Census Transform Histogram (CTH)

Rishiraj Mukati (0801CS171063)
rrmukati98@gmail.com

December 20, 2020

Contents

Chapter 1

Introduction

1.1 Histogram

The histogram is the most commonly used structure to represent any global feature composition of an image. It is invariant to image translation and rotation, and normalizing the histo- gram leads to scale invariance. Exploiting the above properties, the histogram is very useful for indexing and retrieving images.

1.2 Census Transform

The Census Transform R(P) is a non-linear transformation which maps a local neighborhood surrounding a pixel P to a binary string representing the set of neighboring pixels whose intensity is less than that of P.

It is a non-parametric transform that depends only on relative ordering of intensities, and not on the actual values of intensity, making it invariant with respect to monotonic variations of illumination, and it behaves well in presence of multimodal distributions of intensity, e.g. along object boundaries.

The most common version of the census transform uses a 3x3 window, comparing each pixel ***p*** with all its 8-connected neighbours with a function ξ defined as

$$\xi(p, p') = \begin{cases} 0 & \text{if } p > p' \\ 1 & \text{if } p \leq p' \end{cases}$$

The results of these comparisons are concatenated and the value of the transform is an 8-bit value, that can be easily encoded in a byte.

$$\begin{array}{|c|c|c|} \hline 124 & 74 & 32 \\ \hline 124 & 64 & 18 \\ \hline 157 & 116 & 84 \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|} \hline 1 & 1 & 0 \\ \hline 1 & x & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \longrightarrow 11010111$$

Chapter 2

Mathematical Formulation

2.1 Formulation

2.1.1 Border padding (image preparation)

The image which is to be transformed is prepared by padding it with borders of white pixels of length half the window size taken.
Since the window size is an odd integer, therefore,

Border Length = Window Size // 2

2.2 Array for transform

Now since the image is padded with border of half the window size, the size of the transformation should also be change to keep the resolution same as original image,
Therefore an array with all elements zero is made of dimensions.

(H - BL * 2) * (W - BL * 2)

Here, H=Height, W=Width, BL=Border Length

Chapter 3

Algorithm

3.1 Preprocessing

Algorithm 1: Preprocessing Image

Input : Image and Window size.
Output : Image (preprocessed).

---

1. Calculate Half window size from Window size.

   Half window size = Window size // 2
2. Create borders of white pixels of length *half window size* on all four sides.
3. Convert image into grayscale image.
4. Return image(preprocessed).

3.2 Census Transform algorithm

Algorithm 2: Census Transform algorithm

Input : Image(preprocessed) and Window size.
Output : Image (census transformed).

1. Calculate Half window size from Window size.
   HWS = Window size // 2
2. Store width and height of image. **W, H**
3. Convert image to an array of pixel intensity.
4. Initialize transform array of size **(W-HWS*2) * (H-HWS*2)** and data type uint8.
5. Define array of Centre Pixels from original image array, which is offset by **(HWS,HWS)** and is of size equal to the transform array.
6. Create an array of offsets of non-center pixels from center pixels.
7. Pixel Comparison
   a. For each pixel in center pixel array, compare it to non-center pixels using offsets for non-center pixels.
   b. If intensity of non-center pixel is greater than center pixel, append 1bit in corresponding element of transform array.
   c. Shift the bit to the left before moving to the next non-center pixel for comparison.
8. Convert transform array in transform image.
9. Return transform image.

Chapter 4

Documentation of API

4.1 Package Organisation

CensusTransform.py :

class CensusTransformHistogram (win_size=3)

Parameters:
win_size (default 3): Preferred window size, if not passed, default 3 is used

4.1 Methods

preprocess(img):

Takes an image as parameter and returns grayscale and bordered image with border size half of that of window.

Input:
img : Image that is to be preprocessed.

Returns:
gry_img : Grayscale and bordered image.

imghistogram(img, imgtitle):

Takes an image as parameter and converts it into a histogram of figure size (15,15) with title *imgtitle*.

Input:
img : Image that is to be converted into histogram.
imgtitle : Title tob given to that histogram.

censustransform(img):

Takes an image as parameter and converts it into a image of its Census Transform.

Input:
img : Image whose census transform is to be calculated.

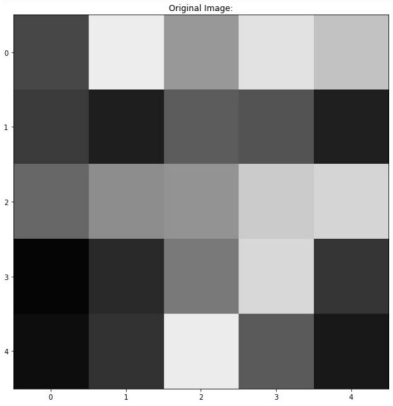
Returns:
out_img : Image of census transform of input image.

Chapter 5

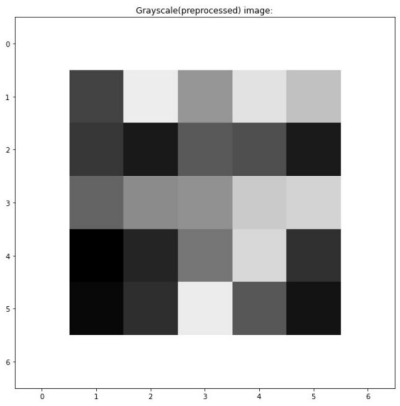
Examples

5.1 Example 1

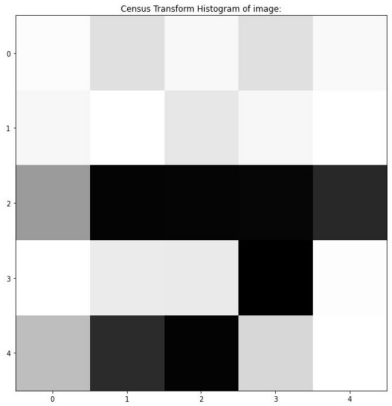
```
#generating random array
rand_arr = np.random.randint(255, size=(5, 5))
rand_img = Image.fromarray(rand_arr)

print('Random generated array:\n\n', rand_arr)

#window size
WS3 = CTH(3)

processedimg = WS3.preprocess(rand_img)
ct_img = WS3.censustransform(processedimg)

print(rand_arr)

#histogram of original image
WS3.imghistogram(rand_img, "Original Image:")

#histogram of preprocessed image
WS3.imghistogram(processedimg, "Grayscale(preprocessed) image:")

#Census transform histogram of image
WS3.imghistogram(ct_img, 'Census Transform Histogram of image:')
```



Original image



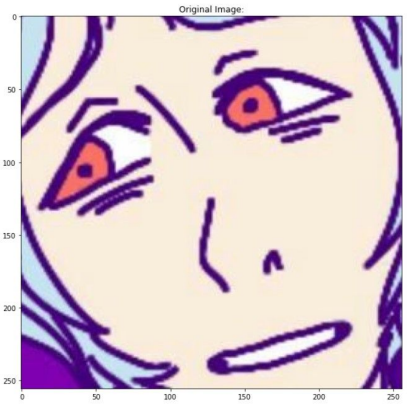
Preprocessed(bordered) image



Census Transform of image

5.2 Example 2

```
image = Image.open('images/ex1.jpg')

#window size
WS5 = CTH(5)
processedimg = WS5.preprocess(image)
ct_img = WS5.censustransform(processedimg)

#histogram of original image
WS5.imghistogram(image, "Original Image:")

#histogram of preprocessed image
WS5.imghistogram(processedimg, "Grayscale(preprocessed) image:")

#Census transform histogram of image
WS5.imghistogram(ct_img, 'Census Transform Histogram of image:')
```



Original image

Preprocessed(bordered) image

Census Transform of image

Chapter 6

Learning Outcomes

- Understanding Census Transform.
- Image handling and histogram plotting with PILLOW library.
- Knowing vast application potential of Census Transform and Its variations.
- Census Transform variations that are currently in use.

Appendix A

References

Bibliography

1) Tavera-Vaca, C.-A.; Almanza-Ojeda, D.-L.; Ibarra-Manzano, M.-A. (2015). Analysis of the efficiency of the census transform algorithm implemented on FPGA. Microprocessors and Microsystems, (), S0141933115001155–. doi:10.1016/j.micpro.2015.08.002
2) Rasmussen, Carl Edward; Bülthoff, Heinrich H.; Schölkopf, Bernhard; Giese, Martin A. (2004). [Lecture Notes in Computer Science] Pattern Recognition Volume 3175 || Efficient Computation of Optical Flow Using the Census Transform. , 10.1007/b99676(Chapter 10), 79–86. doi:10.1007/978-3-540-28649-3_10
3) Froba, B.; Ernst, A. (2004). [IEEE Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. - Seoul, Korea (17-19 May 2004)] Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings. - Face detection with the modified census transform. , (0), 91–96. doi:10.1109/afgr.2004.1301514
4) Perri, Stefania; Corsonello, Pasquale; Cocorullo, Giuseppe (2013). Adaptive Census Transform: A novel hardware-oriented stereovision algorithm. Computer Vision and Image Understanding, 117(1), 29–41. doi:10.1016/j.cviu.2012.10.003