Image Inpainting

PangFeng Zheng
pangfengz@student.unimelb.edu.au

PangFeng Zheng

Task 1:

The RMS error for different location in two images

	[100,150, 50, 100]	[125,175,75,125]	[150,200,100,150]	[175,225,125,175]	[200,250,150,200]	Average RMS
brick.png	40.1196	32.5304	36.6277	32.4691	38.3578	38.3578
basket.png	47.6941	42.3240	43.4142	51.9068	42.0015	45.4693

Limitation:

The masked region cannot be located at the left or upper edge of the image, which will try to access outside of the image. The 'make_context_locations' function creates a set of offsets for the context window. These offsets are all negative or zero. This means that the context window is always located at the top left of the given position.

Analysis:

While RMS error is around 32 ([125,175,75,125]),  the quality of the reconstruction is good and just a little flaw. But if RMS error is over 35, for example in location [150,200,100,150], the RMS of brick.png is 36.6  , we can clearly see the reconstruction of edge is not that well.

For mask location, if it is located in an important or complex area of the image (e.g., the edge of an object/including complex texture), then it may be more difficult to reconstruct this area, resulting in a higher RMS error. Conversely, if the location is in a uniform or simple region of the image, it may be easier to repair this region, resulting in a lower RMS error.

word count:180

Task 2:

The RMS error for different size of context window in two images

	[3,3]	[5,5]	[7,7]	[9,7]	[11,7]	Average
brick.png	40.7631	35.7195	31.2152	29.0946	33.3905	34.0366
basket.png	48.7188	46.5941	51.6438	49.9927	44.4178	48.2734

The run time for different size of context window in two images

	[3,3]	[5,5]	[7,7]	[9,7]	[11,7]	Average
brick.png	25.5569	59.5559	111.6519	143.7446	167.1491	101.5317
basket.png	27.3526	64.7787	125.6634	175.3104	226.5919	123.9394

Qualitative description:

brick.png:  [3,3]  [5,5]  [7,7]  [9,7]  [11,7]

basket.png  [3,3]  [5,5]  [7,7]  [9,7]  [11.7]

Through the reconstructed images' view, while the size of context window increasing, the quality of reconstruction is becoming better as more surrounding information are captured.

The shape of context window is influencing the quality of reconstruction in different way, but also depends on the pixel distribution (feature shape) of original image. For example, a rectangle context window presents a better performance in the "brick.png", comparing with the [7,7] size, the result of size [9,7] obviously provide a better reconstruction quality. But in the "basket.png", a square context window provides a better performance, as the differences of visibility in [7,7] and [9,7], square size [7,7] have a better result.

Quantitative performance:

From the analysis of RMS error, the reconstruction of brick.png is best when the context size is [9,7], while the repair effect of basket.png is best when the context size is [11,7]. For "brick.png", the RMS error decreases with increasing context window size until [9,7] and then increases slightly at [11,7]. But the RMS error of "basket.png" fluctuates greatly under different context window sizes which may mean that the content of "basket.png" is more complex than "brick.png". Overall, a larger context window can capture more information but not always benefit the reconstruction. In some cases, the additional information may not provide any new useful information about the masked area and may instead lead to errors.

For the result of run time, we can see while increasing the size of context window, the time will also increase. We also need to weigh the reconstruction quality against computation time if the dataset is huge.

word count:262

Task 3:

Different context shapes

The RMS error/run time for rectangle_v1 context shape in brick.png

Location - size of context	rectangle (Original)	rectangle_v1	rectangle (step = 2)	rectangle (SAD)
[125,175,75,125] - [7,7]	RMS: 33.2051 Time: 113.0223	RMS: 43.6393 Time: 99.2619	RMS: 42.6785 Time: 39.0676	RMS: 35.8896 Time: 115.9110
[175,225,125,175] - [7,7]	RMS: 33.9610 Time: 111.7660	RMS: 40.4408 Time: 104.4271	RMS: 38.7136 Time: 42.5063	RMS: 42.0261 Time: 110.9669
[200,250,150,200] - [7,7]	RMS: 31.9526 Time: 112.9912	RMS: 31.8978 Time: 117.6225	RMS: 38.3781 Time: 38.7347	RMS: 36.8776 Time: 118.6813
[125,175,75,125] - [9,7]	RMS: 34.8343 Time: 159.4090	RMS: 35.4628 Time: 146.3810	RMS: 37.4197 Time: 46.5658	RMS: 30.9774 Time: 139.5352
[175,225,125,175] - [9,7]	RMS: 37.1959 Time: 147.7341	RMS: 38.9793 Time: 140.6463	RMS: 39.0714 Time: 46.3089	RMS: 35.3816 Time: 153.8598
[200,250,150,200] - [9,7]	RMS: 31.2901 Time: 149.5234	RMS: 32.4666 Time: 140.2039	RMS: 38.9669 Time: 47.0546	RMS: 35.7582 Time: 144.9007
[125,175,75,125] - [7,11]	RMS: 37.7599 Time: 177.7149	RMS: 38.0013 Time: 184.6660	RMS: 41.8017 Time: 54.6746	RMS: 38.2987 Time: 178.6898
[175,225,125,175] - [7,11]	RMS: 34.9595 Time: 170.8833	RMS: 36.5916 Time: 184.2892	RMS: 40.7056 Time: 54.6735	RMS: 34.9857 Time: 173.1168
[200,250,150,200] - [7,11]	RMS: 35.6255 Time: 183.4987	RMS: 34.4804 Time: 173.3777	RMS: 34.4197 Time: 54.3881	RMS: 33.6151 Time: 172.2237
Average:	RMS: 34.5315 Time: 147.3937	RMS: 36.8844 Time: 143.4306	RMS: 39.1284 Time: 47.1082	RMS: 36.0456 Time: 145.3206

Assumption: Only use brick.png for testing the performance; The mask locations and sizes of context are selected from Task 1 & 2 which provide a good performance.

Different methods and purpose:

- rectangle_v1: double the width and half the height and generates the offset more in horizontal. Better capture of horizontal features and textures in images.
- rectangle (step = 2): skipping one in context pixel. Reducing computational complexity and potentially speeds up the reconstructive process.
- rectangle (SAD): change the original SSD algorithm to SAD. Reducing computational complexity by not using multiplication in calculation.

Result:

The best result is provided by the SAD algorithm while Location - size of context is [125,175,75,125] - [9,7]. As we can see, the reconstructed image can only see the problem of edge. 

Based on the data in table, for *RMS error*, the original rectangle method has the lowest average error of 34.5315, which means it provides the best inpainting overall. The average error of the rectangle (step = 2) method is 39.1284, it is the highest among all methods, which may be because it skips some context pixels, causing the inpainting effect to be inferior to other methods. The rectangle_v1 and SAD method have very close error.

For *execution time*, the rectangle (step = 2) method has the lowest average execution time of 47.1082 seconds, as expected since it reduces computation by skipping some context pixels, the other two methods' cost of time are similar with the original method.

In conclusion, for the applications that require a quick reconstruction, to skip some pixels (rectangle step = 2) may be a good choice. But for applications requiring high-quality repair, the original rectangle method or the rectangle (SAD) method may be more suitable. Future work can combine some of the methods together to see the performance, or also to test the influences of different images with different texture (e.g., horizontal/vertical).

word count:315