Presented at the DLSU Research Congress 2017
De La Salle University, Manila, Philippines
June 20 to 22, 2017

Performance Analysis of a Multiple-Image Super-Resolution Implemented on a Mobile Device

Neil Patrick Del Gallego*, Joel Ilao
De La Salle University, Manila, Philippines
*neil.delgallego@dlsu.edu.ph

Abstract:

Multiple-image super-resolution (SR) attempts to recover a high-resolution image (HR) from a set of low-resolution (LR) images. Multiple-image super-resolution restores high-frequency details and produces an HR image by utilizing pixel differences obtained from multiple LR images.

Processing of multiple images and attempting to produce an HR image can be computationally expensive. Thus, SR systems are normally implemented on desktop PCs. However, mobile devices already have capable hardware and mobile image processing libraries, such as OpenCV, are available. A multiple-image SR system can therefore be implemented on a mobile device if available resources are carefully managed. To prove this claim, a prototype application was implemented on an Android device.

This paper discusses the system architecture of the mobile multiple-image SR system which consists of the following modules: Input Module, Edge Detection Module, Denoising Module, Image Alignment Module, Alignment Selection Module, and Image Fusion Module. In terms of performance time, the Denoising Module has the longest average processing time of 3 minutes and 28 seconds. The Image Alignment Module, Alignment Selection Module and Image Fusion Module has an average processing time of 1 minute each. In terms of memory consumption, the Image Fusion Module consumes the largest amount of memory of 3S(R) where S is the scaling factor and R is the resolution size of the input LR images. A total of 3 matrices are required to perform the image fusion operation. Optimization techniques such as matrix pooling and thread barriers are also discussed.

Keywords: Multiple-Image Super-resolution, Mobile Devices, Matrix Pooling, Thread Barriers

1. Introduction and Related Work

Multiple-image super-resolution (SR) attempts to recover a high-resolution image (HR) from a set of low-resolution (LR) images. The system architecture presented in this paper is an improvement of the framework presented by (Del Gallego & Ilao, 2017). The direct predecessor of this study is from the work of (Chu, 2013) where an SR system for mobile devices was discussed.

Images obtained from mobile devices are subjected to noise, downsampling and motion blur, as visualized in **Figure 1**. An SR technique attempts to reverse these effects to obtain an HR image as close as possible to the real-world scene.

Solving downsampling, suppressing image noise and deblurring images are discussed in the following subsections, as observed from related work.

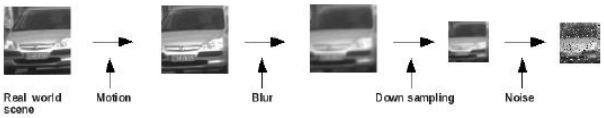

Figure 1: Super-resolution involves the inversion of the image formation process (Mitzel, 2009)

1.1. Solving the Downsampling Problem

Downsampling an image means reducing its original resolution size by a scaling factor of S.

Thus, downsampling a 200 x 200 image by 2 will result in a 100 x 100 image, which removes 10,000 pixels. Given a single image, downsampling can be solved by using known image interpolation methods, namely, Nearest-Neighbor, Bilinear and Bicubic as seen in **Figure 2**.

When multiple images are present, an image fusion can be performed. There are three known image fusion techniques; mean fusion, median fusion (S. Farsiu, 2004), and Shift-Add fusion (Simpkins & Stevenson, 2012). The proposed SR system on a mobile device implements a mean fusion approach as illustrated in **Figure 3**.

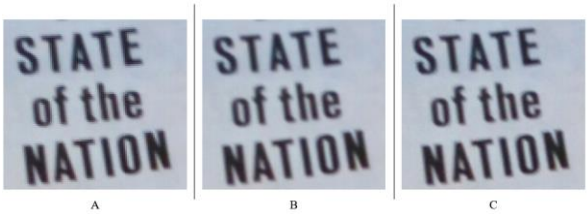

Figure 2: Known interpolation methods. A: Nearest-Neighbor. B: Bilinear. C: Bicubic

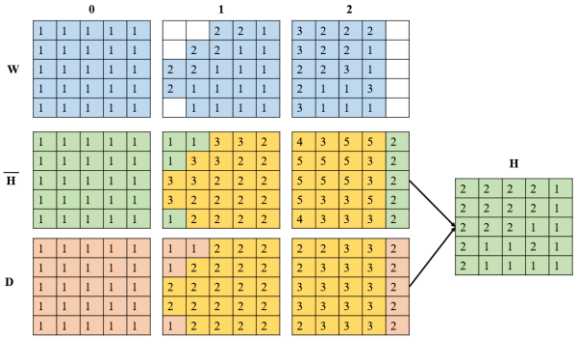

Figure 3: A mean fusion example. Pixel values from warped LR images W are accumulated on matrix $\hat{H}$ and elements in divisor matrix D are incremented accordingly. A per-element division for $\hat{H}$ and D is performed to produce an HR image, H.

1.2. Suppressing Image Noise

Image noise are unwanted artifacts that appear as "grains" or random changes in brightness of pixels. Image noise in mobile devices are caused by low-light conditions.

The most common technique for suppressing image noise is by performing image convolution using a median filter. Unlike local mean filters performed by image convolution, NLM denoising takes into consideration the mean value of all pixels, weighted by how similar the pixels are from the target pixel, supposedly giving the image better clarity and minimal loss of detail (Buades, Coll, & Morel, 2005).

Figure 4 shows an illustration of an image with NLM denoising applied.

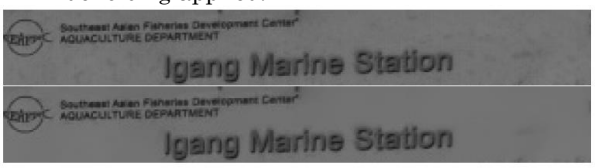

Figure 4: Top image. Monochrome image acquired from a mobile device affected by noise. Bottom image. Image denoised using NLM denoising.

1.3. Image Deblurring

Images captured from mobile devices that are out-of-focus or subjects that produces rapid movement induce motion blurring. An example of an out-of-focus image is observed in **Figure 5**.

Figure 5: The subject becomes out of focus, as seen in image A, if the camera did not properly lock its focus. Image B is noticeably clearer when the camera successfully locked its focus on the subject.

The study of (Chen, He, Yang, & Wu, 2011), (Fergus, Singh, Hertzmann, Roweis, & Freeman, 2006), and (Yuan, Sun, Quan, & Shum, 2007) attempted to deblur single images. However, deblurring images are computationally heavy, aside

from performing super-resolution. Thus it may not be suitable for real-time applications such as imaging applications for mobile devices. However, (Chu, 2013) proposed a simplified approach of deblurring for mobile devices. It has been observed from the experiment of (Chu, 2013) that linear motion deblurring is applicable for mobile devices. As observed in **Figure 6,** inducing camera shake when capturing images from a mobile device creates a linear blur kernel in a slanted dotted line, while a digital camera induces a complex blur kernel. Therefore, (Chu, 2013) concluded that linear motion deblurring is applicable and thus, less computationally expensive for mobile devices.

Figure 6: Image A: Motion blur induced by a digital camera with camera shake. Image B: Motion blur induced by capturing from a mobile device with camera shake.

A simpler alternative would be to carefully select reliable LR images from an initial input LR image set. In the study of (Singh, Lu, Basu, & Mrinal, 2012), the proposed algorithm computes a confidence measure for the LR image sets and assigning a priority value for the individual images. LR images that have the highest priority will maximize the information gain and reduce reconstruction error when processed by a multiple-image SR system. In this study, a feature-selection scheme from (Del Gallego & Ilao, 2017) was used to select reliable LR image sets, by using derivative features suitable for an SR image reconstruction (Yang, Wright, Huang, & Ma, 2010).

1.4. Image Alignment

Prior to performing image fusion, images must be precisely aligned with a reference image. This is to ensure that the correct pixel values are accumulated on the HR grid and misplacement of pixel values are minimized. The most common approach for performing image alignment is by applying image warping to the LR image sets, given a reference LR image, such that key points found across all LR image sets correspond correctly to the same point in the planar surface, as visualized in **Figure 7**.

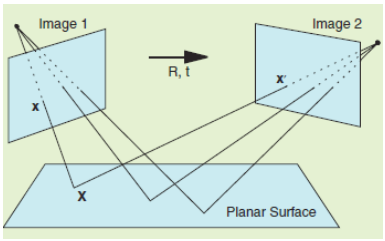

Figure 7: How images should correspond to a certain point in a planar surface (Capel & Zisserman, 2003).

Another approach is to perform Median Threshold Bitmap (MTB) alignment for aligning multiple images (Ward, 2003). According to (Ward, 2003), images captured from handheld devices only require simple translational shifts for aligning the images.

2. System Architecture

A multiple-image SR system was implemented on a mobile device that is guaranteed stable and produces HR images with a *x2* scaling factor[1]. The prototype application is available for Android devices and can be downloaded for free at Google Play Store[2]. The system architecture is shown in **Figure 8**. There are several modules that composes the SR system which are discussed as follows.

The *Input Module* accepts 10 JPEG images from an image gallery application or from images acquired directly from the camera. The *Edge Detection Module* obtains binary edge images from the LR image set using the filters by (Yang, Wright, Huang, & Ma, 2010). A sharpness measure proposed in (Del Gallego & Ilao, 2017) is computed for each LR image that will be used as criteria for

[1] Observable improvement in image quality is observed using x2 scaling factor. x4 scaling consumes high memory and does not work on majority of mobile devices in the market.
[2] Eagle Eye HD Camera: https://play.google.com/store/apps/details?id=neildg.com.eagleeyesr

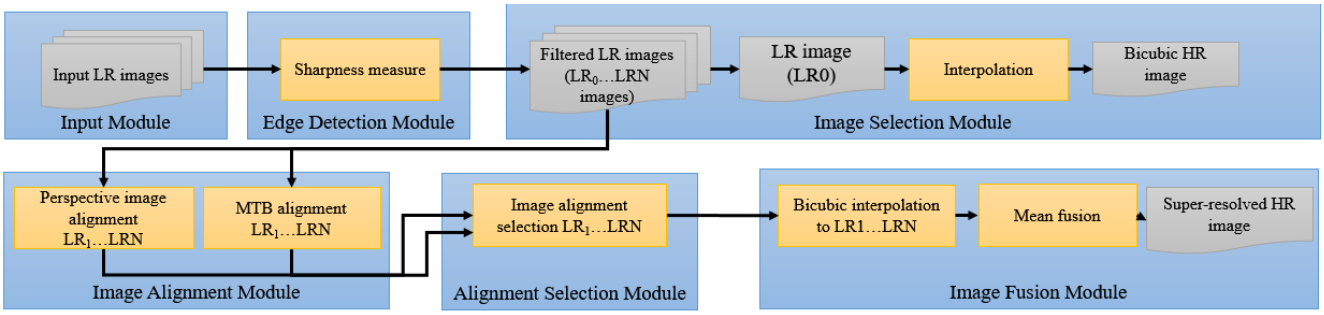

Figure 8: System architecture for the mobile SR system

image selection to be performed by the *Image Selection Module.*

Images that have a higher sharpness measure than the average are selected for HR reconstruction. The first LR image selected by the *Image Selection Module* is upsampled using Bicubic interpolation to produce an initial HR image. It is also at this stage, where an NLM denoising (Buades, Coll, & Morel, 2005) and unsharp masking are performed on selected LR images. Succeeding LR images selected are passed to the *Image Alignment Module*

In the *Image Alignment Module,* the LR images are feature-matched with the first reference LR image and are warped accordingly using a perspective warping approach as discussed in Section 1.4, and another set of warped images are also produced using the MTB alignment technique (Ward, 2003).
To minimize misalignment errors, the *Alignment Selection Module* selects between the two aligned image sets produced by the *Image Alignment Module.* It is observed that misaligned images introduce additional edges when combined to the original reference image. Thus, the edges of the aligned image sets are extracted and a difference value is computed for each.

The least difference is selected as the candidate aligned image set for the image fusion process.

The *Image Fusion Module* accepts the selected aligned image sets and are upsampled using bicubic interpolation. The upsampled images are combined using a mean fusion approach as illustrated in **Figure 3**. This produces the final HR image.

3. Managing Computational Resources

Computational resources are managed using matrix pooling and thread barriers, which guarantee that the SR system is stable and resources are carefully managed.

3.1. Matrix Pooling

Matrix pooling is derived from a software design pattern where a collection of pre-allocated matrices can be retrieved for further use. This ensures faster computation by only instantiating the matrices once upon application startup, and destroying them once the SR task is completed. This also ensures that allocation of memory is fixed for the lifecycle of the application.

Processing modules request for a matrix of fixed size *rows X cols* where *rows* and *cols* refer to the image's height and width respectively and allocate values as needed. Given a *MatrixPool* class, there are three functions: *Initialize(), RequestMatrix(),* and *ReleaseMatrix().*

The *Initialize()* function is called upon application startup where a matrix list of size *N*, with *rows X cols* as the size of the matrices, are instantiated. The elements are populated with zero values. The *RequestMatrix()* requests an available matrix from the pool. It returns a matrix if there are sufficient matrices, otherwise it returns a failure. The *ReleaseMatrix()* is called whenever matrices used by system modules are available for future use by other system modules.

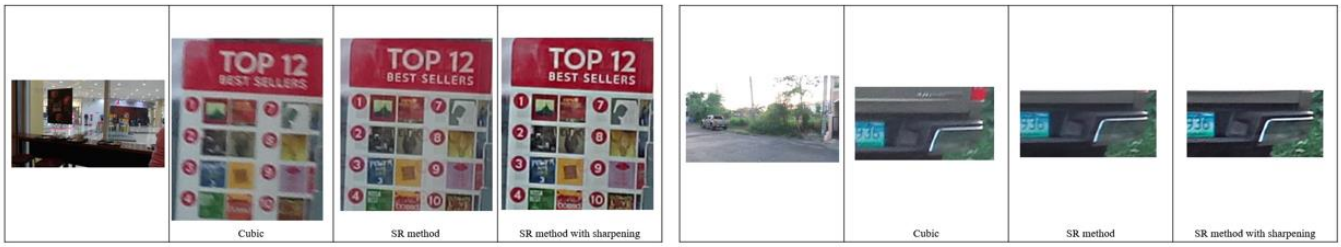

Figure 9: Result for two test images. From left to right: Image thumbnail. Cubic interpolated image. HR image using SR system. HR image with unsharp masking.

Figure 10: Result for two test images. From left to right: Image thumbnail. Cubic interpolated image. HR image using SR system. HR image with unsharp masking.

3.2. Thread Barriers

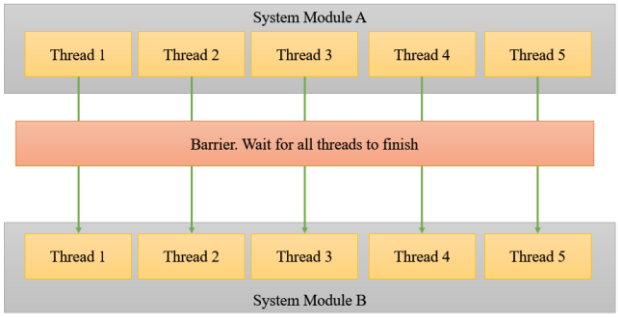

Figure 11: An illustration of a thread barrier.

A thread barrier, illustrated in **Figure 11**, is a concurrency design pattern wherein *N* thread workers executes instructions individually and a thread barrier class checks when all the thread workers are finished before proceeding to the next action. In the SR system, images are first processed concurrently, but at some point, it must be done sequentially (feature-matching and image fusion). Hence, a thread barrier is needed before starting the sequential operation.

4. Image Results and Discussion

Image results are shown in Figure 10 and Figure 11. It is observed that the SR system produces decent HR images and manages to restore high-frequency details. Performing image sharpening using unsharp masking on individual LR images prior to image fusion produces visually better images. In **Figure 9**, the SR system manages to produce clearer and more readable texts. A similar case is observed in the first image shown in **Figure 10**. In the second example in **Figure 10**, while the SR method without sharpening produced a blurry HR image due to misalignment errors, the SR method with sharpening has reduced misalignment errors. This resulted in a clearer HR image.

5. Performance Analysis

This section discusses the time and space complexity of the SR system. A test device was used for benchmarking the prototype application and has the following specifications:

- 2.0 64-bit GHz Octa-core processor. 4GB RAM. 16MP PDAF Rear Camera. Aperture: F/2.0. 5MP Auto-Focus Front-facing camera.

5.1. Performance Time

For assessing the time complexity of the proposed SR method, the actual performance time was measured using a total of 20 test sets, with each set having 10 LR image sequences[3], selected as input on the specified test device. The image sets are 16-megapixel (MP) resolution size of 2992 x 5280 and the up-sampling factor is 2, producing a 50MP HR image of 5980 x 10560.

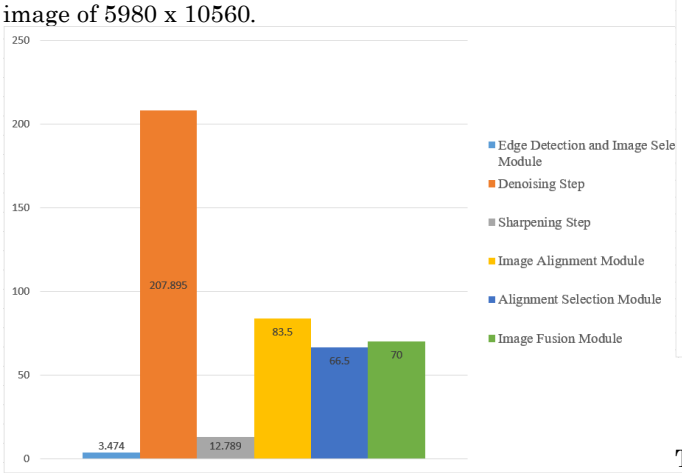

Figure 12: Average performance time of each module in seconds

Figure 12 and **Figure 13** shows the average processing time and the standard deviation of system modules. The *Denoising* feature has the longest average processing time of 3 minutes and 28 seconds, as compared to other modules. While it can be effective on removing noise, the processing time is greatly increased. Depending on the *Image Selection Module,* the quantity of images processed by the Denoising feature varies across test sets, which explains the high standard deviation observed in **Figure 13**.

It is observed that the *Image Alignment Module*, the *Alignment Selection Module* and the *Image Fusion Module* have an average processing time of more than one minute each. Several points of improvements can be deduced from this result.

The *Alignment Selection Module* chooses from two aligned image sets that incurs the least misalignment error. The *Image Alignment* and *Alignment Selection Module* can be simplified if it's guaranteed that images can be aligned correctly using only one alignment technique. In other words, the *Alignment Selection Module* may be removed if the correct alignment of images is guaranteed.

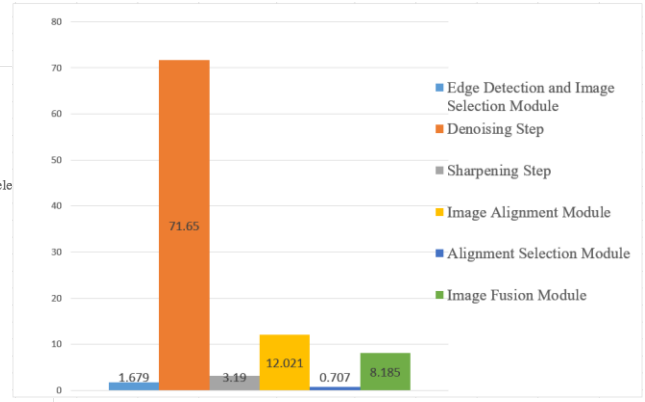

Figure 13: Standard deviation of the performance time of system modules.

The *Image Alignment Module* may be improved such that keypoint detection and feature-matching can be done concurrently since the images are independent, provided that these images can be loaded into the memory safely and at the same time. However, there's a risk of Out-of-Memory errors for low-end mobile devices (those with 1GB RAM or less). Thus, more careful planning of designing concurrency on these modules are needed.

In the *Image Fusion Module*, the specific step that prolongs processing time is performing Bicubic interpolation on all aligned images prior to performing mean fusion. This essentially avoids the need for performing Image Regularization, which is a refinement step according to (Simpkins & Stevenson, 2012). However, it could shorten the processing time if an empty HR grid was created and corresponding pixel values were mapped accordingly, based from the aligned images. Missing pixel values will then be interpolated afterwards, thus the interpolation step is performed only once.

5.2. Space Consumption

Figure 14 shows the summary of allocated space required for each module. The memory

[3] From the experiments conducted, using 10 images yields the best possible results.

MODULE	DATA STRUCTURES	ALLOCATION
Input Module	$S \leftarrow$ file path list	String list of size N
Edge Detection Module	$H \times W \leftarrow$ image resolution size. $E <-$ extracted edge matrix. $k \leftarrow$ downsampling factor. $c \leftarrow$ amount of thread workers. Typically $c = \check{N}$.	$\frac{c(H \times W)}{k}$ (6.1)
Image Selection module	$s \leftarrow$ sharpness list of size $\check{N}$.	$s + \check{N}(H \times W)$
Denoing Step	Operation performed in-place using same matrices used by previous module. N denotes the length of the filtered image set.	$N(H \times W)$

MODULE	DATA STRUCTURES	ALLOCATION
Unsharp Masking Step	Operation performed in-place. N denotes the length of the filtered image set.	$N(H \times W)$
Image Alignment Module	$\{k_0...k_N\} \leftarrow$ detected keypoints. $\{m_1...m_N\} \leftarrow$ matched keypoints. $H \leftarrow 3 \times 3$ homography matrix.	$k_0 + L_i + k_i + m_i + H$ where $i = 1...N$
Alignment Selection Module	$\{A_0...A_N\}$ and $\{B_0...B_N\} \leftarrow$ aligned image set A and B.	$4N(H \times W) + L_0$
Image Fusion Module	$A \leftarrow$ accumulator matrix. $D \leftarrow$ divisor matrix. $S \leftarrow$ scaling factor. $\{W_0...W_N\} \leftarrow$ chosen aligned image sets.	$A+D+W_i$, where A, D and W_i are consuming $S(H \times W)$ each.

Figure 14: Space consumption table

consumption does not vary across different image sets, based from how the SR system was designed for stability. *H x W* denote the image resolution size.

In the *Input Module*, only the file path is needed and thus, will only need a list. The *Edge Detection Module* reads $\check{N}$ down-sampled images where $\check{N}$ is the total number of images read using the file path list from the *Input Module*. It has been observed that the accuracy of detecting edges is not affected even if images are down-sampled. Thus, down-sampling the images can significantly improve processing time. The down-sampling factor is $k = 8$. The *Edge Detection Module* is also threaded such that the number of workers extracting the sharpness measure value is equal to $\check{N}$. In the *Image Selection Module*, the sharpness values are stored in s of size $\check{N}$.

After selecting reliable images, memory is freed and should guarantee that the SR system currently holds at most N images where $N < \check{N}$. Denoising and Unsharp Masking are performed in-place and only consumes at most *N(H x W).*

In the *Image Alignment Module*, the detected keypoints, matched keypoints and a homography matrix needs to be stored. Alignment is performed sequentially and memory is freed after every image alignment performed. For the *Alignment Selection Module*, it needs to store two aligned images (Perspective and MTB alignment) and its corresponding edges produced, thus consuming at most *4N (H x W)* and the first reference LR image, L_0.

The *Image Fusion Module* is the most memory-intensive task. Image fusion is performed sequentially and will need 3 matrices in memory, each consuming *S(H x W)* where *S* is the scaling factor, and *S = 2*. Each image is read from file, loaded to a matrix, *L*, and up-sampled using Bicubic interpolation before being added to the accumulator matrix, *A*. Once performed, *L* is immediately freed for the next image.

6. Conclusion

A system architecture for a multiple-image SR prototype for mobile devices is proposed in this paper. This study discusses the several modules that composes the SR system, namely the *Input Module, Edge Detection Module, Image Selection Module, Image Alignment Module, Alignment Selection Module,* and *Image Fusion Module*. Computational resources are managed using design patterns such as matrix pooling and thread barriers that ensures that the SR system is stable and can produce HR images with a x2 scaling factor.

Denoising feature has the longest average processing time of 3 minutes and 28 seconds, as compared to other modules. *Image Fusion Module* is the most memory-intensive because individual images are up-sampled sequentially using Bicubic interpolation with x2 scaling factor.

The analysis of performance of an SR system on a mobile device proves that a computationally expensive system can be optimized for mobile devices. The SR system can be further improved by redesigning some of the modules for concurrency. The *Edge Detection Module,* for example, has the lowest performance time compared to other modules because of its multi-threaded design.

7. References

Buades, A., Coll, B., & Morel, J.-M. (2005). Non-Local Means Denoising. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 60-65). IEEE.

Capel, D., & Zisserman, A. (2003). Computer vision applied to super resolution. *IEEE Signal Processing Magazine*, 75-86.

Chen, X., He, X., Yang, J., & Wu, Q. (2011). An effective document image deblurring algorithm. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 369-376). Colorado Springs: IEEE.

Chu. (2013). Super-resolution image reconstruction for mobile devices. *Multimedia Systems*, 315-337.

Del Gallego, N. P., & Ilao, J. (2017). Multiple-image super-resolution on mobile devices: an image warping approach. *EURASIP Journal on Image and Video Processing*, 15.

Farsiu, S., Milanfar, Robinson, M. D., & Elad, M. (2004). Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 1327-1344.

Fergus, Singh, Hertzmann, Roweis, & Freeman. (2006). Removing camera shake from a single photograph. *SIGGRAPH '06* (pp. 787-794). New York: ACM.

Mitzel, D., Pock, T., Schoenemann, T., & Cremers, D. (2009). Video Super Resolution Using Duality Based TV-L1 Optical Flow. *Proceedings of the 31st DAGM Symposium on Pattern Recognition*, 432-441.

Simpkins, & Stevenson. (2012). Mathematical Optics: Classical, Quantum, and Computational Methods. In *An introduction to super-resolution imaging* (p. Chapter 16). CRC Press.

Singh, M., Lu, C., Basu, A., & Mrinal, M. (2012). Choice of Low Resolution Sample Sets for Efficient Super-resolution Signal Reconstruction. *Journal of Visual Communication and Image Representation*, 194-207.

Ward, G. (2003). Fast, Robust Image Registration for Compositing High Dynamic Range Photographs from Handheld Exposures. *Journal of Graphics Tools*, 17-30.

Yang, J., Wright, J., Huang, T., & Ma, Y. (2010). Image Super-Resolution Via Sparse Representation. *IEEE Transactions on Image Processing*, 2861-2873.

Yuan, Sun, Quan, & Shum. (2007). Image deblurring with blurred/noisy image pairs. *SIGGRAPH '07*. New York: ACM.