Retrieving textures for 3D scanning system based on grid pattern method

Thibault Yohan
University of Kagoshima, Department of Engneering
Kagoshima, Japan
y.thibault@ibe.kagoshima-u.ac.jp

Kawasaki Hiroshi
University of Kagoshima, Department of Engneering
Kagoshima, Japan
kawasaki@ibe.kagoshima-u.ac.jp

Sagawa Ryusuke
N. Institute of Advanced Industrial Science and Technology
Ibaraki,Japan
ryusuke.sagawa@aist.go.jp

Furukawa Ryo
Faculty of Information Sciences, Hiroshima City University
Hiroshima ,Japan ryo-f@hiroshima-cu.ac.jp

***Abstract*—This paper proposes a method of retrieving texture from images containing pattern used for one shot 3D scanning system based on a grid pattern. Range scanners using projector-camera systems can be used for real-time 3D scanning with accurate results and at a low cost. However, these techniques cannot retrieve color information from the captured object since to obtain the best results, they require dark environments and the projection on the object to reconstruct dense and bright lines pattern masks most of the object textures. Therefore, we propose a technique of slightly modifying the input of a given 3D scanning system in order to have bright images with texture information and then retrieve the textures of the object using an inpainting method. The presented method adapts simple algorithms such as Canny filter and patches based inpainting in order to obtain a fully automatic method that does not require any user intervention.**

***Keywords*-Inpainting; structured light;**

I. Introduction

Recently, several papers on "one shot 3D scanning system based on grid pattern" have been published [3], [4]. These methods perform dense 3D reconstruction of objects by the projection of a single pattern on objects and studying the deformation of the pattern by the object's shape. One of the main advantages of these methods is the possibility of 3D reconstruction using only one projector and one camera. Thus it can be used for interactive system in embedded systems for a reasonable cost. However, these methods are designed to work in dark environments, and the typical projected pattern have a black background and bright lines with several colors as illustrated on Figure 1 (left). Therefore, color information on these images mostly come from the pattern instead of the object's textures. Thus, the results given by these systems can give 3D points but can hardly recover textures from the object.

In this paper, we propose a method for recovering textures of objects in images that contain projected pattern using inpainting method. Our method modifies the pattern of the method proposed in [3] in order to obtain pictures with visible textures on the object. The two main difficulties of performing inpainting in an embedded system are to retrieve the textures that are mostly hidden by the projected pattern and to have a fully automatic method. To our best knowledge, there is no previous work that recovers textures from pictures obtained by this kind of system. None of the previous work on inpainting is suitable for this particular case, because existing methods require user input to select the areas to be inpainted and are not suitable when these areas are dense and spread. Moreover, textures are still slightly visible under the projected pattern and this case is not considered by classical techniques.

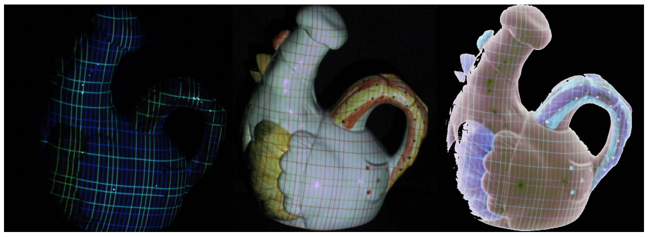
Figure 1. (left) The regular pattern for line detection, (middle) The inverted pattern that contains texture information, (right) The inverted inverted pattern.

The proposed method automatically detects the curves[1] of the pattern to be inpainted. Thus, it does not require any user intervention. To recover the hidden textures with the lack of visible texture, we propose a patch based inpainting method using a pair of patches found in the zone surrounding the curve to be inpainted and mix them with the available color information under the curve. The output is a texture image ready to be applied on the 3D data given by the method in [3]

The paper is structured as follows: Section II presents some inpainting methods that can be used for our problem.

[1]The projected pattern contains lines that become curves following the shapes of the object.

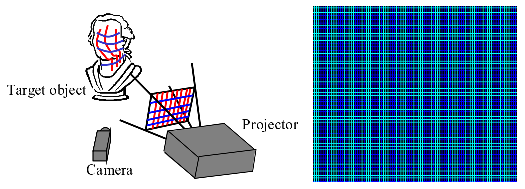

Figure 2. (Left) Scanning system: multiple lines are projected and their intersections are detected and used for reconstruction. (Right) projected pattern.

Section III presents an overview of our method. Section IV explains the required modifications on method proposed in [3] to obtain our input images. Then Section V shows how our method automatically detects the surfaces to be inpainted in the input image and SectionVI explains how to perform the inpainting. Finally, we present some experiments in Section VII and conclude the paper in Section VIII.

II. Previous works

The inpainting problem has been widely studied for about 10 years. Two main approaches can be observed in most of the published papers. One is looking for patches in the picture to inpaint [5], [1] or in a given set of pictures and one that smoothly propagates information from the areas surrounding the part to be inpainted [6].

The methods that propagate information are usually fast and give acceptable results but are not as good as the patch based methods. These methods search into the input image or a given set of images for small patches that can be applied to the area to inpaint. The selection of the patch is usually done using a distance function to find the best match in the picture. These methods are usually slow since the research of the best patch is time consuming. A recent paper [1] proposes a random method of researching to find almost the best patch quickly. This method cannot be applied in our case since the parts to inpaint are regularly disposed in our input picture.

All these methods require user inputs to select the part of the image to inpaint. In our case the parts to inpaint are determined by the projection of the pattern on the object. Thus they should be automatically detected and require no user input.

III. Overview of the method

The proposed 3D measurement system consists of a calibrated camera and a calibrated projector as shown in Fig. 2 (left). A grid pattern of vertical and horizontal lines is projected from the projector and captured by the camera.

We will use the 3D reconstruction technique proposed in [4]. In this technique, first, the projected grid pattern is extracted from the captured image using belief propagation and the De Bruijn sequence. Then, the intersection (grid) points of the detected curves are extracted. From the extracted grid points, simple linear constraints about the crossing pattern planes can be acquired. By solving the above linear

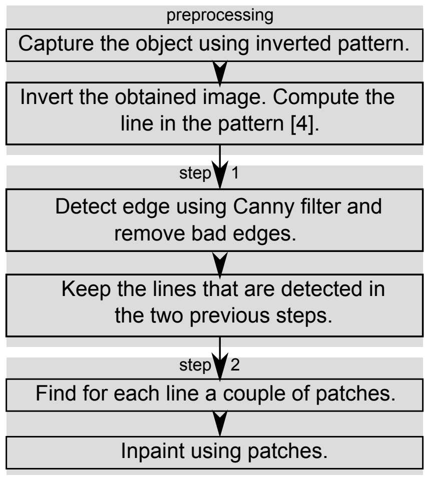
preprocessing
Capture the object using inverted pattern.
Invert the obtained image. Compute the line in the pattern [4].
step 1
Detect edge using Canny filter and remove bad edges.
Keep the lines that are detected in the two previous steps.
step 2
Find for each line a couple of patches.
Inpaint using patches.

Figure 3. The overview of our algorithm plus the preprocessing to obtain data.

equations generated from the grid points, solutions of the pattern planes are obtained. The remaining 1-DOF ambiguity in the solution is decided by matching the actual positions of the pattern planes, estimated planes and consequently 3D shapes of curves are reconstructed.

An overview of our method is shown in Figure 4 and in Diagram 3. Our system takes an image that contains an object with the projected pattern to inpaint as input. It also takes the set of curves in the image that is detected by the algorithm presented in [3] as input.

Our algorithm consists of two main steps plus two steps of preprocessing input data. The first preprocessing captures the data using the inverted pattern Figure 4(top left). The second inverts the obtained image and gives it as input for the detection curve algorithm Figure 4(bottom left). The first step detects the curves that belong to the pattern projected on the object and that should be inpainted, the second step performs the inpainting using the result of the previous step.

The first part of step 1 consists of a modified Canny filter that detects all the edges, even in the dark parts, of the first input image Figure 4(top middle). The result given by this filter is then dilated to identity the zones that could be curves and to determine their thickness.

The second part uses the curves of the pattern detected in the first input image by the algorithm presented in [3]. However, since its input is an image obtained by projecting the inverted pattern, some problems can occur such as false detection of curves such as edges of objects or shadows, and it does not give any information on the real thickness of the curves. Therefore the results given by this algorithm are compared with the one obtained during the first part to robustly obtain the locations of the curves and the areas that should be inpainted Figure 4(bottom middle). Note that areas to be inpainted correspond to the thickness of the curves. Hereafter, "thickness" represents the area to be inpainted.

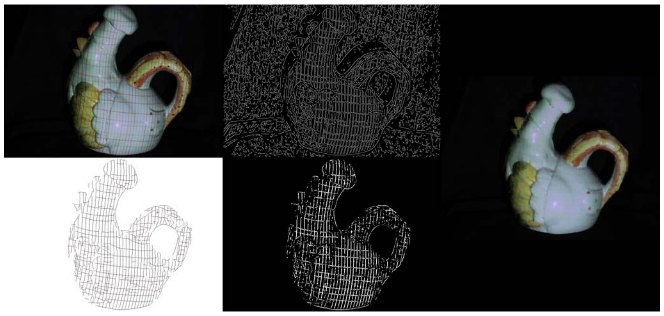
Figure 4. Overview of the method. (top/bottom left) input of the algorithm, (top middle) output of the curves selected after the Canny filter, (bottom middle) curves that will be inpainted, (right) final result.

The second step of our algorithm creates a mask that is the negative of the curves to inpaint and searches in the neighborhood of each curve for a couple of patches that can be applied to inpaint it. Some information from the curves to inpaint are merged with the patches to preserve a part of the textures under the projected pattern as illustrated on Figure 4(right).

IV. Preprocessing

The method presented in [3] uses a pattern with two colors: blue and cyan and a black background as illustrated in Figure 1 (left). This pattern is efficient to accurately detect the curves since it creates sharp borders of curves, does not create any shadows and masks most of the texture of the object. However, it does not contain textures and cannot be inpainted. To solve this problem, our method uses a pattern with a white background and two colors for lines: yellow and red which are the complementary colors of the two original ones. Then the obtained images contain textures of the object which are partially masked by the pattern as illustrated in Figure 1 (right).

Some simple preprocessing should be done before applying our algorithm. The first one is to modify the captured image in order to obtain blue and cyan curves that are easily detected by the algorithm in [3]. These modifications complement 255, the value of each color of each pixel in the image. The only restriction is to keep dark the dark parts of the picture. Indeed, most of the parts that have weak illumination in the image are far from the object and belong to the background. Thus, if these parts are complemented to 255, they will become bright and will disrupt the algorithm that detects curves since it assumes a black background. Then, all pixels which have their three color channel lower than a given threshold (15 in our experiments) is considered as dark and is not complemented. Figure 1(right) illustrates the result of the inversion of Figure 1 (center). Note that the shadows, shading, and dark textures will also become dark and thus not inverted and cause wrong line detection. However, those wrong lines are efficiently removed by our line detection algorithm and thus no particular treatment is required at this stage.

The complemented image is given to the algorithm that will return a binary image containing the detected curves as illustrated in Figure 4 (left bottom). Since the quality of the complemented image is lower than the quality of an image with the original pattern, the detection algorithm can return some false positive curves such as the shapes of the chicken pot illustrated in Figure 4 (left bottom). Moreover, this algorithm does not give any information on the thickness of the curves. Thus, it is not possible to only use its output to detect the curves to inpaint. The next section presents our method to robustly detect the curves of the projected pattern.

V. Detection of surfaces to inpaint

This section presents our method which automatically detects the surfaces in the picture to be inpainted. As explained in Section III, the detection of surfaces to inpaint is done in two steps. Thus, this section first explains how we modify the Canny filter in order to obtain the detection of the edge in all parts of the image. Then it explains how our method detects the edges that could be parts of curves to inpaint. Finally it shows how to use the results given by the algorithm in [3] to create the final mask.

A. *Detection of curves with an adaptive Canny filter*

The detection of curves starts by the application of the Canny filter [2]. Since the two colors for the lines of the projected pattern both contain 0 for their blue channel, it concentrates the gradient estimations on the two other channels. The original Canny filter uses the two same thresholds for the whole image. However, the illumination in our images is not uniform and there can be severe differences of light intensity between two parts of the images. Our algorithm divides the image into small squares and applies for each square a different pair of thresholds depending on their illumination.

Using the modified Canny filter, all the edges of pattern, textures, and others are detected. Thus edges that do not belong to curves of the projected pattern should be removed.

We assume that for each curve on the pattern, two edges are detected and that a pair of edges corresponding to a curve describes two parallel curves. We can also assume that only few pixels separate these two curves since the pattern is supposed to contain thin lines. Our method first detects the vertical curves and then the horizontal curves. Note that curves that are not parallel to x or y axes are normally detected.

The first step, illustrated in Figure 5, identifies pixels that are close enough to two distinct edges. For each pixel (x, y) that belongs to the edges map, it marks its neighbors on the same horizontal line: $(x-i, y), \ldots, (x+i, y)$, where i is the thickness of the curve to inpaint, as reached. If a pixel already reached is marked another time, it marks this pixel

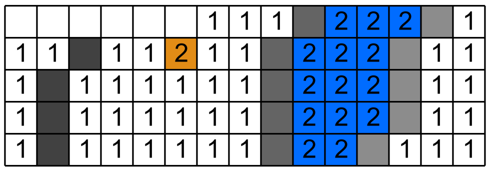

Figure 5. The full gray pixels represents the detected edges. Pixels marked with a one are close to at least one edge. Pixels marked with a two are close of two different edges and should be inpainted. Pixels marked with a two and with a yellow color are considered as isolated and removed from inpainted list.

as "to be inpainted". The pixels that belong to edges map are so far considered as "to be inpainted". Experiments show that a value of 3 for i works fine.

The second step removes isolated pixels, done in two scans. If a pixel marked to be inpainted has at least six on height neighbors to be inpainted, it marks this pixel. On the second scan if a pixel marked to be inpainted but not marked by the first scan, has no neighbor marked by the first scan, we remove it. This step removes all the pixels belonging to edges that do not represent curves. This step also removes the isolated pixel on the left side of Figure 5.

This operation is able to detect vertical curves. The same operation is done to detect horizontal curves but it marks pixels on the same vertical line instead of the same horizontal line.

The method described in this section selects all the curves and their thickness of the projected pattern. However, it may also detect some curves or textures that do not belong to the projected pattern. Then in order to increase the quality of results given by this method, we also use the output of the curves detection algorithm presented in [3].

B. Selection of curves to inpaint

The detection curves algorithm was made to be used on images taken in a dark environment using blue and cyan colors for the lines of the pattern. Thus the image does not contain any shadows and only a few textures. Under these conditions, the algorithm gives excellent results. However, since we consider pictures taken with a white background and red and yellow lines that have been inverted as explained in Section IV, the algorithm gives less good results and can detect edges of the object or shadows as lines of the pattern. Moreover, the algorithm gives curves' position but not their thickness.

The final step of our method is done crossing the set of curves obtained by the method described in Section V-A and the set of curves obtained using the detection curves algorithm in [3]. A curve will be inpainted if and only if it belongs to both sets. In other words, if a curve is not detected by the method presented in [3] or the one in Section V-A it will not be inpainted. The thickness of the curves will be decided using the set obtained in Section V-A.

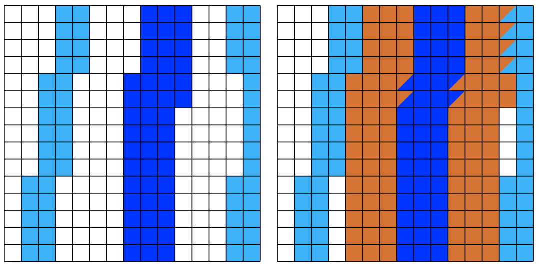

Figure 6. (left) The light and dark blue pixels are set to false and cannot be used to inpaint the blue pixels, (right) the orange pixels represent the two polygons selected to inpaint the dark blue ones. Note that pixels sharing two colors of the selected polygons cannot be used for inpainting.

VI. Inpaint the detected surfaces

In this section, we explain how we inpaint the curves that are detected by the method presented in Section V. It is done in two steps that are repeated for all the detected curves. Firstly a pair of patches, parallels to the curve, around the curve to inpaint is selected. Then, it computes the average between both patches and applies it on the part to inpaint.

A. Selection of the pair of patches

In the following, we consider a unique vertical curve to be inpainted. The first step creates a boolean mask $\mathcal{M}$ with the same size as the input image. $\mathcal{M}(x, y)$ is set to false if the pixel (x, y) is a pixel to inpaint. In Figure 6 (left), the blue pixels are set to false. The next step considers the smallest polygon $\mathcal{P}$ that contains all the pixels of the curve to be inpainted. In Figure 6 the dark blue pixels represents $\mathcal{P}$. Then, it slides $\mathcal{P}$ to the left (resp. right) in order to minimize the number of pixels in $\mathcal{P}$ that are set to false in $\mathcal{M}$. To avoid sliding too far and then to find a patch that has no connection with the part to inpaint, we also introduce the notion of distance between the patch and the curve to inpaint. The two selected patches are represented by the orange pixel in Figure 6 (right). Note that pixels having two colors belong to $\mathcal{P}_1$ or $\mathcal{P}_2$ and to $\mathcal{M}$. They belong to the best selected patch, but cannot be used for inpainting $\mathcal{P}$.

The same operation is repeated for all vertical curves and then for the horizontal curves by sliding $\mathcal{P}$ vertically instead of horizontally. Once the best pairs of patches for all curves to inpaint are found, we apply them as explain in the next section.

B. Application of the patches

As explained in Section VI-A, we consider a pair of patches to ensure that enough data are selected to properly inpaint the curve. Indeed, if the lines in the projected pattern are too close, the data available to inpaint will be missing. Then it becomes necessary to use data from both sides.

In usual inpainting methods, we assume that surfaces to be inpainted do not contain any texture information, and thus should be fully covered by patches. During experiments, we noticed that the curves to be inpainted contain some texture information. Therefore, the inpainting should consider them.

Assuming that the two polygons $\mathcal{P}_1$ and $\mathcal{P}_2$ that contain the pair of patches to inpaint the curve enclosed in the polygon $\mathcal{P}$ are known, we can create the final patch to be applied on $\mathcal{P}$. The value of each pixel in the final patch is obtained by taking an average between the corresponding pixels in $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}$. In order to keep a part of the original pixel and to keep some information on the texture under the curve, the value of the pixel to inpaint in the final patch will be composed by 80% of the corresponding pixels in $\mathcal{P}_1$ and $\mathcal{P}_2$ and by 20% of the original value of the pixel in $\mathcal{P}$.

VII. Experiments

In this section, we present some experiments. Figure 8 shows the result of the full process of the 3D reconstruction and inpainting from three different points of view. The reconstruction is done using the algorithm presented in [3]. The left side of each pair of images is the reconstruction of a plastic manikin wearing a t-shirt using the regular pattern. The reconstruction of the t-shirt globally gives good results as presented in [3]. However, the neck of the manikin is not correctly reconstructed. This is mainly due to the specular effect of bright light on the brilliant surface of the manikin.

Figure 7 presents another experiment where we can note that several curves of the projected pattern are not correctly inpainted. This problem arises on the part of the picture where the curves are not correctly detected by the algorithm proposed in [3]. It also illustrates the shadow problem. While using the regular pattern, the input image does not contain shadows. However, using the inverted pattern, shadows appear and are detected by the curves detection algorithm as curves of the projected pattern (Figure 7 top right). However, the method proposed in Section V-B does not detect these artifacts as curves, and thus they are just ignored during inpainting as illustrated on Figure 7 (bottom).

Note that in the presented experiments, lines in the projected pattern are sparser than lines in the original pattern. Thus results of the 3D reconstruction are less dense than they should be. However, if we consider the dense pattern, the textures not covered by curves are missing for the inpainting. Therefore, it is important to strike a balance between a dense reconstruction and the textures recovery.

Our algorithm, coded in c, no parallelization runs in less than 0.3 second for an input size of 720×480. Our algorithm can be easily parallelized by running one process for each line to inpaint. Thus textures can be retrieved in real time on a smart-phone gpu.

The experiments presented in this section and in Figure 4 show that our method is effective and gives acceptable inpainting results. However, the detection of the curves must be improved and the inpainting should give smoother results.

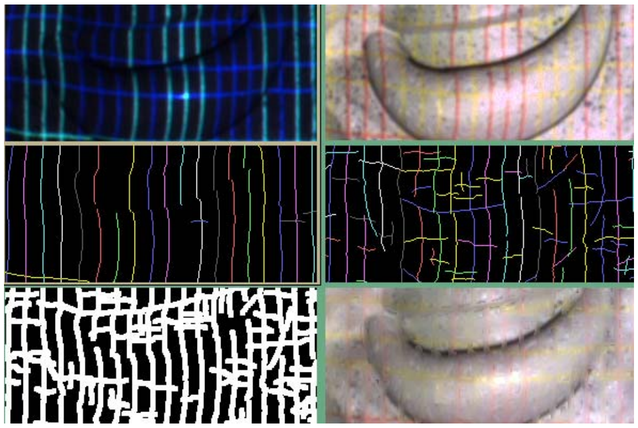
Figure 7. (top left) Regular pattern with its detected lines. (top right) Inverted pattern with detected lines. The shadows are detected as curves by the algorithm in [3]. (bottom) Detected lines to be inpainted and the result on inpainting.

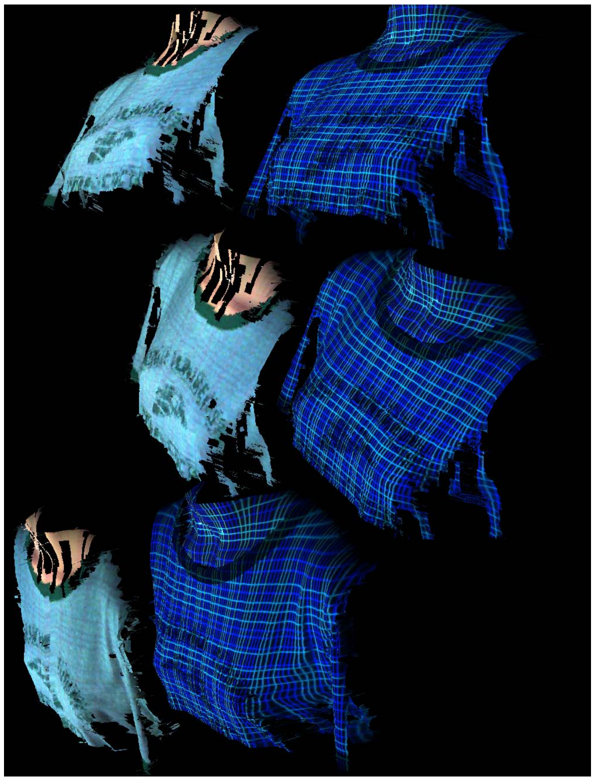
Figure 8. The reconstruction of a manikin wearing a t-shirt and the application of its texture, (right) using the regular pattern, (left) using the inverted and inpainted pattern.

VIII. Conclusion

In this paper, we presented a fully automatic method to inpaint an image that contains structured light used for 3D reconstruction. Our method uses two fast and simple steps. First, it automatically identifies the curves to be inpainted, and thus it requires no human interaction which is a feature required to perform a large number of images to inpaint. Second, using a patch based method, it selects a pair of patches for each curves to be inpainted in order to obtain good results. The experimental results, presented in Section VII, show that our method is fast and efficient if the curves of the projected pattern are correctly detected on the input image .

However, our algorithm can face some limitations when the projected pattern is too dense, i.e. when the lines on the projected pattern are not sparse enough to find patches with good texture information. This is a major problem since to obtain a good 3D reconstruction a dense pattern is required but at the same time, to obtain good texture a sparse pattern is required. As future work, we will improve the quality of inpainting when input images contain only little texture information.

Acknowledgment

This work was supported in part by Strategic Information and Communications R&D Promotion Programme(SCOPE) No.101710002 (Ministry of Internal Affairs and Communications, Japan), Grant-in-Aid for Scientific Research No.21200002 (Ministry of Education, Science, Sports and Culture, Japan) and Funding Program for Next Generation World-Leading Researchers No. LR030 (Cabinet Office, Government Of Japan) in Japan.

References

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009. 2

[2] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8:679–698, 1986. 3

[3] Hiroshi Kawasaki et al. Dynamic scene shape reconstruction using a single structured light pattern. pages 1–8, 2008. 1, 2, 3, 4, 5

[4] Ryusuke Sagawa et al. Dense 3d reconstruction method using a single pattern for fast moving object. 2009. 1, 2

[5] Jiaya Jia and Chi-Keung Tang. Image repairing: Robust image synthesis by adaptive nd tensor voting. In *CVPR*, pages I: 643–650, 2003. 2

[6] Alexandru Telea. An image inpainting technique based on the fast marching method. *journal of graphics, gpu, and game tools*, 9(1):23–34, 2004. 2