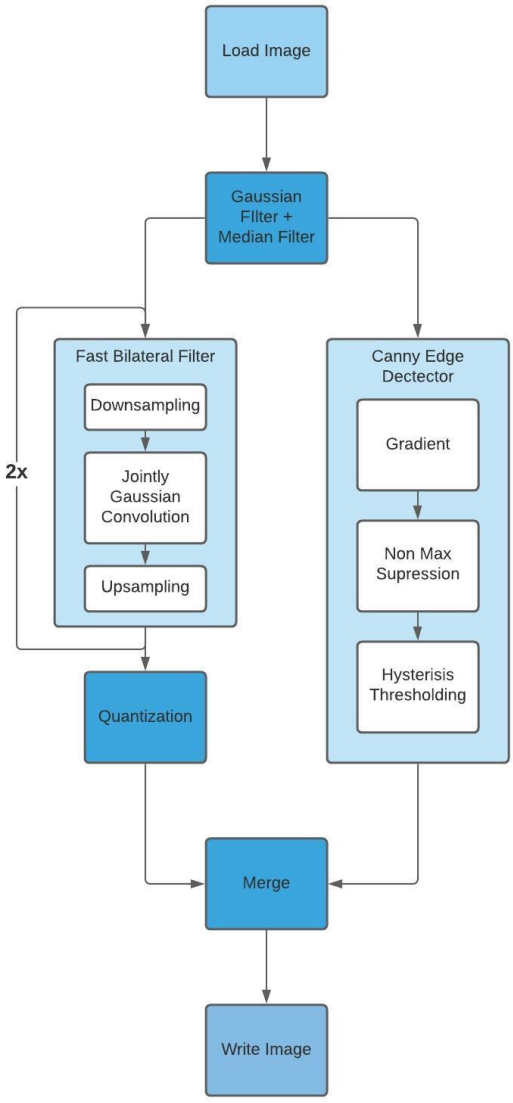


Figure 1: Block Diagram

Canny Edge Detection

We implemented Canny Edge Detection by first reducing noise using Gaussian smoothing. Next up, gradients at each point are computed, and non max suppression is performed by comparing the gradient at points unit vectors ahead and behind the current point, unit vectors having a direction along the gradient, which does require interpolation as these points may not have integral values. This is followed by hysteresis thresholding to trace the edges. The parameters for this threshold were kept at 0.075 $\times$ max(gradient) as the T_{low} and 0.1 $\times$ max(gradient) as T_{high}

Bilateral

It is a filter that homogenizes color regions while preserving edges. Is applied to the images. This filter gives the cartoonish texture to the image. The basic algorithm involves adjusting pixel value for a particular pixel by weighing in all the surrounding pixels based on the distance from this particular pixel in space as well as in terms of intensity. A gaussian kernel is used to define the weights, hence we have 2 tunable parameters, one for distance and one for intensity

$$\begin{pmatrix} W_{\mathrm{p}}^{\mathrm{b}} I_{\mathrm{p}}^{\mathrm{b}} \\ W_{\mathrm{p}}^{\mathrm{b}} \end{pmatrix} = \sum_{\mathbf{q} \in S} G_{\sigma_i}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}\left(|I_{\mathbf{p}} - I_{\mathbf{q}}|\right) \begin{pmatrix} W_{\mathbf{q}} I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix}$$

Fast Bilateral

The above is a non linear approach, a faster computation can be achieved by convoluting the image.

$\mathcal{S}$: spatial domain, $\mathcal{R}$: Intensity domain $\mathcal{G}$: Gaussian kernal $\otimes$: convolution

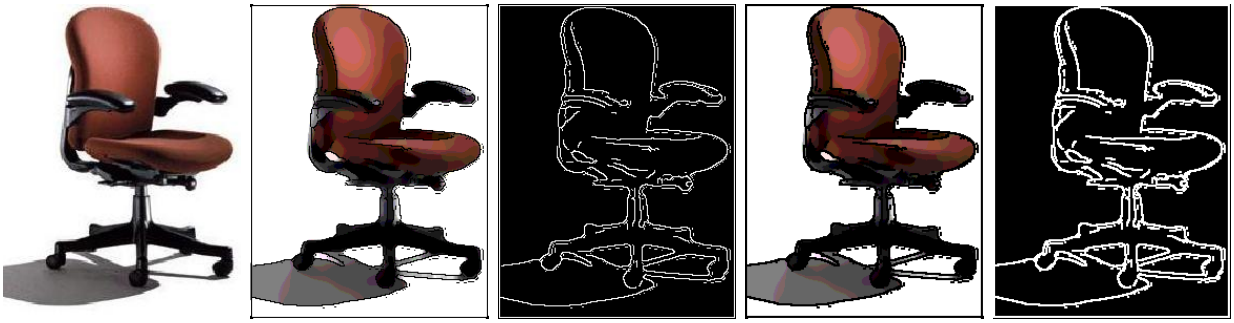
$\sigma_{\mathrm{s}}, \sigma_{\mathrm{r}}$:Gaussian parameters(space and range) $I_{\mathbf{p}}^{\mathrm{b}}$: Output intensity of bilateral

$$g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}}: \quad (\mathrm{x} \in \mathcal{S}, \zeta \in \mathcal{R}) \quad \mapsto \quad G_{\sigma_{\mathrm{s}}}(\|\mathrm{x}\|) G_{\sigma_{\mathrm{r}}}(|\zeta|)$$

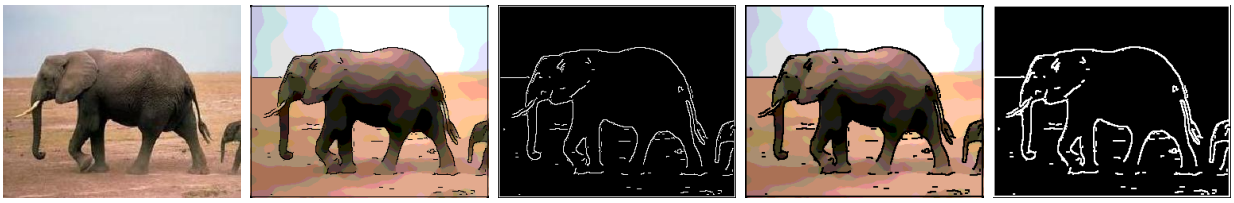
$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{b}} I_{\mathbf{p}}^{\mathrm{b}} \\ W_{\mathbf{p}}^{\mathrm{b}} \end{pmatrix} = \left[g_{\sigma_0,\sigma_r} \otimes \begin{pmatrix} w_i \\ w \end{pmatrix} \right] (\mathbf{p}, I_{\mathbf{p}})$$

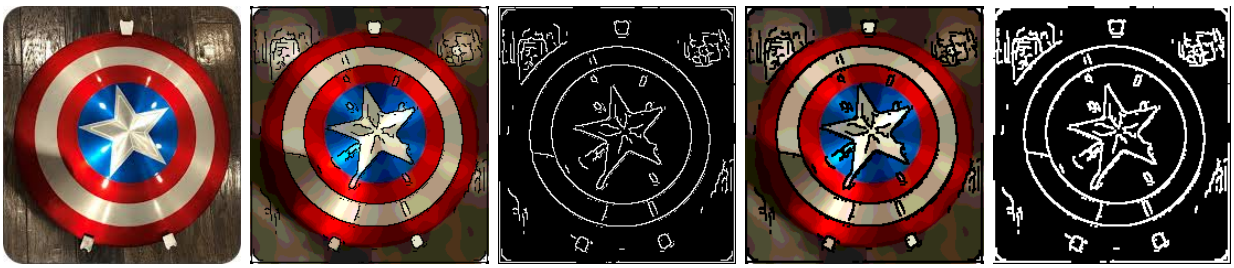
$$I_{\mathbf{p}}^{\mathrm{b}} = \frac{w^{\mathrm{b}}(\mathbf{p}, I_{\mathbf{p}})\, i^{\mathrm{b}}(\mathbf{p}, I_{\mathbf{p}})}{w^{\mathrm{b}}(\mathbf{p}, I_{\mathbf{p}})}$$

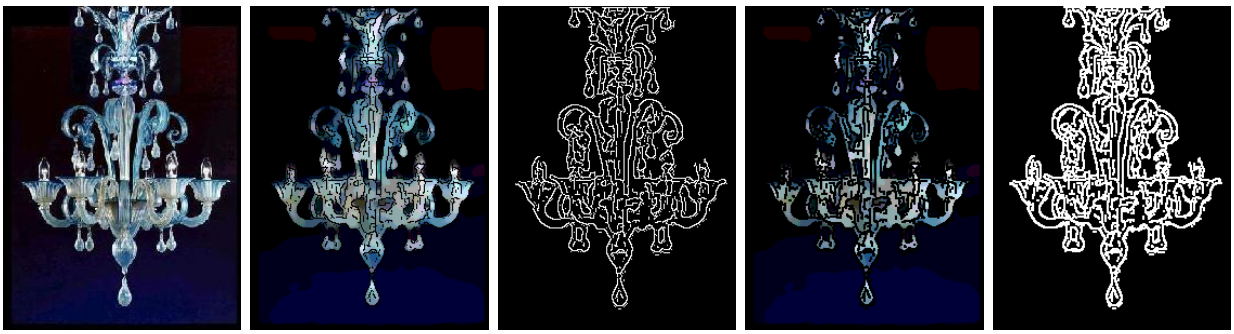
Results

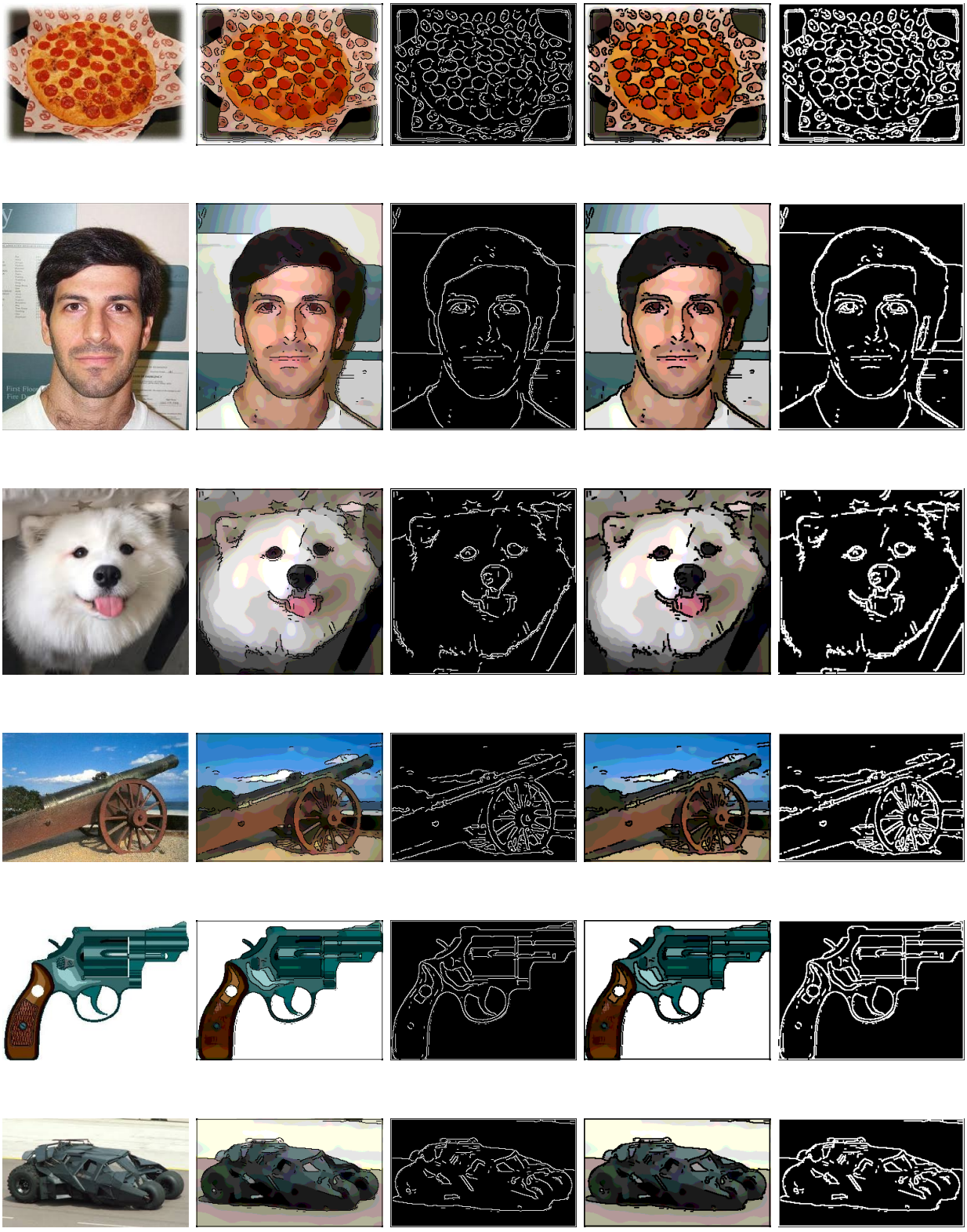


(a) Original image to be toonified (b) Toonified image (c) Edges in the image (d) Toonified image with dilation (e) Dilated Edges in the image









(a) Original image to be toonified (b) Toonified image (c) Edges in the image (d) Toonified image with dilation (e) Dilated Edges in the image

All these images above have been obtained by setting T_{low} and T_{high} constant and equal to 0.075 and 0.1 times the maximum gradient value. The images from left to right are the Original Image, Toonified Image without performing edge dilation, edges in the image, Toonified Image with edge dilation, dilated edges of the image.
We tuned the parameters for few images which didn't give good results using the values given above.

We've implemented the fast bilateral filter, and have compared execution time with the normal implementation, here are the results for $\sigma_s = 4 \& \sigma_r = 8$ and down-sampled by the factor of 2.

Fast Bilateral Filter Time(s)	Normal Implementation Time(s)
0.54	1.35
0.51	1.19
0.55	1.19
0.66	1.19
0.58	1.23

Remarks

1. We found that the dilated edges gave more aesthetically pleasing and more cartoonish images
2. Sometimes if the input image had a lot of edges which weren't actually edges but appeared so due to light variations, non dilated edges seemed better(eg. pizza image up above)
3. Fast bilateral gave much better performance with approximately same results
4. The algorithm seems best suited for images that have a few large regions, each with relatively low color diversity
5. For some images with more egdes we had to tune parameters to output a better image

References

1. Kevin Dade(Stanford University) *Toonify: Cartoon Photo Effect Application* `https://stacks.stanford.edu/file/druid:yt916dh6570/Dade_Toonify.pdf`
2. Sylvain Paris, Fredo Durand(Massachusetts Institute of Technology) *A Fast Approximation of the Bilateral Filter using a Signal Processing Approach* `http://people.csail.mit.edu/sparis/publi/2006/tr/Paris_06_Fast_Bilateral_Filter_MIT_TR.pdf`

Figure 16: **Aesthetically pleasing Toonified Panda for good luck :P**