Tampere University

Mikko Impiö

SGN-31007 PROJECT WORK: SINGLE IMAGE SUPER-RESOLUTION FROM TRANSFORMED SELF-EXEMPLARS

Course project
April 2020

CONTENTS

1 INTRODUCTION

This is a project work report for the Tampere University course SGN-31007, Advanced Image Processing. The course covers more advanced image processing techniques, such as common transforms (i.e. wavelet, DCT), compression and restoration methods. The individual project works consist of common problems in image processing, such as denoising, deblurring, inpainting, and super-resolution. The last one, super-resolution (SR), is the topic of this project report, where the main focus is a study by J.-B. Huang, Singh and Ahuja [1].

The paper, Single Image Super-resolution from Transformed Self-Exemplars, published in CVPR 2015 proposes a novel method of self-similarity based super-resolution by transformed nearest-neigbor patches. J.-B. Huang, Singh and Ahuja propose a method of using affine transformations to increase the internal patch search space, and expand this space further by detecting planes to estimate the perspective deformations of these patches. The authors modify the PatchMatch algorithm [2] to estimate a nearest-neigbor-field (NNF) that takes to account affine transformations and perpective corrections based on detected planes from an image. For each patch, a transformation matrix $\mathbf{T}$ is calculated, that is used to retrieve a similar patch from the input image to produce a estimate of a super-resolution image SR. Authors argue, that this method performs better than searching for a NNF using only translational patch search.

This project report consists of four main parts:

1. Literature review
2. Overview and explanation of the algorithm and provided code
3. Experiments with own images
4. Comparison with other methods

Because the area of super-resolution was new to me, a short literature review was performed. This goes through the most commonly used methods, overview of the history of SR, and how the paper by J.-B. Huang, Singh and Ahuja compares to others.

The overview goes through how the method proposed by J.-B. Huang, Singh and Ahuja works, with details on the code the authors have provided (`https://github.com/jbhuang0604/SelfExSR`). Because the authors had provided a ready made implementation, it was not seen necessary to invent the wheel again by fully implementing the algorithm again. The code provided by the authors was modified to produce a simplified implementation of the

algorithm, and to include helpful visualizations that the original code was lacking.

The modified implementation is then tested on own images to produce super-resolved versions of them. Qualitative and quantitative analysis is performed on several images of different resolutions and SR scales. Finally, comparisons are made to bilinear interpolation and a state-of-the art SRGAN-method [3]. Qualitative and quantitative comparisons are also performed at this stage.

In addition to this report, a presentation was made on the findings of this project. The presentation was performed 15.4.2020 to other students attending the course.

2 LITERATURE REVIEW

This chapter consists of a short literature review on the state of super-resolution research. The review focuses on the background of the paper by J.-B. Huang, Singh and Ahuja and how it compares to other popular techniques used in SR. The paper itself is from the year 2015, and a lot of progress has happened during the five years since the publication. The history of super-resolution research and the current state-of-the-art methods are also discussed shortly.

A few general overview studies have been done on super-resolution. The most recent one by W. Yang et al. [4] from 2019 discusses the state of deep-learning based super-resolution methods. The article focuses on architectures and comparisons between single-image super-resolution (SISR) algorithms based on deep learning. Most of the studies mentioned here focus on SISR methods, however, the field of SR extend beyond single-image methods to include also multi-image methods. A historical overview by J. Yang and T. Huang [5] addresses most super-resolution techniques and their history, including also multi-image techniques. An older study by Nasrollahi and Moeslund [6] also discusses several SR methods, including an helpful taxonomy of different methods. A simplified taxonomy of different methods is presented in figure 2.1, based on [4] and [6].

As can be seen from figure 2.1, the paper by J.-B. Huang, Singh and Ahuja is based on super-resolution performed using only internal patch dictionaries. This means that the SR image is produced using only internal patches. The use of similar patches to produce the SR image makes the method an *learning-based* method. Internal or external patches are used to learn a mapping from a low-resolution (LR) patch to a high-resolution (HR) one [7]. The method of using external libraries is substantially more used and well-researched Ledig et al., Chang, Yeung and Xiong, C. Dong et al., Freeman, Jones and Pasztor, J. Yang et al., than self-similarity (internal library) based methods Glasner, Bagon and Irani, Ebrahimi and Vrscay, Singh and Ahuja, Zontak and Irani.

Self-similarity based methods, such as the one proposed by J.-B. Huang, Singh and Ahuja, are often based on the fractal nature of images. A book by Barnsley [15] discusses the fractal nature of images. Studies by Ebrahimi and Vrscay [12], and Glasner, Bagon and Irani [7] are one of the first studies using self-similarity and internal patches to produce useful super-resolution images. After this, methods based on sparse representations [11] and sub-band self-similarity [13] have been proposed.

External library based methods, especially ones based on deep-learning have gained

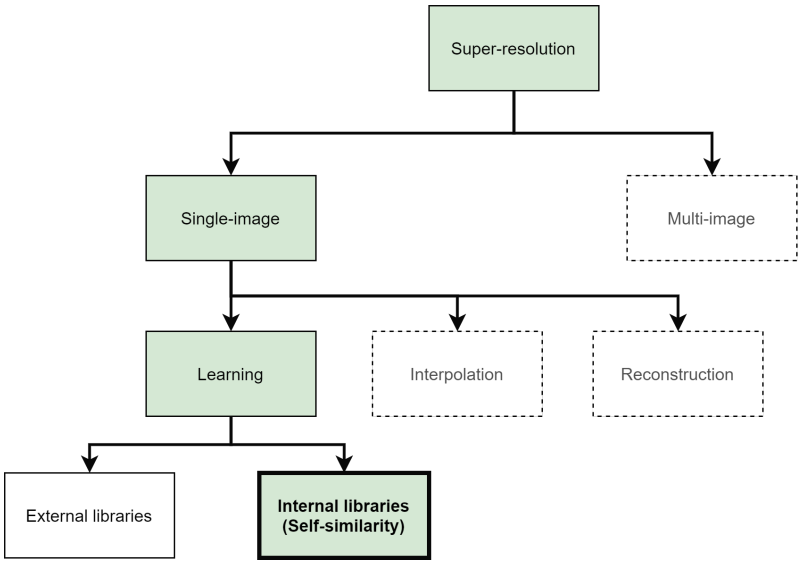


Figure 2.1. *Simplified overview of SR method taxonomy. The area of paper [1] is highlighted in green*

more attention in the past few years. An early study by Chang, Yeung and Xiong [8] use external training images to learn mappings between LR and HR patches. An earlier study by Freeman, Jones and Pasztor uses Markov random fields and a nearest neighbor approach to produce SR images. A substantial step forward was done by Dong, Loy and He [16], with a proposed convolutional neural network (CNN) based approach. CNN based approaches have been producing state-of-the-art results in many benchmarks since. Kim, Kwon Lee and Mu Lee [17] used a recursive CNN to produce results still regarded as very good, and recently Zhang et al. [18] proposed a residual CNN based approach. Generative adversial network based approaches have been proposed for example by Ledig et al. [3], which is also used as a reference point in the comparisons presented later.

Historically, SR techniques have included also interpolation and reconstruction based techniques, where bilinear filtering, bicubic filtering [19] and Lanczos filtering [20] have been fast and popular interpolation methods in the past. Interpolation techniques do not however attempt to construct high-frequency parts of images, and many do not consider them to be SR techniques at all [6].

3 OVERVIEW OF THE METHOD

This chapter goes through the super-resolution technique presented by J.-B. Huang, Singh and Ahuja in their paper Single Image Super-resolution from Transformed Self-Exemplars. This chapter is divided to two sections. The first section reviews the method similarly as in the original paper, but in a more simplified manner. The second section goes through the implementation details of the algorithm, including details not discussed in the original paper. Helpful visualizations are added to demonstrate the method better.

The text in this chapter is based on the original paper [1] unless stated otherwise. The implementation of the algorithm is provided by the original authors at `https://github.com/jbhuang0604/SelfExSR`, which is used as a reference point for the implementation explanantion. Because the implementation was extensive and heavily based on the fairly large codebase of the PatchMatch algorithm, it was not seen necessary to implement it again. Instead, the authors' implementation is reviewed in detail and additional visualization code is written to produce more insight.

3.1 Overview

The main idea behind the original paper is using transformed self-similar patches as reference when creating a super-resolved image. Traditionally methods that use similar patches occurring inside an image use only translational patch search, which might not find the most suitable patches from the image. Often self-similar areas are transformed in some way, meaning that the patch search space should be expanded to include more complex transformations. The figure 3.1 demonstrates this difference in search spaces.

The general overview of the method can be seen in figure 3.2. The high-level steps in the algorithm are:

1. A LR input image is downsampled to produce downsampled, low-pass image of the original.
2. For each 5x5 patch in the image, a best-matching patch in the downsampled image is searched for. The best matching patch is found using a nearest-neigbor-field, where the nearest-neigbor patch is the one that minimizes the objective function $\sum_{i \in \Omega} E_{app}(\mathbf{t}_i, \theta_i) + E_{plane}(\mathbf{t}_i, \theta_i) + E_{scale}(\mathbf{t}_i, \theta_i)$. The NNF also determines the transformation matrix $\mathbf{T}_i$ that is performed to obtain the best matching patch.
3. The best matching patch is then extracted from the original image, from the same

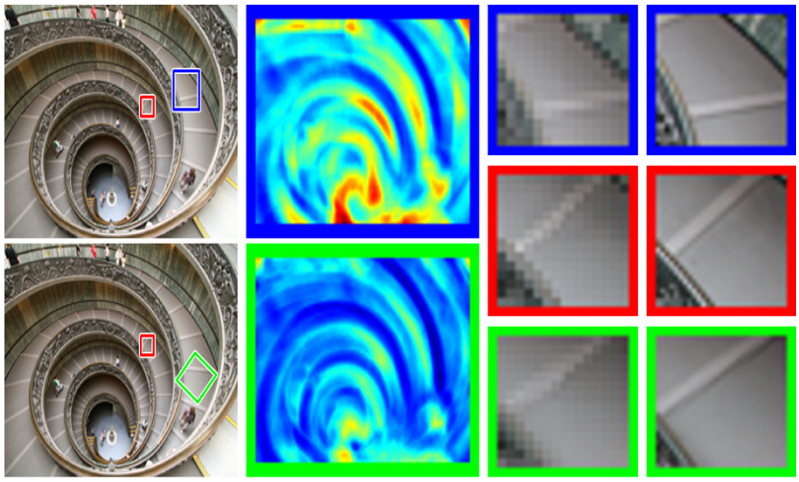


Figure 3.1. *Blue demonstrates the best found translational patch corresponding to the original red patch, and green the best found patch using an affine transformation. Middle column shows the matching error. It can be seen, that affine transformation can find a better matching patch. [1]*

spot as in the downsampled image

4. The patch from the LR image is inverse transformed to the same shape as the reference patch, and is pasted on the SR image.
5. The process is repeated for all patches to produce an estimate of the SR image, which is further backprojected to produce an final image.

The process is very similar to the PatchMatch algorithm [2], which attempts to find best patches in an image using *propagation* and *random search* [21]. Propagation steps attempt to improve a found nearest neighbor by searching for other neighbors in the same direction as previously best NNs were found. Random search tries to find better matches from different parts of the image by selecting an area by random, which is further propagated to find the best matching patch. Note that the "neigbors" in PatchMatch do not have to be necessarily in a spatial dimension, but can be any parametrized space that can be optimized. This notion is used by the authors to produce a custom objective function for the nearest-neighbor-field search, expanding the dimensions to include the transformation matrix parameters θ_i.

An important part of the parameter estimation is the use of prior 3D geometry information extracted from the image. The authors use a technique presented by J.-B. Huang et al. [22], where 3D planes are estimated from the image, and these are used to produce projective parameters for the transformations. The plane that a target patch P in the original image belongs, is marked with m_i in the future. A visualization of plane probabilities can be seen in figure 3.3.

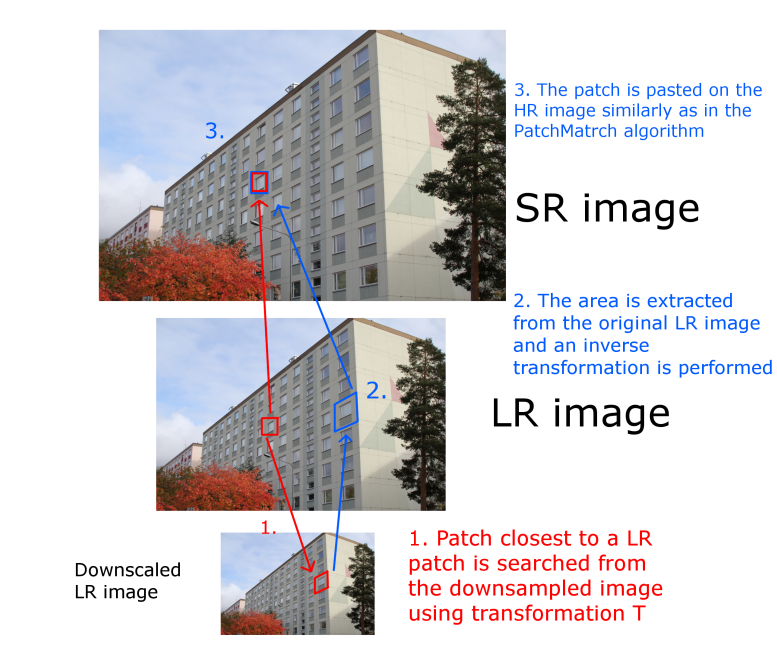


Figure 3.2. *General overview of the proposed SR method. Modified from [1]*

Figure 3.3. *Probabilities of each pixel belonging to three planes marked with cyan, yellow and magenta. Most of the pixels belong in the cyan plane*

Objective function for NNF

The nearest-neigbor-field (NNF) is estimated by the PatchMatch algorithm, and by using an objective function

$$\min_{\theta_i} \sum_{i \in \Omega} E_{app}(\mathbf{t}_i, \theta_i) + E_{plane}(\mathbf{t}_i, \theta_i) + E_{scale}(\mathbf{t}_i, \theta_i) \tag{3.1}$$

Where $\mathbf{t}_i$ is the position of the patch, Ω is the set of all possible pixel indices, and θ_i are the transformation matrix $\mathbf{T}_i$ parameters.

The objective function consists of three parts:

1. Appearance cost
2. Plane cost
3. Scale cost

The most important one is the appearance cost

$$E_{app}(\mathbf{t}_i, \theta_i) = ||W_i(P(\mathbf{t}_i - Q(\mathbf{t}_i, \theta_i)||_2^2 \tag{3.2}$$

That calculates the weighted, squared l^2 norm between the target patch P from the original image and source patch Q in the downsampled image. The cost measures how similar the two patches are, thus having the most importance in finding a similar patch in the downsampled image.

The second cost, plane cost

$$E_{plane}(\mathbf{t}_i, \theta_i) = \lambda_{plane}(Pr[m_i](\mathbf{s}_i^x, \mathbf{s}_i^y) \times Pr[m_i](\mathbf{t}_i^x, \mathbf{t}_i^y) \tag{3.3}$$

takes the planar probabilities to account, giving better costs to neighbors estimated to be in the same plane as the target patch.

Finally, the scale cost is

$$E_{scale}(\mathbf{t}_i, \theta_i) = \lambda_{scale} min(0, SRF - Scale(\mathbf{T}_i)) \tag{3.4}$$

where SRF is the scale factor (for example 2x, 3x, 4x) and $Scale$ denotes the scale estimation of a transformation matrix. The purpose of the cost is to search for patches similar in size as the target patch.

The parameter space θ_i that is searched over, produces the transformation matrix $\mathbf{T}_i$ that maps the target patch P to its nearest neighbor. One option could be to search for nearest neighbors in the full 8D projective transformation space produced by the parameters, but thankfully due to the plane estimation, this can be reduced to a 6D affine transformation space when using the plane parameters.

The matrix can be decomposed now to three transformations:

1. Perspective transformation $\mathbf{H}(\mathbf{t}_i, \mathbf{s}_i^x, \mathbf{s}_i^y, m_i)$, from the plane estimation
2. Similarity transformation $\mathbf{S}(\mathbf{s}_i^s, \mathbf{s}_i^\theta)$ consisting of scaling and rotation
3. Shearing transformation $\mathbf{A}(\mathbf{s}_i^\alpha, \mathbf{s}_i^\beta)$

to produce

$$\mathbf{T}_i(\theta_i) = \mathbf{H}(\mathbf{t}_i, \mathbf{s}_i^x, \mathbf{s}_i^y, m_i)\mathbf{S}(\mathbf{s}_i^s, \mathbf{s}_i^\theta)\mathbf{A}(\mathbf{s}_i^\alpha, \mathbf{s}_i^\beta) \tag{3.5}$$

Here the parameters $\mathbf{s}_i^x, \mathbf{s}_i^y$ can be retrieved from the planar perspective transformation calculated for the full image, leaving only the affine transformation parameters $\mathbf{s}_i^s, \mathbf{s}_i^\theta, \mathbf{s}_i^\alpha, \mathbf{s}_i^\beta$ to be found during the PatchMatch search.

3.2 Implementation of the algorithm

The original authors provided MATLAB code for the implementation and testing of the algorithm. In general, this code was well-readable and highly usable, but the visualizations were broken. Some modifications were made to simplify these scripts, as well as to add some visualizations to illustrate the algorithm. The modified code can be found from `https://github.com/mikkoim/SelfExSR_project`.

This section goes through the implementation of the algorithm, with more details on the implementation part. A top-down approach is taken, by first going through the most high-level functions and then focusing on the lower-level functions. More focus is given to the functions integral to the method presented in the paper. A lot of the code is from the PatchMatch algorithm, so a lot of these details will be only briefly discussed.

3.2.1 High-level implementation

The core of the implementation lies in the `sr_demo`-function. Five major steps are made to produce a super-resolution image from an low-resolution one

1. Reading the image
2. Extracting planes from the image
3. Producing high- and low-frequency image pyramids
4. Patch-based syntesis using PatchMatch
5. Retrieving the correct scale image from a image pyramid produced by the PatchMatch synthesis

In the first part, the image is read and planar structures are extracted from the image. These are the same planes that can be seen in figure 3.3, along with the vanishing point detection. The authors use a separate program, `EdgeDetectTest.exe` to produce the vanishing points. These are then read, and a struct `modelPlane` containing the plane information is created. At this point it only corresponds to a single image scale, but in the next steps the planes are calulated for several scales.

The implementation uses an approach where an image pyramid is calculated for the input image. Here the image is divided to high- and low-resolution bands and downsampled to produce a downscaled version of the image. This is continued to produce images on different scales for both high- and low frequencies. These pyramids are used later to produce the NNFs, with room of upscaling the images. Depending on the scale factor, empty "pyramid slots" are left for different levels of upscaling during the SR progress. Figure 3.4 illustrates the image pyramid.

The fourth and most important part is the patch-based synthesis. This will be studied in depth later. Patch-based synthesis "fills in" the image pyramid's upscaled "slots" to produce upscaled images of several intermediate scales. In the final step, the image

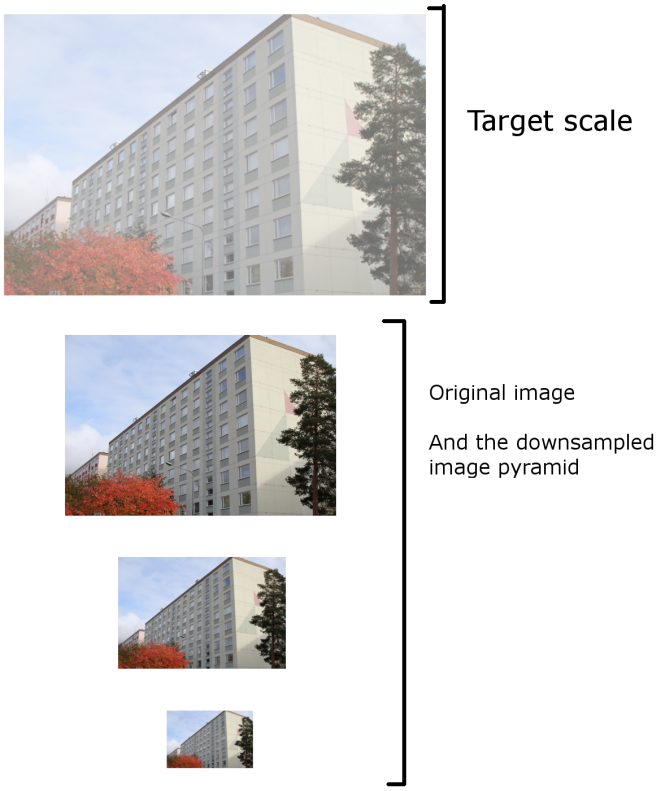


Figure 3.4. *Produced image pyramids. The downscaled images are produced from the original image, but the upscaled ones are produced only by the modified PatchMatch algorithm*

corresponding to the desired scale factor is retrieved from the pyramid and returned by the function.

3.2.2 Patch-based synthesis

The actual super-resolution by patch-based synthesis happens in the `sr_synthesis`-function. Synthesis is performed on a desired amount of upscaling levels determined from the image pyramid. Higher scaling factors need more intermediate synthesis steps. For each scaling level, synthesis can again be divided to different parts:

1. Retrieving the plane parameters for the current level
2. Using these parameters, the current level LR image, and the downsampled images in the pyramid, a NNF is estimated using the modified PatchMatch algorithm.
3. Using the NNF, an upscaled image is created by weighted voting and backprojection

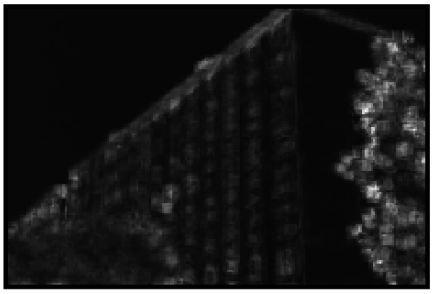
Here the most important step is the NNF estimation that uses the `sr_pass`-function. Another important function is the `sr_voting`-function that produces the actual image by patch-based voting and backprojection. The iterative backprojection algorithm is explained in [23].

The `sr_pass`-function estimates the NNF for the current level of scaling. First, target patches are extracted from the target image. Then the source patches and their corresponding tranformation matrices are initialized. Initial costs are calculated, and the NNF is then updated iteratively.

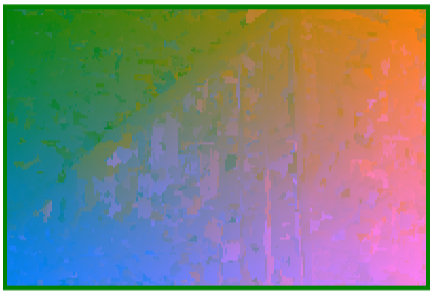
For each iteration, performed by function `sr_update_NNF`, a better NNF is estimated by random search and propagation. Possibly better transformation matrices are created for each target patch, and the area around the tranformations are searched (propagation) to find the best neighbor matches. This is iterated for several iterations to produce a NNF for the scale. The nearest neighbor is chosen so that the cost function 3.1 is minimized.

Figure 3.5 illustrates two different NNF visualizations by cost for each 5x5 patch, and the best patch match by angle (hue) and distance (saturation) for each patch.

Finally, a SR image is created by the function `sr_voting` using the NNF and the high resolution image pyramid. For each target patch, a source patch with the best NNF score is chosen. The resulting estimate is further improved by the backprojection algorithm to produce a final image.



(a) *Cost map. The minimum found cost for each patch*



(b) *Nearest-neigbor field. The nearest neigbor for each patch is illustrated by distance and angle: distance as saturation and hue as the angle*

Figure 3.5. *Different visualizations of the PatchMatch algorithm*

4 EXPERIMENTS AND COMPARISONS

This chapter presents tests and comparisons performed on a custom dataset consisting of images taken by the author of this report. The dataset consists of 12 images chosen to contain both urban and natural scenes, as well as images considered challenging to super-resolution algorithms.

Two ground truth resolutions were used: 1920x1280 and 640x427, roughly corresponding to HD and VGA resolutions. For VGA resolution, scaling factors of 2x and 4x were tested, and for HD, 2x, 4x and 8x. All tests were visually compared to their bilinear interpolation counterparts, as well as bilinear interpolation with sharpening. The HD 4x scaling was also compared to a state-of-the-art algorithm, the SRGAN-method [3]. The weights for the SRGAN neural network were retrieved from `https://github.com/twtygqyy/pytorch-SRResNet.git`.

Figures below visualize some differences in the algorithm performance. PSNR values are added to some of the comparisons. The full images can be found from the repository. It can be seen, that small scale increases like 2x have a similar effect as just bilinear interpolation and sharpening. Better results are gained in 4x SR, when considerable sharpness is gained in geometrically consistent areas, like buildings. With larger scales, like 8x and also 4x for VGA images, undesirable effects arise. A "picasso effect" can be seen in noisy areas. Larger scales also produce a 3D-like effect, with sharp edges but blurry planes. From comparison to SRGAN it can be seen, that SRGAN has better subjective IQ, but SelfExSR performs better in noisy areas, where it does not "hallucinate" unnecessary details.

(a) *High-resolution*

(b) *Bilinear*

(c) *SelfExSR*

(d) *Bilinear with sharpening*

Figure 4.1. *VGA with 2x SR*

(a) *High-resolution*

(b) *Bilinear*

(c) *SelfExSR*

(d) *Bilinear with sharpening*

Figure 4.2. *VGA with 4x SR*

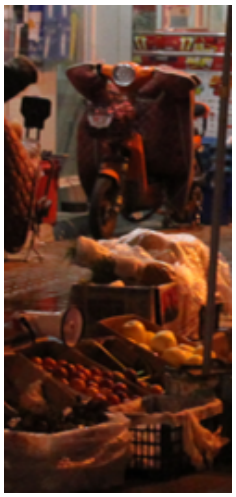
(a) *High-resolution*

(b) *Bilinear*

(c) *SelfExSR*

(d) *Bilinear with sharpening*

Figure 4.3. *HD with 2x SR*



(a) *High-resolution*

(b) *Bilinear*

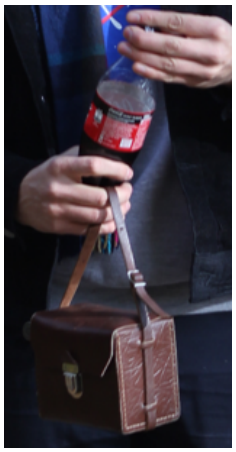
(c) *SelfExSR (PSNR: 31.40)*

(a) *High-resolution*

(b) *Bilinear*

(c) *SelfExSR*

Figure 4.5. *HD with 8x SR*



(a) *High-resolution*

(b) *Bilinear*

(c) *SelfExSR (PSNR: 29.95)*

(d) *SRGAN (PSNR: 29.70)*

Figure 4.6. *HD vs SRGAN with 4x SR*

(a) *High-resolution*

(b) *Bilinear*

(c) *SelfExSR (PSNR: 23.58)*

(d) *SRGAN (PSNR: 22.10)*

Figure 4.7. *HD vs SRGAN with 4x SR*

5 CONCLUSIONS

This report studied the study "Single Image Super-resolution from Transformed Self-Exemplars" by J.-B. Huang, Singh and Ahuja. Overall performance for a single-image super-resolution algorithm was good, especially in geometrically consistent areas. For an algorithm, that uses only self-similarity as its source, this is a fairly good result. Results are almost comparable to the state-of-the-art methods, such as SRGAN, which needs extensive training on large datasets to perform. A downside of SelfExSR is that computation of a single image takes at worst several minutes due to the extensive NNF estimation using PatchMatch. In general the algorithm is one of the best performing SISR methods using only internal dictionaries, competing even with the top deep learning methods.

REFERENCES

[1] Huang, J.-B., Singh, A. and Ahuja, N. Single Image Super-Resolution From Transformed Self-Exemplars. 2015, 5197–5206. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Huang_Single_Image_Super-Resolution_2015_CVPR_paper.html (visited on 04/10/2020).

[2] Barnes, C., Shechtman, E., Finkelstein, A. and Goldman, D. B. PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM SIGGRAPH 2009 papers*. SIGGRAPH '09. New Orleans, Louisiana: Association for Computing Machinery, July 2009, 1–11. ISBN: 978-1-60558-726-4. DOI: 10.1145/1576246.1531330. URL: https://doi.org/10.1145/1576246.1531330 (visited on 04/14/2020).

[3] Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z. and Shi, W. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. 2017, 4681–4690. URL: http://openaccess.thecvf.com/content_cvpr_2017/html/Ledig_Photo-Realistic_Single_Image_CVPR_2017_paper.html (visited on 04/12/2020).

[4] Yang, W., Zhang, X., Tian, Y., Wang, W., Xue, J.-H. and Liao, Q. Deep Learning for Single Image Super-Resolution: A Brief Review. *IEEE Transactions on Multimedia* 21.12 (Dec. 2019). Conference Name: IEEE Transactions on Multimedia, 3106–3121. ISSN: 1941-0077. DOI: 10.1109/TMM.2019.2919431.

[5] Yang, J. and Huang, T. Image Super-Resolution: Historical Overview and Future Challenges. en. *Super-Resolution Imaging*. Ed. by P. Milanfar. 1st ed. CRC Press, Dec. 2017, 1–34. ISBN: 978-1-315-21799-4. DOI: 10.1201/9781439819319-1. URL: https://www.taylorfrancis.com/books/9781439819319/chapters/10.1201/9781439819319-1 (visited on 04/10/2020).

[6] Nasrollahi, K. and Moeslund, T. B. Super-resolution: a comprehensive survey. en. *Machine Vision and Applications* 25.6 (Aug. 2014), 1423–1468. ISSN: 0932-8092, 1432-1769. DOI: 10.1007/s00138-014-0623-4. URL: http://link.springer.com/10.1007/s00138-014-0623-4 (visited on 04/12/2020).

[7] Glasner, D., Bagon, S. and Irani, M. Super-resolution from a single image. *2009 IEEE 12th International Conference on Computer Vision*. ISSN: 2380-7504. Sept. 2009, 349–356. DOI: 10.1109/ICCV.2009.5459271.

[8] Chang, Yeung, D.-Y. and Xiong, Y. Super-resolution through neighbor embedding. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. ISSN: 1063-6919. June 2004, I–I. DOI: 10.1109/CVPR.2004.1315043.

[9] Dong, C., Loy, C. C., He, K. and Tang, X. Image Super-Resolution Using Deep Convolutional Networks. *arXiv:1501.00092 [cs]* (July 2015). arXiv: 1501.00092. URL: http://arxiv.org/abs/1501.00092 (visited on 04/12/2020).

[10] Freeman, W., Jones, T. and Pasztor, E. Example-based super-resolution. *IEEE Computer Graphics and Applications* 22.2 (Mar. 2002). Conference Name: IEEE Computer Graphics and Applications, 56–65. ISSN: 1558-1756. DOI: 10.1109/38.988747.

[11] Yang, J., Wright, J., Huang, T. S. and Ma, Y. Image Super-Resolution Via Sparse Representation. *IEEE Transactions on Image Processing* 19.11 (Nov. 2010). Conference Name: IEEE Transactions on Image Processing, 2861–2873. ISSN: 1941-0042. DOI: 10.1109/TIP.2010.2050625.

[12] Ebrahimi, M. and Vrscay, E. R. Solving the Inverse Problem of Image Zooming Using "Self-Examples". en. *Image Analysis and Recognition*. Ed. by M. Kamel and A. Campilho. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, 117–130. ISBN: 978-3-540-74260-9. DOI: 10.1007/978-3-540-74260-9_11.

[13] Singh, A. and Ahuja, N. Super-Resolution Using Sub-Band Self-Similarity. en. *Computer Vision – ACCV 2014*. Ed. by D. Cremers, I. Reid, H. Saito and M.-H. Yang. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, 552–568. ISBN: 978-3-319-16808-1. DOI: 10.1007/978-3-319-16808-1_37.

[14] Zontak, M. and Irani, M. Internal statistics of a single natural image. *CVPR 2011*. ISSN: 1063-6919. June 2011, 977–984. DOI: 10.1109/CVPR.2011.5995401.

[15] Barnsley, M. F. *Fractals Everywhere*. en. Academic Press, May 2014. ISBN: 978-1-4832-5769-3.

[16] Dong, Loy and He. *Learning a Deep Convolutional Network for Image Super-Resolution | SpringerLink*. 2014. URL: https://link.springer.com/chapter/10.1007/978-3-319-10593-2_13 (visited on 04/10/2020).

[17] Kim, J., Kwon Lee, J. and Mu Lee, K. Deeply-Recursive Convolutional Network for Image Super-Resolution. 2016, 1637–1645. URL: http://openaccess.thecvf.com/content_cvpr_2016/html/Kim_Deeply-Recursive_Convolutional_Network_CVPR_2016_paper.html (visited on 04/12/2020).

[18] Zhang, Y., Tian, Y., Kong, Y., Zhong, B. and Fu, Y. Residual Dense Network for Image Super-Resolution. en. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, June 2018, 2472–2481. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00262. URL: https://ieeexplore.ieee.org/document/8578360/ (visited on 04/12/2020).

[19] Keys, R. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.6 (Dec. 1981). Conference Name: IEEE Transactions on Acoustics, Speech, and Signal Processing, 1153–1160. ISSN: 0096-3518. DOI: 10.1109/TASSP.1981.1163711.

[20] Duchon, C. E. Lanczos Filtering in One and Two Dimensions. *Journal of Applied Meteorology* 18.8 (Aug. 1979). Publisher: American Meteorological Society, 1016–1022. ISSN: 0021-8952. DOI: 10.1175/1520-0450(1979)018<1016:LFIOAT>2.0.CO;2. URL: https://journals.ametsoc.org/doi/abs/10.1175/1520-0450(1979)018%3C1016:LFIOAT%3E2.0.CO;2 (visited on 04/12/2020).

[21] Barnes, C., Shechtman, E., Goldman, D. B. and Finkelstein, A. The Generalized PatchMatch Correspondence Algorithm. en. *Computer Vision – ECCV 2010*. Ed. by K. Daniilidis, P. Maragos and N. Paragios. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, 29–43. ISBN: 978-3-642-15558-1. DOI: 10.1007/978-3-642-15558-1_3.

[22] Huang, J.-B., Kang, S. B., Ahuja, N. and Kopf, J. *Image completion using planar structure guidance*. July 2014. URL: https://doi.org/10.1145/2601097.2601205 (visited on 04/10/2020).

[23] Irani, M. and Peleg, S. Improving resolution by image registration. en. *CVGIP: Graphical Models and Image Processing* 53.3 (May 1991), 231–239. ISSN: 10499652. DOI: 10.1016/1049-9652(91)90045-L. URL: https://linkinghub.elsevier.com/retrieve/pii/104996529190045L (visited on 04/14/2020).