Texture Mapped Paper Pop-Ups

Armandarius Darmadji[1], Liliana[2]
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jalan Siwalankerto 121-131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) - 8417658
armandarius.darmadji@gmail.com[1], lilian@petra.ac.id[2]

ABSTRAK

Origamic architecture (OA) merupakan *papercraft* yang dapat mereplika struktur arsitektural, pola geometri, dan objek tiga dimensi (3D) lainnya dalam bentuk *pop-up* hanya dengan melipat dan menggunting satu buah kertas. Rancangan *image* 2-dimensi yang dapat direalisasikan menjadi OA disebut OA *plan*. Pemberian *texture* pada OA *plan* dapat digunakan untuk menampilkan detail visual pada OA yang dihasilkan. Akan tetapi, desain OA *plan* cenderung memiliki bentuk geometri yang berbeda dengan objek aslinya sehingga *texture* objek asli tidak dapat dengan mudah diberikan kepada OA *plan*.

Terdapat 2 metode untuk memberikan *texture* pada OA *plan*, yaitu 45^o *texture projection* dan 90^o *texture projection*. Metode dirancang agar dapat mencegah perembesan *texture*, meminimalisasi bagian OA *plan* yang belum mendapatkan *texture* (*holes*) dengan *inpainting*, dan meminimalisasi distorsi pada *texture* yang diberikan. Disediakan juga metode untuk memvisualisasikan OA *plan* yang diberi *texture* dalam 3-dimensi agar memudahkan pengguna dalam membayangkan hasil realisasi OA *plan*.

Hasil pengujian menunjukkan bahwa *texture* yang diberikan sudah berhasil mencegah perembesan *texture* dan meminimalisasi *holes*. Metode 90^o *texture projection* memberikan distorsi pada objek yang memiliki *texture* berstruktur. Metode visualisasi sudah dapat memvisualisasikan OA *plan* dengan baik.

Kata Kunci

Origamic Architecture, *texture mapping*, *inpainting*, dan *visualization*

ABSTRACT

Origamic architecture (OA) is a papercraft technique that can replicate structures, geometric shapes, and 3D objects in a paper pop-up that can be made by folding and cutting a single piece of paper. A 2D image can be made into OA called an OA plan. Texture mapping into an OA plan can be used to show visual details in the resulting OA. But OA plan tend to have a different geometric shape compared to the original object, therefore making the original object's texture hard to be mapped into the OA plan.

There are 2 methods to give texture into an OA plan, the 45^o texture projection dan the 90^o texture projection. The methods were designed to prevent texture leak in the resulting textured OA plan. Furthermore, the methods aimed to minimalize the "holes" (untextured part) by using inpainting technique and minimalize texture distortion of the resulting textured OA plan. Method to visualize textured OA plan in 3D to ease the user in imagining the realization of the resulting OA plan is also provided in this work.

The testing shows that the method succeeded in both preventing texture leak and minimalizing holes. The 90^o texture projection shows distortion on the object that have structured texture. Visualization method was able to show OA plan well.

Keywords

Origamic Architecture, texture mapping, inpainting, and visualization

1. PENDAHULUAN

OA merupakan *papercraft* yang dapat mereplika struktur arsitektural, pola geometri, dan objek tiga dimensi (3D) lainnya dalam bentuk *pop-up* hanya dengan melipat dan menggunting satu buah kertas. Mirip dengan *pop-up* lainnya, OA disimpan dengan melipat dua bagian terluar kertas sehingga apabila lipatan ini dibuka pada sudut tertentu akan menampilkan *pop-up* yang diinginkan. Pada umumnya, OA menampilkan *pop-up* yang diinginkan pada sudut 90^0, desain OA yang ditampilkan pada sudut 90^0 yang akan digunakan. Sudut 180^0 atau 360^0 adalah pilihan sudut lainnya dalam menampilkan *pop-up* yang diinginkan.

OA memberikan batasan dalam perancangannya. OA dibatasi hanya boleh menggunakan satu lembar kertas saja dan tidak mengizinkan pengeleman. Batasan ini membuat OA lebih susah dalam mereplika kontur objek 3D yang diinginkan. Terdapat beberapa algoritma yang dapat digunakan untuk menghasilkan OA secara otomatis [3], [5]. Algoritma yang digunakan dalam jurnal ini berdasar pada [3].

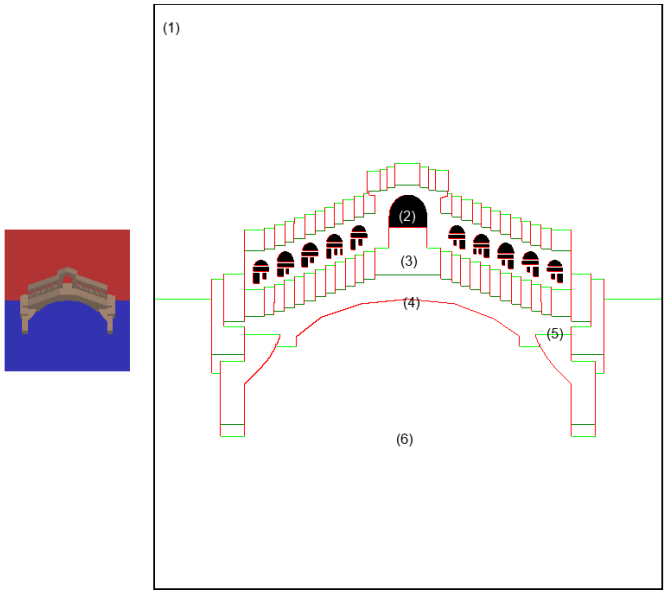
Algoritma ini menerima *input* objek 3D dan membuat *image* 2D yang dapat dibuat menjadi OA, *image* 2D ini disebut OA *plan*. Salah satu kelebihan dari algoritma ini adalah dapat mempertahankan tampilan visual OA dengan menjaga *surface* dan detail-detail halus dari objek 3D yang dimasukkan. Akan tetapi, OA *plan* yang dihasilkan cenderung memiliki bentuk geometri yang lebih sederhana dan berbeda dibandingkan objek 3D aslinya. Hal ini menyebabkan beberapa detail tampilan visual menjadi hilang. Oleh karena itu, diusulkanlah *texture mapping* untuk meningkatkan sekaligus mempertahankan detail tampilan visual yang hilang dalam proses pembuatan OA *plan*.

2. TEORI DASAR

2.1 *Origamic Architecture*

Algoritma yang digunakan dalam menghasilkan desain OA atau disebut juga OA *plan* secara otomatis berdasar pada *paper* berjudul "*Surface- and Contour- Preserving Origamic Architecture Paper Pop-Ups*". OA *plan* yang dihasilkan merupakan OA *plan* yang menghasilkan *pop-up* pada sudut 90°. Selain dapat menghasilkan desain OA secara otomatis, algoritma ini juga dapat memenuhi kondisi geometri yang ditetapkan untuk menjaga *foldability* dan *stability* pada *pop-up*. Desain yang dihasilkan dengan algoritma ini juga meyerupai hasil desain yang dibuat oleh *artist* sesungguhnya [3].

Contoh OA *plan* dapat dilihat pada Gambar 1. Kumpulan *pixel* berwarna putih pada OA *plan* menandakan 1 *patch* atau bidang pada OA (nomor 3). Tiap *patch* akan dibatasi dengan potongan atau lipatan, dimana potongan direpresentasikan dengan garis berwarna merah (nomor 4) dan lipatan direpresentasikan dengan garis berwarna hijau (nomor 5). Kumpulan warna hitam pada *patch* menandakan lubang atau bagian yang dihilangkan (nomor 2). *Patch* paling bawah disebut *ground patch* (nomor 1) dan *patch* paling atas disebut *back patch* (nomor 6), nama ini diberikan karena ketika OA *plan* dibuat dalam 3-dimensi, *patch* ini akan berada paling bawah (*ground*) atau paling belakang (*back*).



Gambar 1. Contoh OA *plan*

2.2 PatchMatch

Penelitian dalam *graphics* dan *vision* seringkali mengungkit pencarian *nearest neighbor* untuk menyediakan bermacam-macam metode *image editing*. Sayangnya, pencarian *nearest neighbor* seringkali membutuhkan waktu yang lama. Hal ini disebabkan pencarian dilakukan pada keseluruhan gambar untuk menemukan *match* terbaik. *PatchMatch* pada paper berjudul "*PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing*" menggunakan kesempatan ini untuk memperkenalkan algoritma baru dalam pencarian *nearest-neighbor* dengan cepat. Pada saat *PatchMatch* diperkenalkan, metode ini dapat berjalan 20 hingga 100 kali lebih cepat dibandingkan metode pencarian *nearest neighbor* sebelumnya, seperti kd-*tree*.

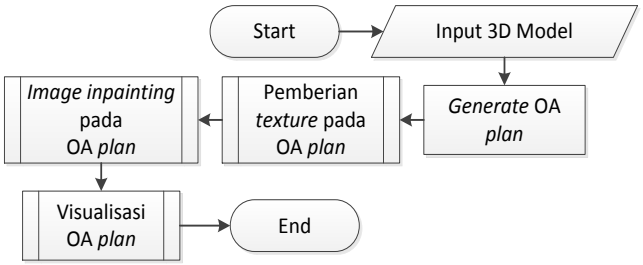
Kunci utama dalam metode *PatchMatch* adalah ide yang menyatakan bahwa beberapa *match* yang baik dapat ditemukan melalui pengambilan sampel secara acak, dan koherensi natural pada gambar dapat digunakan dalam membantu pencarian *match* disekitarnya. Algoritma *PatchMatch* yang cukup sederhana ini membuat *PatchMatch* dapat dijadikan basis dalam bervariasi *image editing tools*, seperti *image retargeting*, *completion*, dan *reshuffling* [1]. *Source code* untuk PatchMatch bersumber dari [2].

2.3 *Billboarding*

Billboard atau *imposter* merupakan istilah dalam penamaan gambar sebuah objek yang akan digunakan sebagai *texture map* [4]. *Texture map* akan dilakukan terhadap sebuah *polygon* yang pada umumnya memiliki dimensi lebih besar dibandingkan dimensi *texture* yang digunakan. Dimensi *polygon* yang lebih besar ini bertujuan agar semua bagian *texture* dapat dimuat. Namun, dampak dari penggunaan *polygon* yang lebih besar ini adalah akan terdapat bagian *polygon* yang tidak mendapatkan *texture*. Hal ini dapat diatasi dengan mentransparansikan bagian *polygon* yang tidak mendapatkan *texture* dengan menggunakan *alpha channel*.

3. DESAIN SISTEM

Sistem akan dimulai dengan memberikan *input* objek 3-dimensi kedalam sistem dan sistem akan menghasilkan OA *plan* berdasarkan objek yang diberikan. Sistem akan lanjut dengan memberikan *texture* kepad OA *plan* dan mengisi bagian yang belum mendapat *texture* dengan *image inpainting*. Tahap terakhir merupakan visualisasi OA *plan* menjadi 3-dimensi (OA). Garis besar alur kerja sistem dapat dilihat pada Gambar 2.

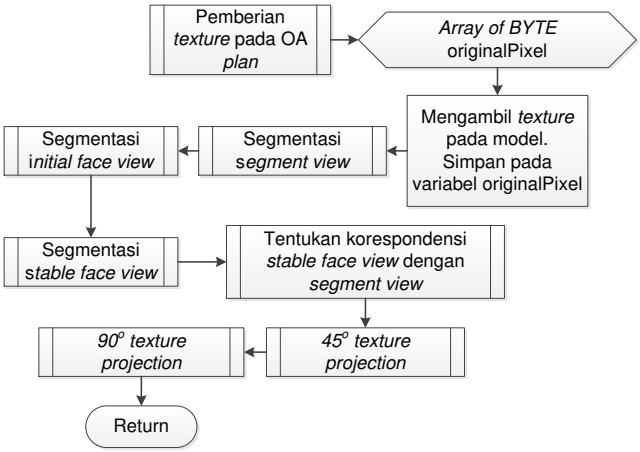


Gambar 2. Garis besar alur kerja sistem

3.1 Pemberian *Texture*

Tiga kriteria yang harus dipenuhi dalam pemberian *texture* adalah mencegah *texture leak*, meminimalisasi bagian pada OA *plan* yang tidak mendapatkan *texture* (*holes*), dan meminimalisasi distorsi pada *texture*. Untuk memenuhi kriteria, metode 45° *texture projection* dan metode 90° *texture projection* diajukan dalam proses

penentuan posisi *texture* yang akan diambil. Untuk mencegah *texture leak*, ditentukan korespondensi antara segmentasi objek asli dengan OA *plan* agar proses pemberian *texture* tiap segmen dapat dibatasi pada *patch-patch* tertentu saja. Garis besar alur kerja sistem pemberian *texture* dapat dilihat pada Gambar 3.



Gambar 3. Alur kerja sistem pemberian *texture*

3.1.1 Korespondensi OA plan dengan segmen

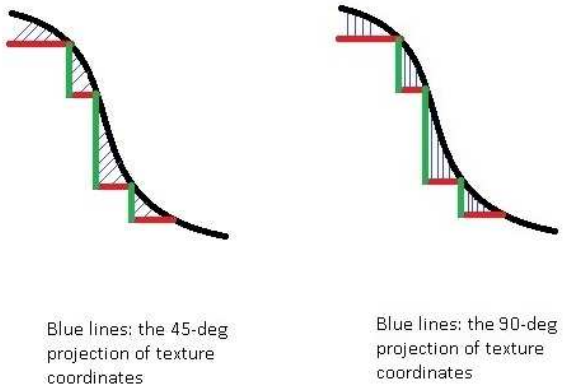
Korespondensi antara *stable face view* (OA *plan*) dengan *segment view* segmen bertujuan untuk mencegah *texture leak*. Ide utama dari proses ini adalah memberikan relasi pada tiap segmen objek asli agar hanya dapat memberikan *texture* yang dimiliki kepada 1 atau beberapa *patch* saja. Proses penentuan korespondensi dimulai dengan menentukan segmen yang hanya dapat mengisi satu buah *patch* saja. Proses selanjutnya dilanjutkan dengan menentukan segmen yang dapat mengisi 2 atau lebih *patch*. Proses terakhir dalam penentuan korespondensi anatar OA *plan* dengan segmen adalah menentukan korespondensi bagi segmen dan *patch* yang masih belum memiliki relasi dengan *patch* dan segmen manapun.

3.1.2 45° texture projection

45° *texture projection* akan memberikan warna *texture* pada *patch* di OA *plan* dengan memproyeksikan posisi *patch* tersebut terhadap kontur objek asli dengan sudut 45°. Warna *texture* pada kontur objek yang ditemukan pada sudut 45° akan diambil dan diberikan pada *patch*. Dengan metode ini, akan diasumsikan hasil pemberian *texture* akan terlihat paling baik pada sudut 45°. Metode 45° *texture projection* dapat dilihat pada Gambar 4.

3.1.3 90° texture projection

90° *texture projection* akan memberikan warna *texture* pada *patch* di OA *plan* dengan memproyeksikan posisi *patch* tersebut terhadap kontur objek asli dengan sudut 90°. Warna *texture* pada kontur objek yang ditemukan pada sudut 90° akan diambil dan diberikan pada *patch*. Dengan metode ini, akan diasumsikan hasil pemberian *texture* akan terlihat paling baik pada sudut 0° atau 90°. Metode 90° *texture projection* dapat dilihat pada Gambar 4.



Gambar 4. 45° dan 90° *texture projection*

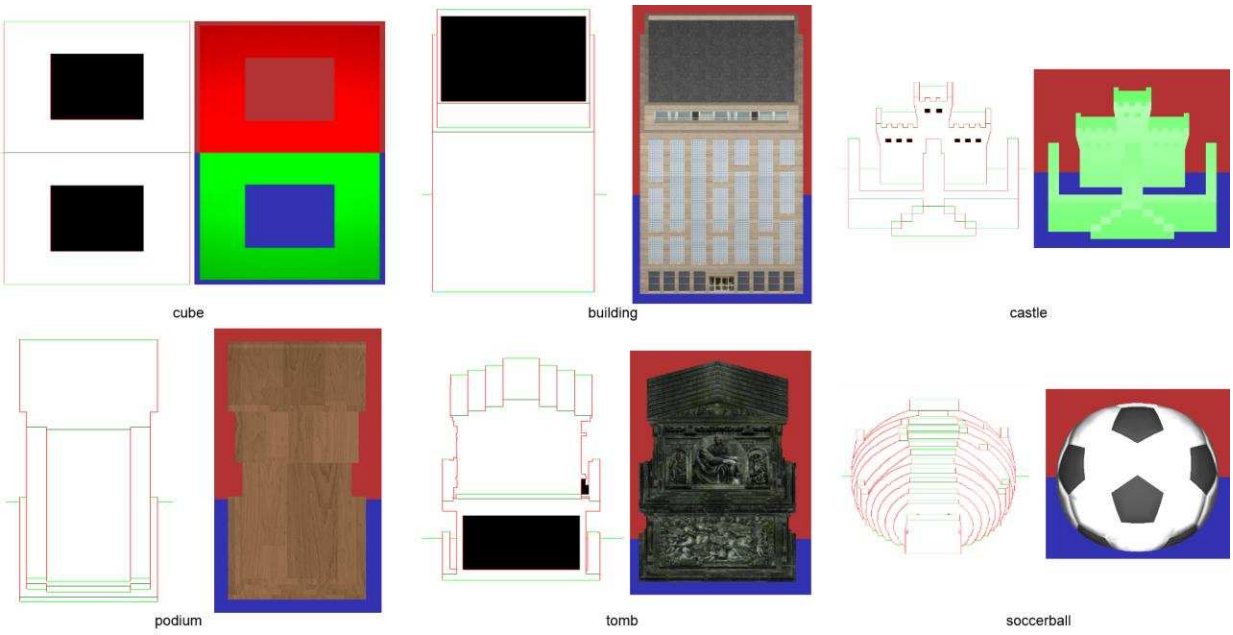
3.2 *Image Inpainting*

Image inpainting bertujuan untuk mengisi *holes* pada OA plan. Proses *inpainting* akan dilakukan pada tiap *patch* untuk mencegah *texture leak*. *Texture* dan *mask* tiap *patch* akan diambil dan kemudian di-*inpaint*. Setelah proses *inpainting* selesai dilakukan, *image* hasil *inpainting* akan dikembalikan pada OA *plan*.

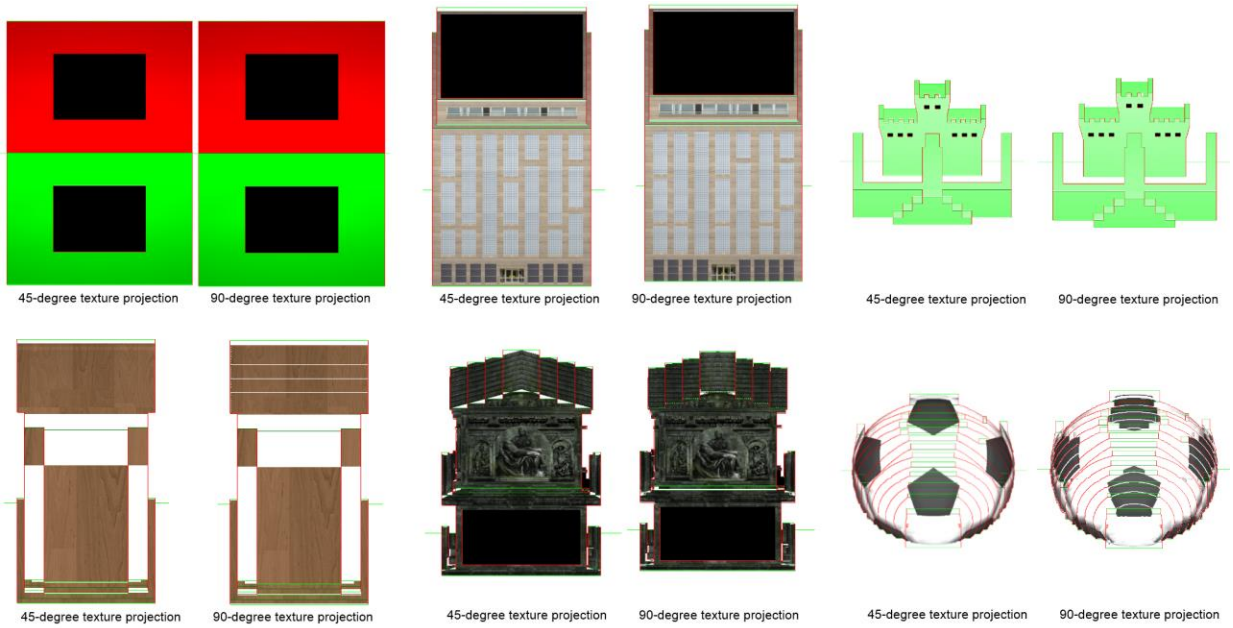
3.3 Visualisasi OA *plan*

Proses visualisasi bertujuan untuk merealisasikan OA *plan* (*image* 2-dimensi) menjadi 3-dimensi (OA). Secara umum, sistem perlu menentukan posisi 3-dimensi bagi semua *pixel* yang terdapat dalam OA *plan*. Penentuan posisi 3-dimensi dapat ditentukan dengan mentranslasi posisi 3-dimensi pixel pada OA *plan* yang sudah diketahui dengan metode *scanline* sesuai dengan level pembuatan OA, yaitu urutan *patch* yang ditentukan posisi 3-dimensinya terlebih dahulu. Proses visualisasi dimulai dari *ground patch*, karena pixel paling bawah kanan pada *ground patch* telah diberikan posisi 3-dimensi awal, hal ini menyebabkan *ground patch* memiliki level pembuatan OA bernilai 0. Proses visualisasi dilanjutkan dengan menghitung posisi 3-dimensi pada *patch-patch* yang berhubungan dengan *ground patch*, *patch-patch* ini memiliki level pembuatan OA bernilai 1. Proses visualisasi dilanjutkan lagi dengan menghitung posisi 3-dimensi pada *patch-patch* yang berhubungan dengan *patch* berlevel 1, *patch-patch* ini memiliki level pembuatan OA bernilai 2. Penentuan posisi 3-dimensi akan terus dilanjutkan sampai semua *patch* pada OA *plan* telah dihitung posisi 3-dimensinya.

Setelah selesai menghitung posisi 3-dimensi bagi semua *pixel* pada OA *plan*, sistem akan menyimpan *boundary box*, *texture*, dan *mask image* pada tiap *patch*. *Boundary box* dimaksudkan untuk menentukan *polygon* yang akan diberikan *texture*, dan *mask image* digunakan untuk menentukan transparansi. Dengan konsep *billboarding*, tiap *patch* dapat di-*render* untuk menghasilkan visualisasi OA *plan*.



Gambar 5. Objek yang digunakan dalam eksperimen



Gambar 6. Hasil pemberian *texture*

4. HASIL EKSPERIMEN

Dua buah set objek digunakan dalam melakukan eksperimen. Set pertama merupakan objek 3-dimensi yang memiliki desain OA *plan* cukup sederhana dan mirip dengan objek aslinya, yaitu objek *cube, building*, dan *castle*. Set kedua merupakan objek 3-dimensi yang memiliki desain OA *plan* cukup kompleks dan berbeda dengan objek aslinya, yaitu objek *podium*, *tomb*, dan *soccerball*. Set data yang digunakan dapat dilihat pada Gambar 5.

4.1 Hasil *Pemberian Texture*

Hasil pemberian *texture* dapat dilihat pada gambar 6. Dari hasil yang didapat, dapat dilihat bahwa *texture* yang diberikan sudah memenuhi kriteria mencegah *texture leak*. Hal ini dapat dilihat pada objek *tomb*, dimana *texture* pada atap tidak merembes ke tembok depan, dan sebaliknya *texture* pada tembok depan tidak merembes pada atap. Persentase kuantitas *holes* paling kecil yang dihasilkan pada eksperimen adalah 0.0000% pada objek *cube* dan paling besar adalah 30.6347% pada objek *podium*.

Berdasarkan pengamatan, kuantitas *hole* akan bertambah banyak apabila terdapat area *patch* diluar area objek aslinya. Persentase kuantitas *holes* dapat dilihat pada Tabel 1.

Tabel 1. Persentase kuantitas *holes*

Nama Objek	Persentase *holes* dengan metode 45° *texture projection*	Persentase *holes* dengan metode 90° *texture projection*
Cube	0.3000	0.0000
Building	2.4239	2.9664
Castle	0.8718	0.4029
Podium	29.0641	30.6347
Tomb	2.1804	2.5977
soccerball	2.3455	10.9696

Distorsi pada *texture* terjadi dalam metode 90° *texture projection* terutama pada objek yang memiliki *texture* berstruktur. Hal ini dapat dilihat pada hasil pemberian *texture* pada objek *soccerball*. Pola segilima pada objek *soccerball* tampak tidak menyatu. Hal ini disebabkan karena perhitungan posisi *texture* dengan metode 90° *texture projection* mengalami diskritisasi yang menyebabkan terjadinya efek *aliasing*.

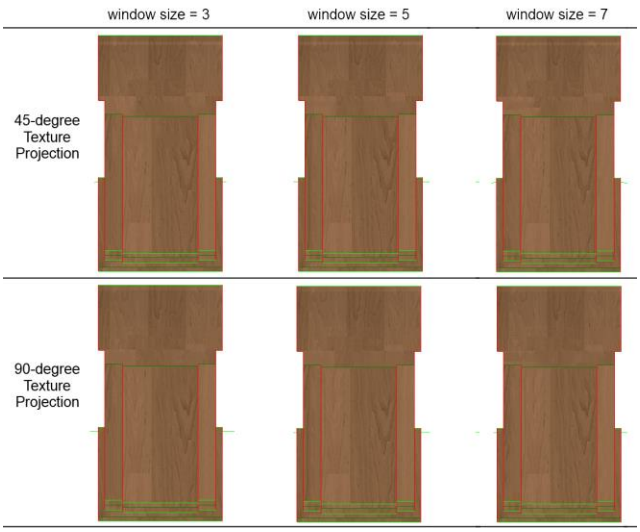
4.2 Hasil *Image Inpainting*

Hasil *image inpainting* pada *podium* dapat dilihat pada Gambar 7. Eksperimen dilakukan dengan mengubah ukuran *window* yang digunakan dalam algoritma. Ukuran *window* yang digunakan adalah 3, 5, dan 7. Perbandingan dilakukan dengan membandingkan tampilan visual dan waktu yang dibutuhkan. Perbedaan ukuran *window* mempengaruhi waktu secara signifikan. Hal ini dapat dilihat pada pertambahan waktu yang signifikan seiring bertambahnya ukuran *window*. Waktu yang digunakan dalam proses *image inpainting* dapat dilihat pada Tabel 2.

Berdasarkan hasil *image inpainting* yang didapat, perubahan ukuran *window* tidak mempengaruhi hasil *image inpainting* secara signifikan. Hal ini disebabkan kuantitas *holes* yang tidak begitu besar pada masing-masing objek dan pola *texture* yang berulang sehingga mudah di-*inpaint* (seperti misalnya *texture* kayu yang berulang pada objek *podium*).

Tabel 2. Waktu prose *image inpainting*

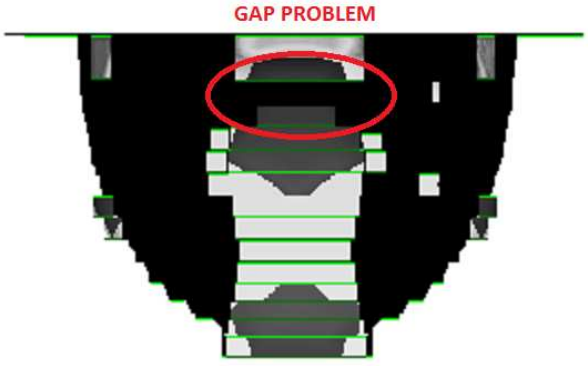
Nama Objek	Waktu dengan Ukuran *Window* (detik)		
	3	5	7
Cube	60	130	242
Building	199	516	896
Castle	78	188	416
Podium	96	343	728
Tomb	119	320	697
Soccerball	667	1898	4086



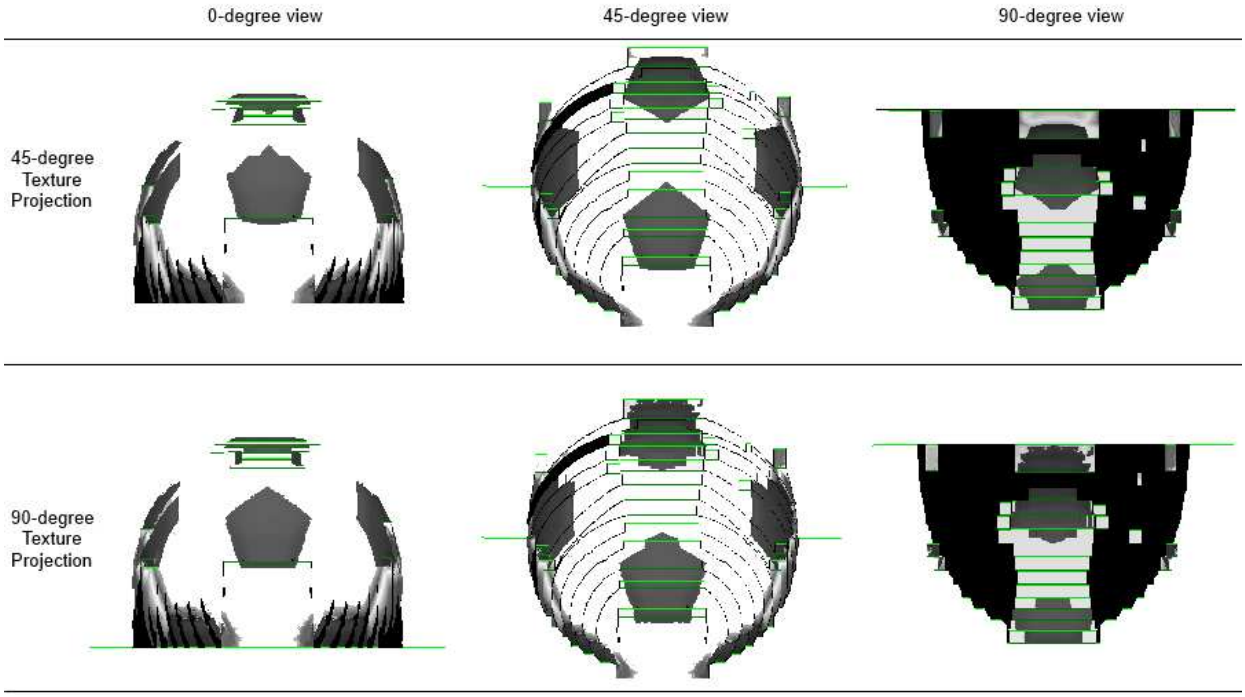
Gambar 7. Hasil *image inpainting* dengan ukuran *window* yang berbeda

4.3 Hasil *Visualisasi OA plan*

Dari hasil visualisasi, dapat dilihat bahwa OA *plan* yang diberi *texture* dengan metode 45° *texture projection* memiliki hasil visualisasi lebih baik pada sudut 45° dan OA *plan* yang diberi *texture* dengan metode 90° *texture projection* memiliki hasil visualisasi lebih baik pada sudut 0° atau 90°. Contoh kasus ini dapat dilihat pada hasil visualisasi objek *soccerball*. Pada sudut 45°, pola segilima pada *soccerball* terlihat menyatu pada OA *plan* yang diberi *texture* dengan metode 45° *texture projection* dan terlihat tidak menyatu pada OA *plan* yang diberi *texture* dengan metode 90° *texture projection*. Pada sudut 0°, pola segilima pada *soccerball* terlihat menyatu pada OA *plan* yang diberi *texture* dengan metode 90° *texture projection* dan terlihat tidak menyatu pada OA *plan* yang diberi *texture* dengan metode 45° *texture* projection. Hasil visualisasi OA *plan* objek *soccerball* dapat dilihat pada Gambar 8. Akan tetapi, terdapat masalah *gap problem*, *patch* yang tidak terhubung satu sama lain. Kesalahan ini dapat dilihat pada Gambar 9. Masalah ini dapat terjadi apabila terjadi kesalahan posisi *fold line* pada desain OA *plan*.



Gambar 9. *Gap problem*



Gambar 8. Hasil visualisasi OA *plan* pada objek *tomb* dan *soccerball*

5. KESIMPULAN

Dari hasil pengujian yang didapat, diperoleh kesimpulan sebagai berikut:

- Metode pemberian *texture* yang digunakan sudah dapat mengatasi permasalahan *texture leak*.
- Kuantitas *holes* akan semakin banyak apabila terdapat area *patch* pada desain OA *plan* yang berada terdapat di luar area objek 3-dimensi aslinya.
- Distorsi pada *texture* yang diberikan dapat terjadi dalam metode 90^o *texture projection*, terutama pada *texture* yang berstruktur.
- Metode *PatchMatch* yang digunakan dalam *inpainting* dapat mengisi *holes* pada OA *plan*. Ukuran *window* optimal dari segi hasil dan waktu dalam proses *inpainting* adalah 3.
- Algoritma yang digunakan dalam memvisualisasikan OA *plan* sudah dapat memberikan hasil visualisasi yang baik. Masalah *gap* antar *patch* dapat terjadi apabila terdapat kesalahan posisi lipatan pada desain OA *plan*.
- Hasil visualisasi OA *plan* dengan *texture* dari metode 45^o *texture projection* lebih baik dilihat pada sudut pandang 45^o, dibandingkan pada sudut pandang 0^o atau 90^o. Namun hal ini tidak berarti hasil visualisasi ini tidak baik dipandang dari sudut 0^o dan 90^o.
- Hasil visualisasi OA *plan* dengan *texture* dari metode 90^o *texture projection* lebih baik dilihat pada sudut pandang 0^o atau 90^o, dibandingkan pada sudut pandang 45^o. Namun hal ini tidak berarti hasil visualisasi ini tidak baik dipandang dari sudut 45^o.

6. REFERENSI

[1] Barnes, C., Shechtman, E., Finkelstein, A. & Goldman, D.B. (2009, August). *PatchMatch*: *A Randomized Correspondence Algorithm for Structural Image Editing*. Paper presented at ACM Transactions on Graphics (Proc. SIGGRAPH), New York, USA.

[2] [Java]-PatchMatch. Retrieved March 12, 2013, from http://www.developpez.net/forums/d947804/autreslangages/algorithmes/contribuez/java-patchmatch-inpainting-texture/

[3] Sang, L.N. (2013). *Surface- and Contour-Preserving Origamic Architecture Paper Pop-Ups*. Unpublished Ph.D. thesis, National University of Singapore, Singapore.

[4] Slater, M., Steed, A. & Chrysantou, Y. (2001). *Computer Graphics And Virtual Environments: From Realism to Real-Time*. USA: Addison-Wesley.

[5] X.-Y. Li, C.-H. Shen, S.-S. Huang, T. Ju, and S.-M. Hu. (2010) "*Popup: Automatic Paper Architectures from 3D Models*". Paper presented at ACM SIGGRAPH 2010 papers, pp. 1–9.