Compressing Stereo Images Using a Reference Image and the Exhaustive Block Matching Algorithm to Estimate Disparity between the Two Images

Ashish Agarwal
Student, School of Engineering Science
Simon Fraser University, Canada
ashish-agarwal@hotmail.com

Abstract

One method to compress a pair of stereo images is to compress the first image independently and then compress the second image by estimating the disparity between the two images. The first image was transformed using 2-D Discrete Cosine Transform, quantized using the JPEG quantization matrix and finally encoded into a bitstream using arithmetic encoding. The disparity between the two images was estimated by the Exhaustive Block Matching Algorithm (EBMA) and the resulting disparity vector was then encoded into a bitstream. The images were then decoded and were compared with the original images.

Keywords: *stereo imaging, stereoscopy, stereo image compression*

1. Introduction

A pair of stereo images is very similar to each other as they are the images of a stationary object taken from two different angles. This is why compressing both images independently is an in-efficient way of compressing stereo images. An efficient way to compress a pair of stereoscopic images is to calculate the difference between the two images; also known as disparity estimation and then compress one image independently. This image is known as the reference image and can be either the right image or the left image. The reference image and the disparity vectors are then used to reconstruct the second image.

A stereo image is produced by taking two cameras, separated by a distance of 6.5 centimeters (which is approximately the distance between the human eyes) and recording the perspectives of the right eye and the left eye using different lenses [2]. The left image is seen through the left lens and the right image is seen through the right lens. The brain then merges the two images into one and also perceives the depth of the object.

Stereo images can be produced cheaply using inexpensive digital cameras and are used widely in clinical applications, mining, metallurgy, environmental science and entertainment [1].

In this paper, the left image is taken as the reference image and the disparityvectors between the two imagesare estimated using the Exhaustive Block Matching Algorithm (EBMA). The reference image is transformed using two-dimensional discrete cosine transform (DCT-II) and quantized using the JPEG quantization matrix. The resulting matrix is compressed into a bitstream using arithmetic coding.

2. Stereoscopic Image Encoder

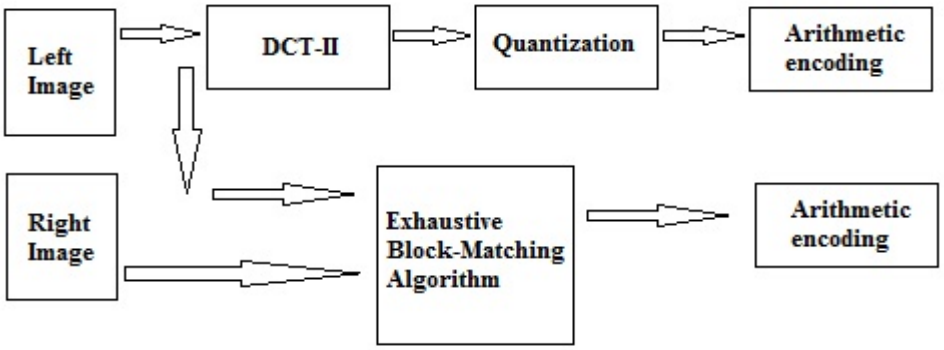


Figure 1: Compressing a Pair of Stereo Images

The left image is taken as the reference image and is transformed using two dimensional forward discrete cosine transform (DCT-II). The resulting image matrix is quantized using the JPEG quantization matrix in (3) and then compressed using Arithmetic coding.

The second part of the encoder involves compressing the right image. Since the two images are very similar to each other, disparity vectors between the two images are estimated. The resulting disparity vectors are compressed into a bitstream using arithmetic encoding as described in 2.3.

2.1. Transformation

Transform coding is a component of image processing and is very effective in the compression of still images [5]. Transformation is mainly used to remove the duplicate information between the neighboring pixels in the image. It depends on the assumption that a pixel in an image is correlated to its neighbor, and this correlation is used to predict the value of the neighboring pixel.The 2-D Discrete Cosine Transform (DCT) was used to perform the transformation because of its strong "energy compaction" [6] property.

The steps to perform transformation are –

Step 1 – To perform DCT-II, the first step is to ensure that the pixels are centered around zero. Since the value of a pixel is in the range [0, 255], it can be centered around zero by extracting the value of each pixel and subtracting from it 128.

Step 2 – The image is divided into NxN blocks, and the 2-D DCT for each block is calculated as –

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \quad (1)$$

for u,v = 0,1,2…..,N-1 and alpha(u), alpha(v) are the horizontal and vertical spatial frequencies and are defined as –

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad u = 0 \\ \sqrt{\frac{2}{N}} & for \quad u \neq 0. \end{cases} \quad (2)$$

2.2. Quantization

The human eye is insensitive to variations in brightness of high-frequency components over a large area. Therefore, the high frequency values in the image matrix can be rounded off to zero without the user noticing any difference in the quality of the image [7].Quantization achieves this compression by dividing the DCT output for each block by a quantization co-efficient, and then rounding the result into the closest integer [7]. The 8x8 quantization matrix used in this paper is [8] –

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad (3)$$

Once DCT has been performed, the image is divided into 8x8 blocks, and each pixel is quantized as –

$$C_{i,j}{}' = round(\frac{C_{i,j}}{Q_{i,j}}) \quad (4)$$

2.3. Arithmetic Encoding

Arithmetic coding achieves lossless compression by assigning short codewords to the frequencies that have a higher probability of occurrence and longer codewords to frequencies whose probability of occurrence is lower [11].

The pseudo code for Arithmetic coding is given below –

```
low <-0;
high <-1;
range <- high-low;
for (each input symbol) do
      Sn<- read the next input symbol
      range <- high-low;
      high <- low + range * CDF(Sn);
      low <-  low + range * CDF(Sn-1);
end
return(low, high);
```

2.4. Disparity Estimation Using the Exhaustive Block Matching Algorithm

The Exhaustive Block Matching Algorithm is the most accurate block matching algorithm and is also the easiest to implement [10]. In Figure 3, if the left image is assumed as the anchor frame B_m, the problem is to determine a block in the target frame i.e. the left frame where the difference between the two blocks is minimum.

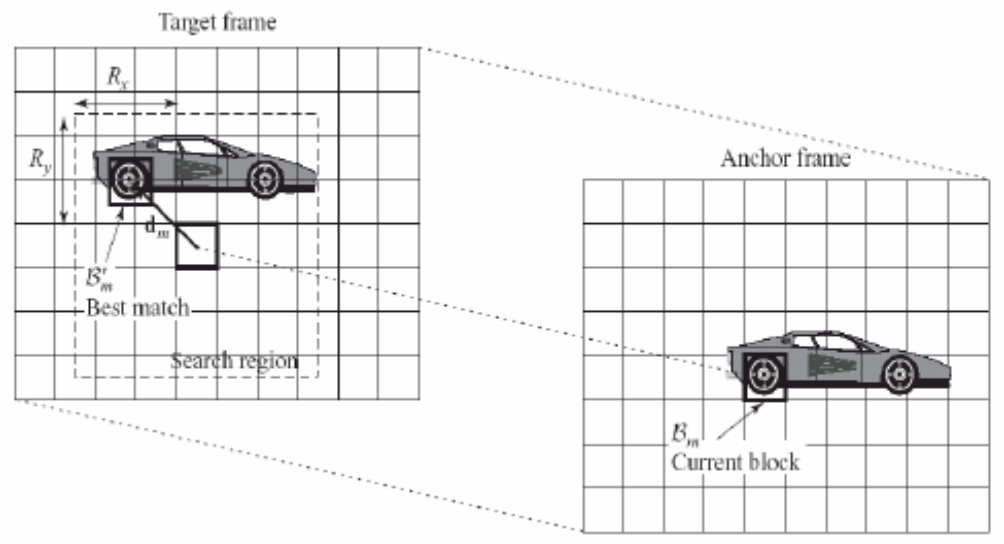


Figure 2: Disparity Estimation in Exhaustive Block Matching Algorithm

In this paper the size of a macro block was taken as 16x16 pixels and each block in the anchor frame was compared with each block in the target frame. The block with the minimum error was calculated by the equation [10] –

$$E_{\mathrm{DFD}}(\mathbf{d}_{\mathrm{m}}) = \sum_{\mathbf{x}\in B_m} \left|\psi_2(\mathbf{x}+\mathbf{d}_m) - \psi_1(\mathbf{x})\right|^p \tag{5}$$

The displacement between the current block and the best matching block is the displacement vector d_m and is known as the disparity.

2.4.1. Complexity of EBMA: In this paper an operation is considered as one subtraction, one addition or calculating one absolute difference, the total number of operations for –

Calculating the Minimum Absolute Difference for each block is N^2

Calculating the disparity for one block is (2R+1)^2N^2

where, the block size is assumed to be NxN pixels and the search size is ± R pixels.

2.4.2 Algorithm: The blocks will have M rows and N columns and the search size is assumed to be ± R pixels. The parameter search_size is the range over which the reference image will search for the best matching block in the second image. This means that if the search size is 8 pixels, the best matching block will be searched over ±8 pixels relative to the position of the current block. The x-component and the y-component of the image vectors were stored in mvx and mvy respectively. Therefore, mvx(i, j) stores the x-component of the disparity vector corresponding to (i,j) -th and mvy (i,j) stores the y-component of the disparity vector corresponding to the particular block.

The algorithm to estimate the disparity is –

Step 1 determine the size of the reference frame – refImage.

Step 2 calculate the size of disparity vectors mvx and mvy, based on the size of refImage, M and N

Step 3 for each 8x8 block in refImage do

i. Extract the block from refImage
ii. Search the second image, over a region of ±R pixels relative to the position of the current block and calculate the sum of absolute difference between the pixels of the reference image and the second image. The minimum SAD is the best matching block.
iii. The difference between the reference image and the second image is the disparity vector for the two images. This value is stored in the appropriate locations in mvx and mvy.

3. Decoding the Image

To decode the image, all the steps in encoding were performed in reverse order.

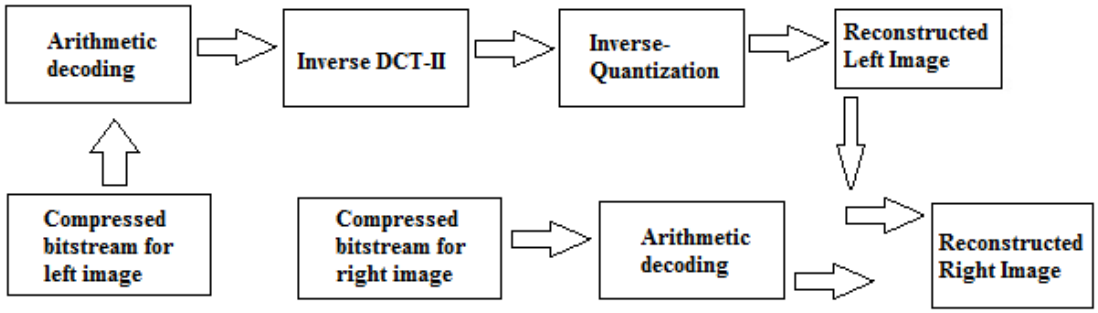


Figure 3: Reconstructing the Image

3.1 Arithmetic Decoding

The pseudo code for arithmetic decoding is –

```
low <- 0;
high <- 1;
x <- encoded number
while (Sn!=0) do
    Sn<- Decode next symbol in x;
    Print Sn;
    x <- (x-CDF(n-1))/(CDF(n)-CDF(n-1));
end
```

3.2 Inverse Quantization

The image matrix is divided into 8x8 blocks and each block is multiplied by the Quantization matrix in (3). This is step is referred to as inverse quantization even though quantization cannot be inversed as it is lossy [6].

3.3 Inverse Transform

The image is divided into NxN blocks and the inverse 2-D DCT of each pixel is calculated as –

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \quad (6)$$

The number 128 was subtracted from each pixel while calculating the forward DCT-II, therefore 128 is now added to each pixel value.

3.4 Re-constructing the Second Image

To reconstruct the second image, we need image matrix of the reconstructed left image and the disparity vectors calculated. The algorithm for reconstructing the right image is –

Step 1 Calculate M and N, based on the size of the reference image and mvx.

Step 2 Initialize disparity compensation vector (disp_comp) and second image (secImage) as the same size as the reference image (refImage).

Step 3 For each block in the reference image, do
   i. Extract the current block from reference image
   ii. Extract the disparity vectors corresponding to the current location.
   iii. Add the disparity vectors to the current block

4. Results

The decoded left and right images were compared with the original left and right images. The Mean Square Error (MSE) between the original and decoded left and right images was calculated as –

$$\text{MSE} = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}\left[I(x,y) - I'(x,y)\right]^2 \quad (7)$$

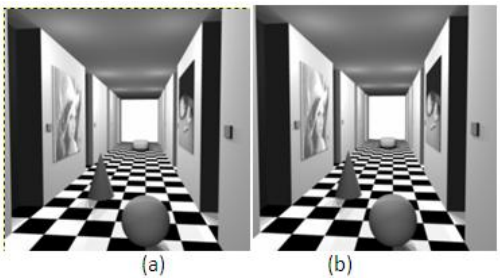
The MSE of the image is the average of the MSE of the left image and the MSE of the right image.

$$\text{MSE} = (\text{MSE}_L + \text{MSE}_R)/2 \quad (8)$$

The MSE was converted into Peak-Signal to Noise Ratio according to the formula –

$$\text{PSNR} = 10\log_{10}\frac{255^2}{\text{MSE}} \quad (9)$$

4.1 Corridor



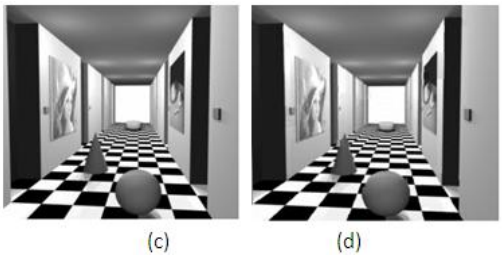


Figure 4 : (a) and (b) Original Left and Right Images of Corridor (c) and (d) Reconstructed Left and Right Images of Corridor

Table 1 – Data for Corridor

Quality	Total Size (2 bitfiles + 1 header file)	Bitrate	PSNR(dB)
30	65359	0.498649597	33.2861
40	72709	0.554725647	34.3359
50	79594	0.607254028	35.1582
60	86709	0.66153717	35.8647
70	97058	0.740493774	36.9033
80	112667	0.859580994	38.2231
90	143378	1.093887329	40.081

The PSNR vs bitrate graph is given below –

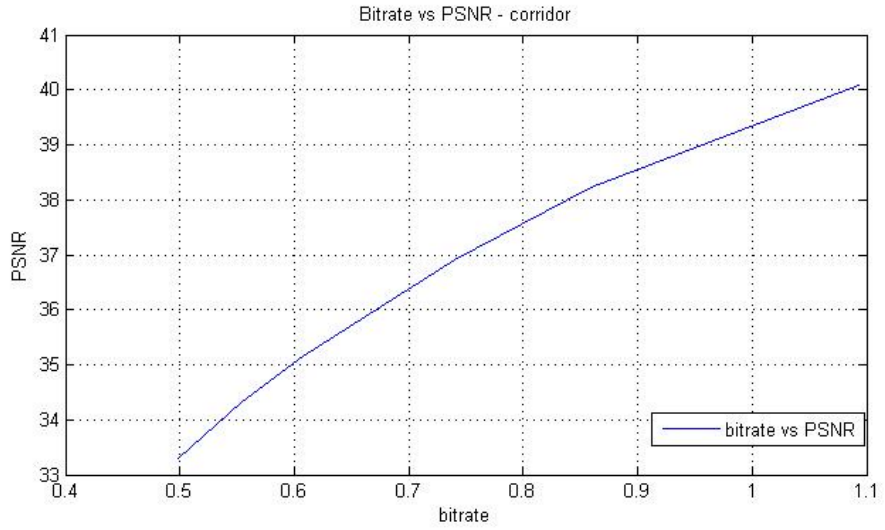


Figure 5 – PSNR vs Bitrate for Corridor Image

4.2 Tsukuba

Figure 6 : (a) and (b) Original Left and Right Images of Tsukuba (c) and (d) Reconstructed Left and Right Images of Tsukuba

Table 2: Data for Tsukuba

Quality	Total Size (2 bitfiles + 1 header file)	Bitrate	PSNR (dB)
30	96662	0.437020761	32.7579
40	113304	0.512261285	33.60646
50	128567	0.58126718	34.3472
60	144045	0.651245117	35.0353
70	166062	0.750786675	36.0063
80	200820	0.907931858	37.2164
90	274019	1.238873517	39.1148

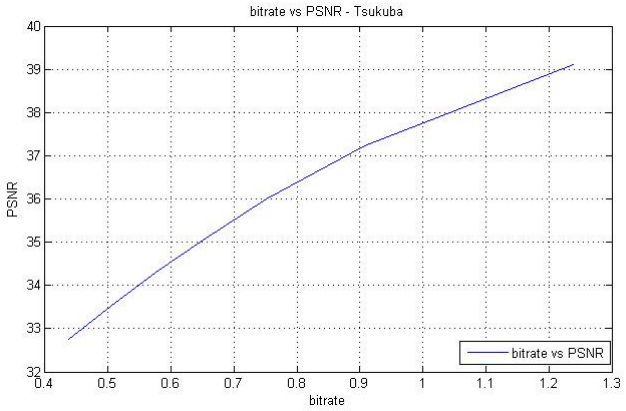


Figure 7: PSNR vs bitrate for tsukuba image

4.3 Rocks

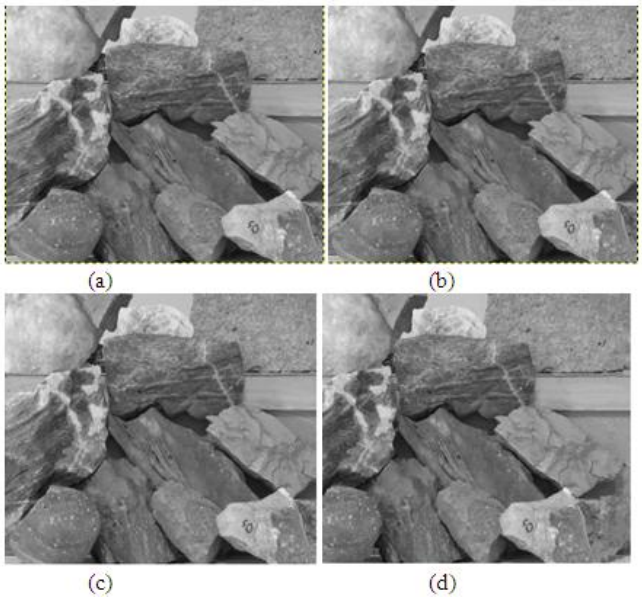


Figure 8 : (a) and (b) Original Left and Right Images of Rocks
(c) and (d) Reconstructed Left and Right Images of Rocks

Table 3: Data for Rocks

Quality	Total Size (2bitfiles + 1 header file)	Bitrate	PSNR(dB)
30	91327	0.483395791	31.7651
40	109095	0.5774422	32.2967
50	125189	0.662628091	32.756
60	141803	0.750566353	33.2211
70	165679	0.876942539	33.8689
80	202564	1.072175644	34.7961
90	279507	1.479436611	36.3321

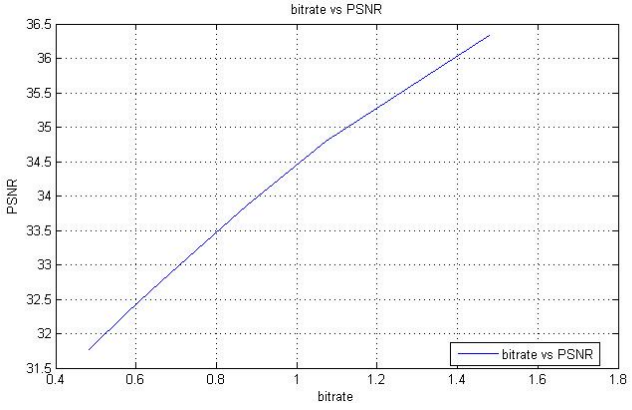


Figure 9: PSNR vs Bitrate for Rocks Image

4.4 Aloe

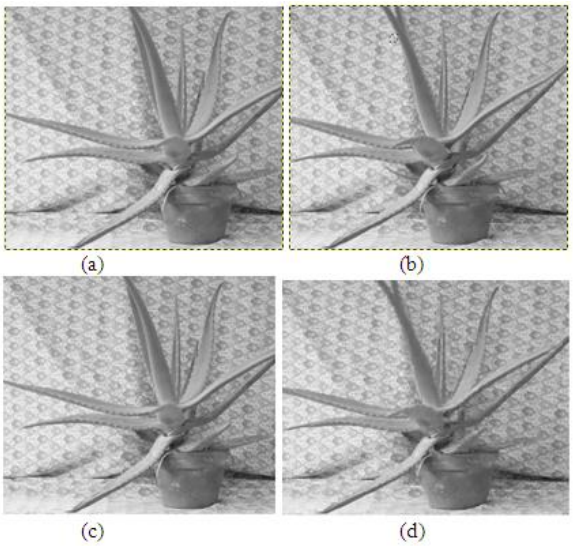


Figure 10 : (a) and (b) Original Left and Right Images of Rocks of Aloe (c) and (d) Reconstructed Left and Right Images of Rocks of Aloe

Table 4: Data for Aloe

Quality	Total Size(2 bitfiles + 1 header file)	Bitrate	PSNR
30	106623	0.564357851	30.1277
40	126190	0.667926406	30.5764
50	143952	0.761941057	30.8876
60	162989	0.862704311	31.2566
70	190925	1.010570164	31.8232
80	234355	1.240446096	32.803
90	321256	1.700414973	34.7952

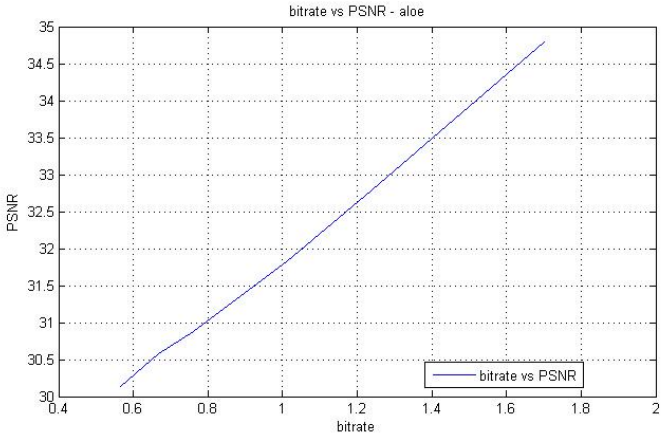


Figure 11: PSNR vs Bitrate for Aloes Image

5. Conclusion

A method to compress stereo images using four algorithms – DCT-II, quantization, arithmetic encoding and exhaustive block matching algorithm was proposed in this paper. Four pair of images were compressed and then reconstructed by reversing the steps followed to compress the images.

The reconstructed images were then compared with the original images.

References

[1] W.Beil andI.C.Carlsen, "Surface reconstruction from stereoscopy and "shape from shading" in SEM images" in Machine Vision and Applications, 1991, pp 271-285

[2] A.Karthik, S.Chandra, B.Ramamoorthy and S.Das, "3D Tool Wear Measurement and Visualization Using Stereo Imaging" in International Journal of Machine Tools and Manufacture, 1997, pp 1573-1581

[3] J.Cardenas-Garcia, "3D reconstruction of objects using stereo imaging" in Optics and Lasers in Engineering, 1995, pp 193-213

[4] S.Sengupta, "Effects of unequal focal lengths in stereo imaging" in Pattern Recognition Letters, 1997, pp 395-400

[5] G.Strang "The Discrete Cosine Transform" in SIAM Review, 1999, pp 135-147

[6] S.Bique "New Characterizations of 2D Discrete Cosine Transform" in IEEE Transactions on Computers, 2005, pp 1054-1060

[7] A.Gersho and R.M.Gray "Vector quantization and signal compression", Springer, 1991

[8] C.Y.Wang, S.M.Lee and L.W.Cheng, "Designing JPEG quantization tables based on human visual system" in Signal Processing: Image Communication, 2001, pp 501-506

[9] W.Niehsen "Fast full-search block matching" in IEEE Transactions on circuits and systems for video technology, 2001, pp 241-247

[10] Y.Wang, J.Ostermann, Y.Q.Zhang (2001) "Video Processing and Communications", Prentice Hall, Upper Saddle River

[11] N.S.Sulthana, M.Chandra "Image Compression with Adaptive Arithmetic Coding" in International Journal of Computer Applications, 2010, pp 31-34

Author

Ashish Agarwal

He was born in Kanpur, India in 1989. He is currently pursuing is BASc in Computer Engineering from Simon Fraser University in Vancouver, Canada.

He has had a passion for computers since he was a child and his research interests include image processing, cryptography and computer networks