A Comparison of Feature-Combination for Example-Based Super Resolution

Shwu-Huey Yen(✉), Jen-Hui Tsao, and Wan-Ting Liao

PRIA. Laboratory, Department of Computer Science
and Information Engineering, Tamkang University, New Taipei,
Taiwan, Republic of China
105390@mail.tku.edu.tw, 700410268@s00.tku.edu.tw,
602410044@s02.tku.edu.tw

Abstract. Super resolution (SR) in computer vision is an important task. In this paper, we compared several common used features in image super resolution of example-based algorithms. To combine features, we develop a cascade framework to both solve the problem of deciding weights among features and to improve computation efficiency. Finally, we modify the framework to have an adaptive threshold such that not only the computation load is much reduced but the modified framework is suitable to any query image as well as various image databases.

Keywords: Super resolution (SR) · Example-based · Bicubic · Cascade

1 Introduction

Image super-resolution (SR) refers to the process by which a higher-resolution enhanced image is synthesized from one or more low-resolution images. Multiple-frame super resolution uses the sub-pixel shifts between multiple low resolution images of the same scene. The multi-frame SR problem was first addressed in [1], where they proposed a frequency domain approach. But subject to the sequential images are difficult to obtain in reality, the application of multiple-frame super resolution is not widely used. Single-frame SR methods use other parts of the low resolution images, or other unrelated images, to guess what the high-resolution image should look like. There are many single frame SR methods. Sun et al. [2] explored the gradient profile prior for local image structures and applied on SR. Such approaches are effective in preserving the edges in the zoomed image. Assuming that low resolution (LR) image patches and their high resolution (HR) counterparts share a similar geometry, Chang et al. [3] developed super resolution with neighbor embedding. One of the most famous is the example-based super resolution algorithm. As the name suggested, example-based method is to look for most suitable example from pre-prepared database to reconstruct the desired high resolution image. In general, the reconstructed result is database dependent. A database consisting of various examples will provide a better result but cause a burden in searching candidates. Glasner et al. [5] proposed a novel example-based method that does not rely on an external database. However, due to the limited database of examples from original image and its down-sample versions, the result

W.-C. Peng et al. (Eds.): PAKDD 2014 Workshops, LNAI 8643, pp. 744–753, 2014.
DOI: 10.1007/978-3-319-13186-3_66

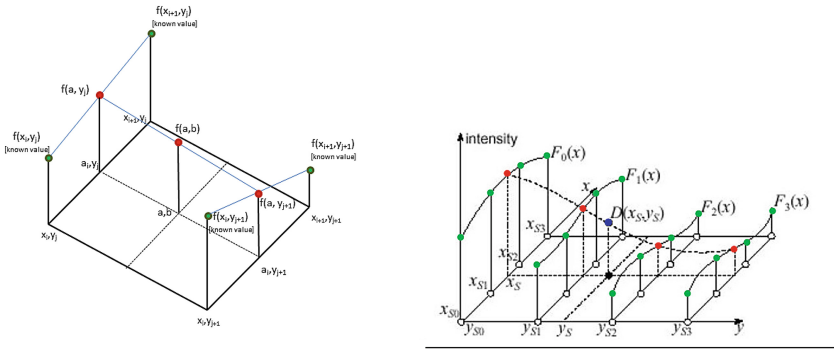
cannot be guaranteed. Presented by Freeman et al. proposed in 2002 [4], it divides a large amount of training images into small patches and uses them in the analytical process. Example-based super resolution is intuitive and simple to implement, the experimental results is also good for the visual, but because of the method proposed by Freeman learned target HR patch of size 5 × 5 from LR input patch of size 7 × 7, as opposed to the method proposed by Chang [3] learned target HR patch of size 12 × 12 from LR input patch of size 3 × 3, the method proposed by Freeman is inferior in terms of efficiency performance. Due to the attractive properties of intuitivism and simplicity, we explore the effectiveness of common used features of SR in the example-based method. In addition, based on the exploration conclusion, a cascade method of magnification factor 4 is proposed.

2 Related Work

Two methods most related to our study are bicubic interpolation and example based super resolution. We present a brief introduction on these methods in the following.

2.1 Bicubic Interpolation

Bicubic and bilinear interpolation are very common methods to resize images. The former is often chosen over the latter when speed is not an issue. In contrast to bilinear interpolation, which only takes 4 (2 × 2) pixels into account (Fig. 1a), bicubic interpolation considers 16 (4 × 4) pixels (Fig. 1b). Images resampled with bicubic interpolation are smoother and have fewer interpolation artifacts.



(a) Bilinear (b) Bicubic(imagefrom[6])

Fig. 1. Interpolation methods

2.2 Example-Based Super Resolution

Example-based super-resolution algorithms involve a training set, which is usually composed of a large number of HR patches and their corresponding LR patches. The target input LR image is split into overlapping patches. Then, for each LR patch from

the input image, one best-matched patch LR patches is selected from the training set. The corresponding HR patch is used to reconstruct the output HR image. Due to the fact that SR is to estimate missing high-resolution detail that is not present in the original LR image, Freeman et al. proposed an example-based method based on a Markov network [4]. The authors embedded two matching conditions into the network. One is that the LR patch from the training set should be similar to the input observed patch, while the other condition is that the contents of the corresponding HR patch should be consistent with its neighbors. They also proposed an one-pass algorithm to improve the performance.

3 Comparison on Various Features

To enlarge an image, the target LR image is divided into patches of size 3 × 3, and for each of these patches, we calculate three types of features: luminance (L), first order derivative (D) and bicubic intensity (B). Assume a 3 × 3 target LR patch, as in Fig. 2, features are described below where $I(p)$ is the intensity value of a point p.

- L: a 9-dim vector, $(I(a), I(b), \ldots, I(i))$
- D: a 18-dim vector, $(\partial I(a)/\partial x), \ldots, \partial I(i)/\partial x), \partial I(a)/\partial y), \ldots, \partial I(i)/\partial y)$
- B: a 144-dim vector of the bicubic result of the LR patch (magnification factor is 4)

a	b	c
d	e	f
g	h	i

Fig. 2. A target LR patch of 3×3

To prepare the database, a collection of sample images of various content are divided into 12 × 12 patches. Figure 3 shows the sample images used in the database (for all the experiments in the paper). To form a patch pair, for each 12 × 12 patch, we pair it with its down-sampled patch of size 3 × 3.

For a target LR image, we first use the Sobel edge detector to locate edge pixels. For every edge pixel P, we take a 3 × 3 patch centered at P so called the target patch. Three types of features (L, D, and B) are evaluated on the LR patches of the database (so called candidate LR patch) and the target patch as well. To find the most similar patch, we compute Euclidean distance of the features from between the target patch and candidate LR patches. Once such patch is found, its counterpart HR patch is adopted to reconstruct the HR version of the target image. As for the overlapped portion, we simply average the pixels. Finally, the metrics SSIM and PSNR are used for evaluation. To testify the effectiveness of features, we perform a series of tests. Figure 4 shows three test images (*Butterfly*, *Girl*, and *Couple*). In most of tests, for time efficiency, only partial images are used as shown on the right of the images.

3.1 Using Only One Feature

To find the most similar candidate patch, we first use only one feature. Table 1 shows the SR results using only single feature on three test images. As observed, feature B performs the best among features. However, it is computation intensive due to the large dimensions.

Table 1. Comparison on single feature

×4	Baby		Butterfly		Couple		Girl	
	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR
Feature L (dim = 9)	0.63	24.2	0.54	17.62	0.77	25.44	0.73	27.86
Feature D (dim = 18)	0.62	24.32	0.52	17.19	0.79	26.17	0.74	28.27
Feature B (dim = 144)	0.65	25.04	0.52	16.68	0.8	26.13	0.74	28.02
Bicubic	**0.75**	**27.57**	**0.64**	**18.12**	**0.91**	**30.4**	**0.8**	**28.92**

3.2 Multiple Features in a Cascade Framework

Before multiple features comparison, we need to decide the way to combine different features. Features usually are linear combined with different weights. How to decide these weights to optimize the performance is a difficult task. A cascade framework can mitigate this problem. We divide the SR process into two or three rounds and each round we only use one feature. For example, if two features are used (i.e., 2 rounds), there will be only top k candidates from first round are kept for further computation for the second feature. By this way, although every patch pair in the database has to be compared with the target patch in the first round, there are only k comparisons required at the second round to find the best HR patch. Similarly, if three features are used, k_1 candidates are kept after the first round, and k_2 candidates are kept from those k_1 candidates after the second round, and finally, in the third round, the best candidate among k_2 candidates is used for SR reconstruction.

With this cascade framework, there are two new problems introduced: the order among these features and parameters k, or k_1 and k_2. For these problems, we design a complete test to find out the appropriate feature order and parameters.

The tests are first on combination of two features. As shown in Table 2, there are six possible combinations. For every target patch, "$Feature_1$-$Feature_2$" means that there are k best candidates kept according to the $Feature_1$ distance between the target patch and every candidate patch of the database. Then, the best candidate according to the $Feature_2$ distance between the target patch and those k candidates is used for SR.

To decide k, we first set k to be 250. As in Table 2, the combinations "L-D" and "B-D" outperform the other 4 combinations in *Butterfly* and *Couple*. Once tests completed, on the best feature combinations "L-D" and "B-D", we again test on

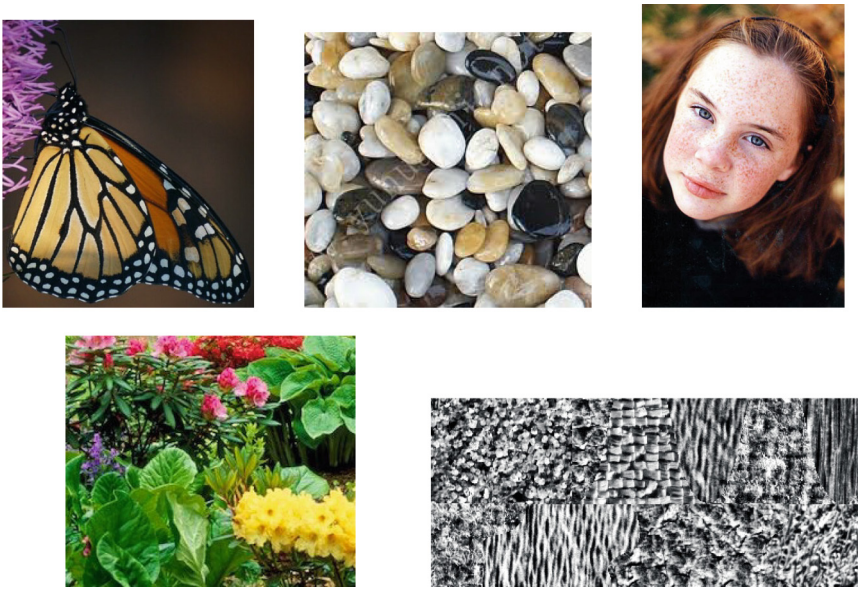


Fig. 3. Training images

Fig. 4. Testing images

Table 2. Two features combinations on k = 250

$Feature_1$-$Feature_2$	Butterfly		Couple	
	SSIM	PSNR	SSIM	PSNR
B-D	0.54	17.41	0.79	26.4
B-L	0.55	17.76	0.78	25.56
D-B	0.53	16.82	0.8	26.25
D-L	0.54	17.54	0.78	25.91
L-B	0.53	16.87	0.80	26.25
L-D	0.54	17.43	0.79	26.43
Bicubic	**0.64**	**18.12**	**0.91**	**30.40**

Table 3. Different k values on features "L-D" and "B-D"

Feature	K	Butterfly		Couple	
		SSIM	PSNR	SSIM	PSNR
L-D	50	0.55	17.56	0.80	26.56
	150	0.54	17.47	0.80	26.37
	250	0.54	17.43	0.79	26.43
	350	0.54	17.37	0.79	26.47
	450	0.53	17.32	0.79	26.41
B-D	50	0.53	17.32	0.80	26.71
	150	0.54	17.44	0.80	26.58
	250	0.54	17.41	0.79	26.40
	350	0.54	17.36	0.80	26.43
	450	0.54	17.36	0.79	26.48
Bicubic		**0.64**	**18.12**	**0.91**	**30.40**

different k values. The results are shown in Table 3. In general, when k = 50 or 150 it has better SSIM or better PSNR. In considering computation cost, 50 is a better choice.

In testing three features, we have done many possible combinations in feature ordering as well as parameters. Table 4 shows some of the results on three features combinations where "$Feature_1$-$Feature_2$-$Feature_3$" indicates the order of features and k_i is the number of best candidates kept after the *i*th round. The results in Table 4 are based on the best combination "L-D" and "B-D" of Table 2. To simplify the possible combinations, k_2 is set to be a dependent variable on k_1. According to the result of Table 3, we consider some possible values of k_1 ranging from 25 to 150. Overall, from Table 4, "L-B-D" with $k_1 = 20$, $k_2 = 5$ has the best performance.

Table 4. Three features combinations

Feature	K_1	K_2	Butterfly		Couple	
			SSIM	PSNR	SSIM	PSNR
L-B-D	25	0.25 * K_1	0.51	16.87	0.75	25.07
	50		0.45	16.35	0.76	25.13
	100		0.42	16.01	0.73	24.4
	150		0.41	15.74	0.71	24.18
	25	0.50 * K_1	0.49	16.62	0.75	24.96
	50		0.45	16.35	0.75	24.74
	100		0.43	16.22	0.72	23.94
	150		0.41	15.64	0.72	24.39
B-L-D	25	0.25 * K_1	0.49	16.53	0.77	25.44
	50		0.46	16.19	0.76	25.42
	100		0.44	16.19	0.74	24.68
	150		0.41	15.84	0.72	24.02
	25	0.50 * K_1	0.46	16.26	0.77	25.54
	50		0.44	15.96	0.74	24.91
	100		0.42	15.94	0.73	24.27
	150		0.41	15.86	0.72	24.05
Bicubic			**0.64**	**18.12**	**0.91**	**30.4**

4 Adaptive Thresholds

In the previous tests, we fixed the number of candidates kept for further examination. However, they may not be suitable if different database or different target LR image is used. To solve this problem, we propose an adaptive threshold T_i so that only those patches whose feature distance is less than T_i will be kept after round i. T_i is defined as

$$T_i = \min_i + n_i \cdot \sigma_i$$

with

$$n_i = \mathrm{MIN}(1, 0.5 \cdot (\mu_i - \mathrm{m}_i)/\sigma_i),$$

where m_i is the minimum feature distance when comparing to the i^{th} feature (i.e., the i^{th} round), μ_i and σ_i are the mean and standard deviation of all feature distances from the candidates for $i = 1, 2$. In particular, at round 1, the candidates are all the patches of the database; at round 2, the candidates are patches with feature distances less than T_1; finally, at round 3, the candidates are those from last round and their feature distances are less than T_2. We apply this adaptive threshold on feature "L-B-D" since it has a good performance according to Table 4.

In order to give privileges to those patches that have small feature distances in the previous round, the previous distance will be carried over to the distance in the current round. To do so, the distance has to be normalized since they have different

Table 5. Comparison of Adaptive and Fixed (L-D, K = 50)

	Adaptive/Fixed K(50)/Bicubic		
Picture	SSIM	PSNR	Time (s)
Baby	0.74/0.74/0.83	26.58/26.60/29.92	182/210/1
Butterfly	0.80/0.80/0.88	23.10/23.12/25.09	303/323/1
Couple	0.80/0.80/0.90	26.65/26.81/29.48	135/151/1
Girl	0.81/0.81/0.87	30.84/30.80/32.51	116/136/1

Fig. 5. Experimental results

dimensions. We first normalize the distance to be within 1 on each dimension. For a 3×3 candidate patch, we use notation dist_i to represent its feature distance comparing to target's after round i. Then, before round i, the initial distance for each candidate patch is defines as

$$\begin{aligned} i = 1: \ & \text{dist}_1 \leftarrow 0, \\ i = 2: \ & \text{dist}_2 \leftarrow \alpha \cdot [\text{dist}_1 / \text{T\_Dist}_1] \cdot (114/9), \\ i = 3: \ & \text{dist}_3 \leftarrow \alpha \cdot [\text{dist}_2 / \text{T\_Dist}_2] \cdot (18/144), \end{aligned}$$

where T_ Dist_i is the sum of all distances in round i. When carrying dist_1 to round two (feature one is L of dim = 9 and feature 2 is B of dim = 144), to be compatible to dist_2, a factor of (114/9) is multiplied in the initialization of dist_2. Similarly, feature 3 is D of dim = 18, a factor of (18/144) is multiplied in the initialization of dist_3.

In Table 5, statistics of SR results from bicubic, fixed (L-B-D, $k_1 = 20$, $k_2 = 5$), and adaptive (L-B-D) are shown. Some of results are given on Fig. 5. As observed, although values in SSIM and PSNR are not as good as bicubic, the reconstructed images of our both methods are visually pleasing and sharper. In comparing "fixed" and "adaptive" methods, they have similar performances but "adaptive" one can suit for different databases.

5 Conclusion and Future Work

In this paper, we explored three common used features in example-based super resolution algorithms and we also developed a cascade framework to solve the weighting problem in feature combination. We also provided an adaptive way in deciding the number of candidates for further checking in a cascade method. By this way, the computation burden in example-based method is much reduced and our method can suit for different database as well. In addition, we utilize the idea of distance initialization to give privileges to those better candidates in successive comparisons.

Because of the rich variability of images, a larger up-sample factor would make the HR patch harder to predict. In the future work, in order to enhance the accuracy of prediction, we will study how to reduce the size of the input patch to make the correct HR patch easier to predict. On the other hand, because example based super resolution methods are susceptible of training images, the study of diversity and availability of training images is also our future concern.

References

1. Huang, T.S., Tsai, R.Y.: Multi-frame image restoration and registration. Adv. Comput. Vis. Image Process. **1**, 317–339 (1984)
2. Sun, J., Xu, Z., Shum, H.-Y.: Steinke: image super-resolution using gradient profile prior. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008, pp. 1–8, 23–28 June 2008

3. Chang, H., Yeung, D.Y., Xiong, Y.: Super-resolution through neighbor embedding. In: Proceedings of the IEEE Computer Society of Conference on Computer Vision Pattern Recogonition, pp. 275–282 (2004)
4. Freeman, W.T., Jones, T.R., Pasztor, E.C.: Example-based super-resolution. Comp. Graph. Appl. **22**(2), 56–65 (2002)
5. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: Proceedings of ICCV (2009), pp. 349–356 (2009)
6. http://software.intel.com/sites/products/documentation/hpc/ipp/ippi/ippi_appendices/Images/ippi_appB_2.jpg. Downloaded on June 2013