Non-Local Means Denoising

If we are going to describe and discuss an algorithm which removes noise from (denoises) images, then we must first understand how images are formed. A digital image is a graphical representation of a scene, for a colour image this representation is obtained by measuring the value of the three primary light colours (red, green, and blue) at each pixel (picture element) on the camera's sensor. A 2D grid of values is then formed, indexed as (x,y). However, no digital image is a perfect representation of the original 2D signal, since they are limited in resolution by sampling and they contain noise. This is why noise removal is such a major component of image processing, because it is important that we are able to limit the effects it has on image visualisation and analysis. In an ideal world, there would be no noise in images. Every pixel in an image is subject to some perturbations, which can be due to the random nature of the photon counting process in each sensor. Therefore, an image is often not a perfect representation of the real world scene it captures. Noise can also be amplified by digital corrections of the camera or by image processing software such as tools removing blur or increasing contrast in the image [2]. This is how noise occurs.
We can summarise this idea as $v(i) = u(i) + n(i)$, where *v(i)* is the observed value, *u(i)* is the "true" value, and *n(i)* is the noise perturbation at a pixel i [1].
The idea behind the non-local means denoising algorithm is that we replace the colour of a target pixel with a weighted average (mean) of the colours of similar pixels, and we iterate through all of the pixels in the image doing this, vastly reducing or even (ideally) eliminating noise in the image [2]. To do this, we must first scan a large neighbourhood, which may even be the entire image, since the most similar pixels may not be close to the target pixel at all. We want to take most notice of the pixels which most closely resemble the pixel being processed, to evaluate this we compare their neighbourhoods for similarity. A pixel's neighbourhood in this context is the group of pixels which immediately surround the pixel in question, this is typically an N x N grid of pixels where N is an odd number and greater than or equal to 3. Here we are dealing with two different sizes of neighbourhoods, the larger neighbourhood which we scan for similar pixels, and the local neighbourhood of each pixel which we use to evaluate similarity.
We replace the value of the target pixel with a weighted mean of the value of the pixels in the larger neighbourhood. This weighting is according to how similar they and their local neighbourhoods are to the target pixel, so the more similar the pixel and its neighbourhood are, the greater weight it carries.
To generalise, Output pixel = weighted sum of other pixels, weighted according to similarity measure.

$$I_{output}(i,j) = \frac{\sum W_{neighbourhood}(k,l)\, I_{input}(k,l)}{\sum W_{neighbourhood}(k,l)}$$

A commonly used similarity measure is obtained by applying an exponential function on the squared Euclidean distance between local neighbourhoods. One such method, proposed in [2], to compute the weights $w(p,q)$ is:

$$w(p,q) = e^{-\frac{max(d^2 - 2\sigma^2, 0.0)}{h^2}},$$

where *p* is the target pixel, *q* is the pixel we are comparing it to, σ is the standard deviation of the noise, and *h* is the filtering parameter which is set depending on the value of σ. d^2 is the squared Euclidean distance between the colour patches centered at *p* and *q* respectively, in other words, it is a measurement of how similar the pixels *p* and *q* are because it measures the similarity of their local neighbourhoods.

$$d^2 = d^2(B(p,f), B(q,f)) = \frac{1}{3(2f+1)^2} \sum_{i=1}^{3} \sum_{j \in B(0,f)} (u_i(p+j) - u_i(q+j))^2,$$

where u_1, u_2, u_3, are the 3 colour channels of an RGB colour image, and the local neighbourhoods centered around each of the pixels *p* and *q* are of size $(2f+1) \times (2f+1)$. This is how we compute the similarity between two pixels.
Now that we understand how the squared Euclidean distance is calculated, we can better understand the previously mentioned equation, computing $w(p,q)$.
The way this all works in conjunction is that patches with squared Euclidean distances, d^2, smaller than $2\sigma^2$ are set to 1, meaning they are the most significant and carry the most weight. This is because the max of $d^2 - 2\sigma^2$ and 0.0 will be 0.0, so $w(p,q) = e^{-0.0} = 1$ for small Euclidean distances (similar pixels). Meanwhile, as d^2 increases (pixels become less similar) the exponent will increase and $w(p,q)$ will rapidly decrease, since the exponent is negated. This means that dissimilar pixels are weighted much less, so their values are less significant in the calculation.

There are two different implementations of the non-local means denoising algorithm, they are pixelwise implementation and patchwise implementation.

Firstly we will look at pixelwise implementation, which is the way in which the algorithm was originally presented. Here, as described above and in [2], the value of each pixel is restored as an average of the most resembling pixels. Since we are dealing with colour images, for each pixel, each channel value u_1, u_2, u_3 (R,G, B) can have its average computed independently. We use an exponential kernel, as previously described, to compute the weightings. We can bring everything together with the aforementioned equations as follows:

$u_i(p) = \frac{1}{C(p)} \sum_{q \in B(p,r)} u_i(q)\, w(p,q)$, where $C(p) = \sum_{q \in B(p,r)} w(p,q)$ [2],

where p is the target pixel, $i = 1, 2, 3$, and $B(p, r)$ is a large neighbourhood centered at p and of size $(2r + 1) \times (2r + 1)$. This is how pixelwise non-local means denoising is implemented.

Next we will look at patchwise implementation, which is an elegant improvement over the original pixelwise implementation. Here we denoise a colour image one patch at a time rather than one pixel at a time, iterating over patches instead of pixels in the image, it is implemented as follows:

$B_i = \frac{1}{C} \sum_{Q=Q(q,f) \in B(p,r)} u_i(Q)\, w(B,Q)$, where $C = \sum_{Q=Q(q,f) \in B(p,r)} w(B,Q)$ [2],

where $i = 1, 2, 3$ as before, and $B(p,r)$ is a neighbourhood with centre p and size $(2r + 1) \times (2r + 1)$.
The key realisation here is that C is constant for every pixel within a particular patch, so we can use the same value of C to denoise an entire patch. This is therefore much quicker than before when we were iterating sequentially through each pixel in the image and calculating a new value of C. The increase in efficiency is due to disposing of $N^2 = (2f + 1)^2$ possible estimates for each pixel [2], since all pixels in the patch have the same importance, and therefore the same weight can be used to denoise all pixels in the patch and not just the target pixel.
These estimates can be finally averaged at each target pixel to construct the denoised image as follows:

$u_i(p) = \frac{1}{N^2} \sum_{Q=Q(q,f) \mid q \in B(p,f)} Q_i(p)$ [2].

The patchwise implementation is more efficient than the pixelwise, however, the overall quality in terms of preservation of details is not improved [2].

The size of the comparison patch (local neighbourhood) and research block (larger neighbourhood) depend on the value of σ [2]. σ represents the standard deviation of the noise in the image, so if the value of σ is small then there is relatively little noise, and vice versa. In order to maximise the effectiveness of the non-local means denoising algorithm it is necessary to change its parameters depending on how noisy the image is. As σ increases, in order to get the best output possible, the size of the comparison patch (comp. patch) must also increase, meaning that the local neighbourhood of each pixel that we are testing for similarity increases in size, in order to make the comparison function robust enough to the higher level of noise. We must also increase the size of the research block (res. block), this is so that we are able to find more similar pixels within the larger neighbourhood, in order to increase the noise removal capacity of the algorithm. We can also change the value of the filtering parameter $h = k\sigma$, by decreasing the value of k as the value of σ increases, since we are increasing the size of the comparison patch, the Euclidean distance between two pure noise patches is concentrated more around $2\sigma^2$, so a smaller value of k can be used [2]. The table shown on the right, from [2], shows the set of parameters used by the non-local means denoising algorithm (patchwise implementation) for colour images.

σ	Comp. Patch	Res. Block	h
$0 < \sigma \leq 25$	3×3	21×21	0.55σ
$25 < \sigma \leq 55$	5×5	35×35	0.40σ
$55 < \sigma < 100$	7×7	35×35	0.35σ

Here is a demonstration of the influence of these algorithmic parameters on the output image, specifically what happens when we try to remove a lot of noise from the image when it does not actually contain a huge amount of noise.


Figure 1.


Figure 2.


Figure 3.

Fig. 1 shows a noisy image, where $0 \leq \sigma \leq 25$, so we should use the parameters shown in the first row of the table above, however if we were to implement the algorithm as if $\sigma = 100$, such that the comparison patch is 7 x 7, the research block is 35 x 35, and k = 0.35, then the output is shown in Fig. 2, which shows an image which has been denoised, but has lost a significant amount of detail, so it has been overly smoothed and overly denoised. If we were to use the parameters suggested by [2], supposing that $\sigma = 25$, such that the comparison patch is 3 x 3, the research block is 21 x 21, and

$k = 0.55$, then the output is shown in Fig. 3. This image has been denoised, but has lost significantly less fine detail than the image in Fig. 2, it has not been overly smoothed since we have used more appropriate parameters for the relatively small amount of noise which was initially present did not require larger blocks or patches or a smaller k value. It has been sufficiently denoised.
Here is another example, this time demonstrating what happens if we don't allow the algorithm to remove enough noise when there is quite a lot of noise in the image.

Figure 4.

Figure 5.

Figure 6.

Fig. 4 shows a highly noisy image, where $25 \leq \sigma \leq 55$, so we should be using the parameters shown in the second row of the table above, but if we were to use implement the algorithm as if $\sigma = 25$, when in reality it is greater, then the output is the image shown in Fig.5, where there is still noise present in the supposedly denoised image. Whereas if we were to use the correct variables then we would get the output shown in Fig. 6, where the noise has been sufficiently removed.
We can now compare Fig. 3 to Fig. 5. These two images were output by the program when using the same parameters, however, it is noticeable that Fig. 3 has significantly less noise present than Fig. 5 does, and this is due to the fact its input image (Fig. 4) originally contained more noise, and since we have used the same algorithmic parameters, we have only been able to remove the same amount of noise, leaving more left over in Fig. 5.
We can also compare Fig. 3 to Fig. 6. These two images have both been sufficiently denoised, but since the original input image for Fig. 6 (Fig. 4) contained more noise, the output image has less fine detail present than Fig. 3 does because it was necessary to use different parameters in order to allow the algorithm to eliminate more noise, which also causes the loss of more fine detail.
The tradeoff that we have with denoising algorithms in general is always between removal of noise and removal of fine detail, we would ideally be able to eliminate all of the noise from an image, whilst removing none of the fine detail, so in practice we aim to maximise the former and minimise the latter.

The non-local means denoising algorithm is close to state-of-the-art noise removal for photographic imagery. Non-local means denoising is a form of spatial filtering algorithm, but there are other spatial filtering algorithms available, in fact there are two different categories of them, linear and non-linear spatial filtering. Linear spatial filtering means that each output pixel is a linear combination of the corresponding input pixel's neighbourhood, whereas non-linear spatial filtering means that each output pixel is not a linear function of the corresponding pixel's neighbourhood, there is some decision based algorithm used instead.
A mean filter is an example of a linear spatial filter and is a simpler alternative to non-local means. Here we simply replace a given pixel with the (unweighted) mean of its local neighbourhood. This is good at reducing the magnitude of sudden intensity jumps such as salt and pepper noise, just as non-local means does. However it does not reduce the magnitude as much as non-local means so it is not as effective. A downside to the mean filter is that it causes edge blurring, since it removes the high frequency image detail, this is still a problem with non-local means, but not as big an issue since edges are blurred less. Overall, mean filtering performs fairly well against Gaussian noise but still causes the loss of fine detail, and it performs poorly against salt and pepper noise, the upside is that the algorithm is faster than the non-local means algorithm.
Minimum, maximum, and median filtering are all examples of non-linear spatial filtering, since the output pixel is not a linear function of the corresponding input pixel's neighbourhood and is instead, respectively, the minimum, maximum, and median value of the input neighbourhood.
A more sophisticated alternative, but still not as sophisticated as non-local means, is Gaussian filtering. In a Gaussian filter, the new intensity of a pixel is the weighted sum of the intensities of the pixels in its neighbourhood. The Gaussian filter is good at noise removal, meaning it is optimal in flat parts of the image, but it also removes detail from the image, blurring edges and textures [1], this is the same issue we have with mean filtering. Non-local means is better than Gaussian filtering at retaining the fine detail of images.
One of the most notable strengths of the non-local means denoising algorithm is that it performs well against both Gaussian noise and salt and pepper noise, better than any other algorithm, which is why it is close to state-of-the-art. Another advantage is that, as demonstrated above, the parameters of the algorithm can be changed according to the level of noise present in the input image, in order to produce a better output image.

A disadvantage of non-local means denoising, however, is that it can be slow if it is not optimised, the running time increases exponentially as the parameters increase linearly, since it is $O(N^2)$. If we were to use a research block (search window) of 21 x 21 pixels, a comparison patch (similarity square) of 7 x 7 pixels, and if N^2 is the number of pixels in the image, then the final complexity of the algorithm is roughly 49 x 441 x N^2 [1].
Yet another advantage of the non-local means denoising is that it can be easily parallelised, one way in which this could be done is by simply splitting the computation between 3 processors, where each one would calculate the value of each pixel in a particular colour channel, since the value of the 3 colour channels are computed independently of each other.
The non-local means denoising algorithm is proven to be asymptotically optimal under a general statistical image model [3].

Since the non-local means denoising algorithm was initially proposed in [1] in 2005, there have been modifications to it and extensions of it proposed in literature. A modification to improve the efficiency of the algorithm was proposed in [2] which was that it was possible to implement the algorithm patchwise rather than pixelwise, as aforementioned. This hugely decreases the running time.

There are many applications of the original non-local means denoising algorithm and its extensions. The biggest application of this algorithm, and of denoising algorithms in general, is to allow for better image recognition and further image analysis such as object recognition or facial recognition. An image containing noise will make it much more difficult for any kind of imaging software to perform its task on the image, since the variability would be so much greater for what could be images of the same scene or object. Images which are the output of the non-local means denoising algorithm are much easier to process, since there is much less variability, making the job of any kind of assistive imaging or computer vision software much easier and more efficient, which is clearly a huge positive.
Non-local means denoising is the algorithm which will accentuate all of these benefits the most, since it is able to remove the most amount of noise from images whilst also retaining the most fine detail in them.

References:

[1] A. Buades, B. Coll, J.M. Morel. “A non-local algorithm for image denoising.” 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05). Vol. 2. IEEE, 2005. http://dx.doi.org/10.1109/CVPR.2005.38

[2] A. Buades, B. Coll, J.M. Morel. “Non-local means denoising.” Image Processing On Line 1 (2011): 208-212. http://dx.doi.org/10.5201/ipol.2011.bcm_nlm

[3] A. Buades, B. Coll, J.M. Morel, “A review of image denoising methods, with a new one”, Multiscale Modeling and Simulation, Vol. 4 (2), pp: 490-530, 2006. http://dx.doi.org/10.1137/040616024

[4] A. Buades, B. Coll, J.M. Morel, “Image data processing method by reducing image noise, and camera integrating means for implementing said method”, EP Patent 1,749,278 (Feb. 7), 2007.

[5] A. Buades, B. Coll, J.M. Morel. On image denoising methods. Technical Report 2004-15, CMLA, 2004.

[6] A. Buades, B.Coll, J.M. Morel. Neighborhood filters and pde’s. Technical Report 2005-04, CMLA, 2005.

Appendix:

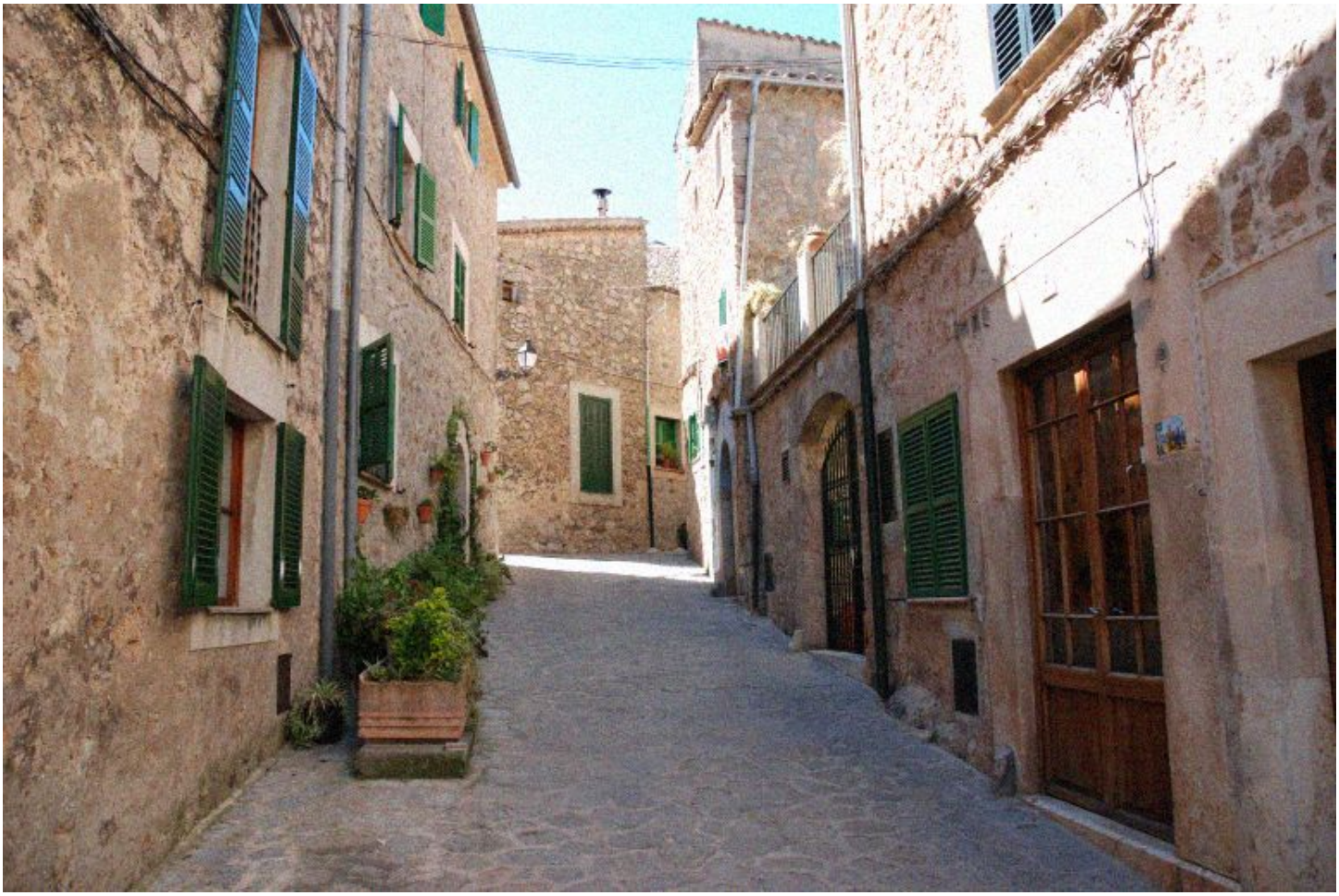

Figure 1.

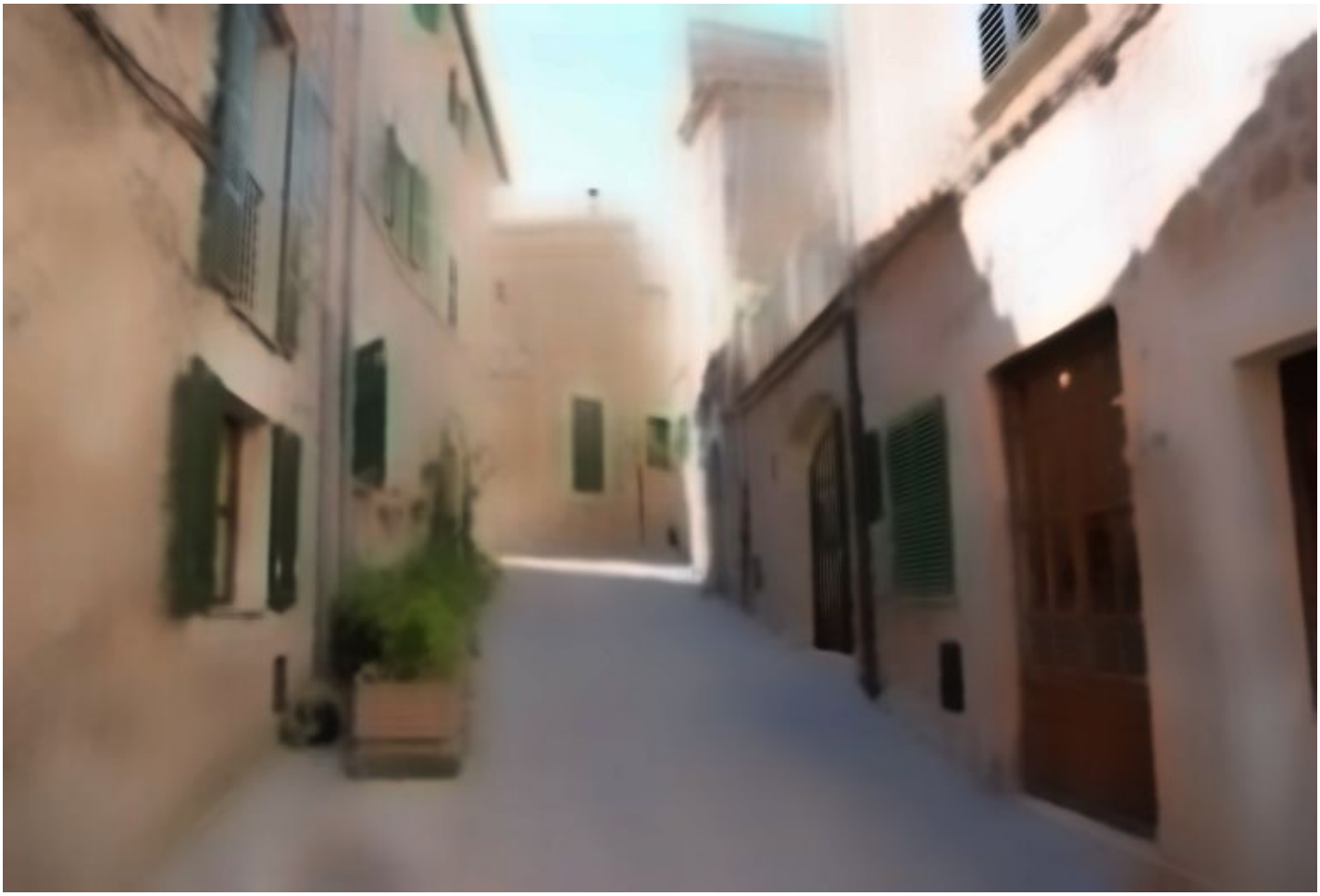

Figure 2.

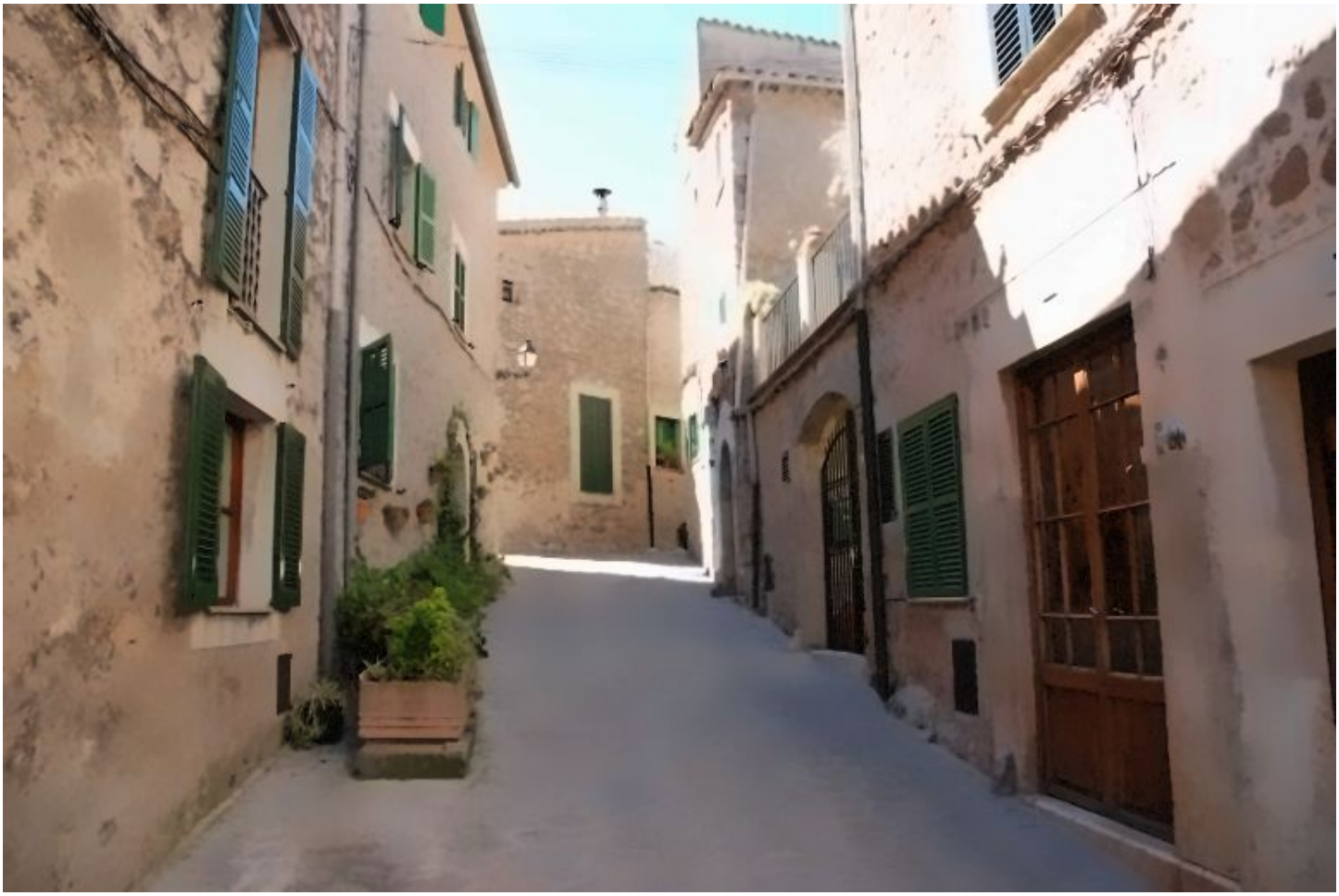

Figure 3.

Figure 4.

Figure 5.

Figure 6.