Perbesaran Gambar Digital 2-Dimensi dengan Algoritma *Pixel Art Scaling*

Ahmad Ayyub Mustofa (13506020)

Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung
Jl. Ganeca 10, Bandung 40132
email: if16020@students.if.itb.ac.id

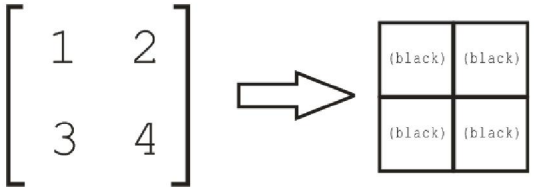
ABSTRAK

Sebuah gambar yang disimpan dalam bentuk data digital dapat ditampilkan oleh layar komputer dengan pencitraan menggunakan *picture elements* atau pixel. Pixel merepresentasikan sebuah gambar dalam kumpulan titik-titik kecil yang menyimpan informasi warna tertentu pada suatu koordinat di layar. Jumlah pixel yang digunakan dalam pencitraan dinyatakan dalam satuan resolusi. Semakin tinggi resolusi suatu gambar, semakin banyak jumlah pixel yang digunakan. Ini berarti bahwa informasi warna per titik semakin detil, dan gambar akan semakin tajam.
Salah satu masalah yang kerap kali ditemui dalam pencitraan gambar secara digital adalah bagaimana menampilkan sebuah gambar yang memilik resolusi rendah ke dalam resolusi tinggi tanpa merusak kualitasnya. Saat ini telah berkembang berbagai jenis algoritma yang dapat digunakan untuk menampilkan gambar resolusi rendah ke dalam resolusi tinggi dengan baik, bahkan secara real-time. Algoritma demikian umum disebut sebagai *"pixel art scaling algorithm"*.

Kata Kunci: pixel art, scaling, algorithm

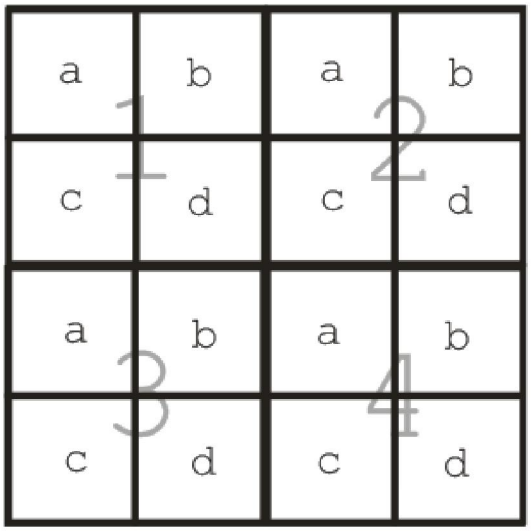
1. PENDAHULUAN

Sebuah gambar yang ditampilkan dalam bentuk digital sejatinya adalah kumpulan dari titik-titik warna pada layar yang membentuk suatu kesatuan. Setiap titik mengandung warna tertentu, dan terletak pada posisi tertentu dalam suatu satuan ukuran atau resolusi. Dengan setiap titik terletak secara benar pada posisi yang seharusnya, akan dibentuk suatu gambar sesuai dengan yang diinginkan. Pixel pada suatu gambar dapat diibaratkan sebuah matrix ukuran m x n, dengan setiap elemen matriksnya adalah sebuah titik pixel dengan warna tertentu. Sebagai contoh, sebuah persegi berwarna hitam dengan resolusi 2x2 pixel direpresentasikan sebagai berikut :



Gambar 1: Representasi *picture elements* dalam matriks

Bagaimana jika gambar ini hendak ditampilkan dalam ukuran yang lebih besar? Misalnya, gambar ini hendak ditampilkn dalam ukuran 4x4 pixel. Secara umum, perbesaran dilakukan dengan cara memetakan pixel yang ada ke dalam pixel-pixel baru dengan jumlah pixel yang lebih banyak. Posisi dari tiap pixel baru ini didapat dengan mengambil posisi dari masing-masing pixel semula dan memindahkannya ke dalam posisi baru dengan rasio pergeseran sesuai rasio perubahan resolusi yang terjadi. Ini dapat direpresentasikan dengan mengubah tiap elemen matriks menjadi sebuah matriks, sehingga matriks awal berubah menjadi kumpulan *matrix of matrices*. Sebagai contoh, persegi 2x2 pixel di atas apabila diperbesar ke dalam resolusi 4x4 pixel direpresentasikan sebagai berikut :



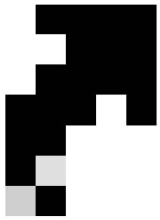
Gambar 2: Matriks perbesaran pixel sebesar 2x

Posisi tiap-tiap pixel baru ini dipetakan oleh posisi pixel semula. Ujung kiri atas dari pixel 1 menjadi ujung kiri atas untuk pixel 1a, kemudian posisi dari pixel 1b, 1c, dan 1d dipetakan dengan perhitungan dari posisi terhadap pixel 1a.
Dari gambar asli yang diberikan, sebenarnya hanya didapat diperoleh informasi dari 4 buah pixel yang telah ada, yaitu pixel 1, 2, 3, dan 4. Pixel-pixel ini dipetakan ke dalam empat pixel baru yaitu pixel 1a, 2a, 3a, dan 4a. Informasi dalam pixel-pixel lainnya diperoleh dengan memperkirakan isi informasi yang seharusnya dimiliki dengan mengambil informasi dari pixel yang berdekatan dengannya.

Metode pengisian informasi kosong dengan perkiraan berdasarkan informasi di sekitarnya dikenal dengan istilah *interpolation*. Dengan demikian, metode pengisian pixel dengan perkiraan berdasarkan informasi pixel-pixel yang ada di dekatnya seperti ini disebut sebagai metode *nearest-neighbor interpolation* [5]. Ini adalah metode perbesaran resolusi gambar yang umum dipakai secara konvensional.
Pada kasus ini, gambar yang dipetakan adalah sebuah gambar blok berbentuk persegi, sehingga tidak ada masalah dalam perbesarannya. Tetapi apabila yang hendak ditampilkan adalah sebuah gambar dengan bentuk yang rumit, maka saat perbesaran dapat terjadi masalah berupa hilangnya informasi pixel yang dibutuhkan. Hal ini terjadi karena keterbatasan pixel sebagai bentuk pencitraan elemen digital. Pixel adalah sebuah titik berbentuk persegi dengan ukuran tertentu pada posisi tertentu. Ketika gambar menjadi rumit, maka bentuk persegi yang dimiliki oleh pixel dan representasi posisi pixel yang diperbesar tidak dapat dipetakan dengan baik. Sebagai contoh, perbesaran sebuah icon berbentuk panah ↗ apabila diperbesar 4x akan menjadi seperti ini :

Gambar 3: Icon panah yang diperbesar 4x

Dan apabila diperbesar 4x lagi, gambar tersebut akan menjadi seperti ini :



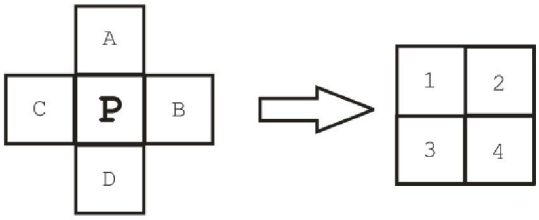
Gambar 4: Icon panah yang diperbesar 16x

Gambar yang diharapkan adalah gambar sebuah icon panah dengan bentuk kurva yang melengkung mulus, akan tetapi gambar yang dihasilkan telah kehilangan banyak sekali informasi pixelnya sehinggal menjadi bergerigi (*jaggy)*. Cacat pada pixel seperti ini disebut *alias*.
Untuk menghindari rusaknya gambar akibat perubahan resolusi ini, telah dikembangkan berbagai macam algoritma yang menarik.

2. METODE

2.1 Eric's Pixel Expansion (EPX)

EPX adalah algoritma *pixel art scaling* yang dikembangkan oleh Eric Johnston dari perusahaan LucasArts pada tahun 1992 [5]. Algoritma ini dikembangkan ketika perusahaan tersebut hendak mem-*port* sebuah *game* dari *platform* IBM PC ke dalam Macintosh. Pada saat itu, IBM PC berjalan pada gambar 256 warna dengan resolusi 320x200 pixel, sedangkan Macintosh berjalan pada resolusi sekitar dua kali lipatnya. Oleh karena itu, tujuan dari algorima ini adalah melakukan perbesaran sebuah gambar sebesar 2x dengan menjaga kualitas gambar tersebut.
Algoritma EPX memetakan sebuah pixel P ke dalam *matrix of pixels* berukuran 2x2, dan melakukan pengisian terhadap matriks tersebut dengan informasi-informasi pixel yang diperoleh dari perhitungan algoritma yang unik. Algoritma ini mengambil informasi pixel P dan 4 buah pixel yang mengelilingi P, yaitu satu pixel di atas, di kanan, di bawah, dan di kiri P, seperti pada gambar berikut :



Gambar 5: Pemetaan pixel P dan pixel-pixel yang mengelilinginya ke dalam matriks 2x2

Informasi yang didapat dari kelima pixel ini kemudian diproses dengan algoritma sebagai berikut [5]:

```
IF C==A => 1=A
IF A==B => 2=B
IF B==D => 4=D
IF D==C => 3=C
IF of A, B, C, D, 3 or more are identical: 1=2=3=4=P
```

Dari algoritma ini, dapat kita lihat bahwa EPX tidak serta-merta memetakan sebuah pixel ke dalam sebuah matriks, tetapi ia mengisi matriks tersebut dengan nilai pixel-pixel yang bersebelahan. Matriks 2x2 yang terbentuk akan berisi satu warna (blok) apabila dari 4 pixel yang bersebelahan dengan pixel P terdapat 3 atau lebih pixel yang berwarna sama, sedangkan apabila keempat pixel berisi warna yang berbeda-beda, maka matriks tersebut akan berisi warna yang berbeda-beda pula.

2.2 Scale2x/AdvMAME2x

Dari melihat algoritma EPX di atas dapat kita lihat bahwa kemungkinan terjadi kesalahan penggambaran pixel sangat mudah terjadi. EPX hanya membandingkan antara dua pixel yang berdekatan dan memetakannya ke matriks pixel yang bersesuaian. Apabila gambar yang bersangkutan menyimpan informasi pixel dengan tingkat kedalaman warna yang tinggi (misalnya untuk menghasilkan gradien), maka kemungkinan terjadi kesalahan sangat besar.
Pada tahun 2001, tim pengembang *emulator* [1]

[1] *Software* yang memungkinkan pengguna menjalankan sebuah *software* pada *platform* lain. Salah satu jenis *emulator* yang populer

AdvanceMAME menyusun algoritma yang bertujuan untuk memperbaiki kualitas gambar pada game-game lama yang diemulasikan dengan AdvanceMAME. Algoritma ini disebut Scale2x atau AdvMAME2x. Ide dari Scale2x adalah memperbesar ukuran bitmap dengan menebak isi pixel yang hilang tanpa teknik interpolasi. Secara matematis Scale2x memiliki cara kerja yang mirip dengan EPX, tapi dengan pendekatan algoritma yang lebih efisien dan lebih akurat.
Seperti pada EPX, Scale2x mengambil sebuah pixel P dan empat pixel yang mengelilinginya, lalu memetakannya ke dalam matriks 2x2. Informasi pixel yang didapat kemudian diproses sebagai berikut [5]:

```
1=P; 2=P; 3=P; 4=P;
IF C==A AND C!=D AND A!=B => 1=A
IF A==B AND A!=C AND B!=D => 2=B
IF B==D AND B!=A AND D!=C => 4=D
IF D==C AND D!=B AND C!=A => 3=C
```

Dari algoritma ini dapat dilihat bahwa sebelum dilakukan perbandingan pixel, terlebih dahulu matriks diinisialisasi dengan isi pixel P. Kemudian pixel-pixel yang mengelilingi P dibandingkan sedemikian rupa sehingga dapat dipetakan ke dalam matriks. Algoritma perbandingan yang digunakan bertujuan untuk membedakan antara kesamaan pixel yang terjadi karena gradien, blok warna, atau pola tertentu (misalnya garis atau sudut).
Hasil dari algoritma Scale2x ini dapat dilihat dalam gambar berikut :

Gambar 6: Game "Metal Slug" berjalan pada *emulator* AdvanceMAME tanpa Scale2x

Gambar 7: Game "Metal Slug" berjalan pada *emulator* AdvanceMAME dengan Scale2x

Algoritma Scale2x ini adalah salah satu algoritma yang populer digunakan pada *emulator*. Beberapa *emulator* yang menggunakan algoritma ini antara lain AdvanceMAME (*emulator arcade*), ScummVM (*emulator* SCUMM *engine*), Visual Boy Advance (*emulator* Game Boy Advance), DosBox (*emulator* DOS), dan lain-lain. Saat ini, algoritma Scale2x ini telah dikembangkan lebih lanjut untuk tingkat perbesaran yang lebih tinggi, dan menghasilkan Scale3x dan Scale4x.

2.3 Eagle

Eagle dapat dikatakan sebagai sebuah algoritma yang merupakan *cross-over* antara metode *nearest-neighbor interpolation* dan EPX. Berikut ini adalah algoritma yang digunakan dalam Eagle [5]:

```
First:
. . . --\ CC
. C . --/ CC
. . .

Then:
S T U  --\ 1 2
V C W  --/ 3 4
X Y Z
IF V==S==T => 1=S
IF T==U==W => 2=U
IF V==X==Y => 3=X
IF W==Z==Y => 4=Z
```

Eagle bekerja dengan memetakan sebuah pixel C ke dalam matriks 2x2 sebagaimana halnya metode *nearest-neighbor*. Kemudian dari pixel tersebut Eagle akan melihat pixel-pixel di atas dan di samping kirinya (pixel T, S, dan V). Apabila pixel-pixel tersebut memiliki isi yang sama, maka pixel kiri-atas pada matriks (pixel 1) diisi dengan nilai pixel tersebut. Ini kemudian diulangi untuk pixel di kanan-atas, kanan-bawah, dan kiri-bawah matriks yang dibentuk. Dapat dilihat bahwa cara kerja algoritma Eagle ini mirip dengan EPX. Perbedaannya adalah algoritma EPX diawali dengan

adalah *emulator* untuk menjalankan game-game konsol pada PC.

mengambil nilai sebuah pixel dan 4 pixel yang ada di sekitarnya (atas, bawah, kiri, kanan). Sedangkan Eagle bekerja dengan mengambil nilai sebuah pixel dan 8 pixel yang ada di sekitarnya.
Algoritma Eagle ini menjadi dasar pengembangan untuk algoritma 2xSaI yang sangat populer digunakan saat ini.

2.4 2x Scale and Interpolation Engine (2xSaI)

Mari kita telaah kembali algoritma Eagle. Sekilas algoritma ini terlihat cukup baik karena algoritma ini dapat memperbesar pixel suatu gambar tanpa ada informasi yang hilang, dan juga dapat menerjemahkan informasi gradasi warna. Akan tetapi, apa yang akan terjadi apabila pixel C berisi warna hitam sedangkan pixel S, T, U, V, W, X, Y, dan Z berisi warna putih? Ya. Warna hitam pada pixel C akan menghilang dan menjadi matriks 2x2 yang terbentuk hanya akan berisi warna putih. Ini menunjukkan salah satu kelemahan pada Eagle, yaitu ketidakmampuannya menerjemahkan pola warna (*pixel pattern*) pada gambar dengan tepat. Eagle dapat menangani gradasi, tapi pada kasus-kasus dimana pixel yang diterima adalah bagian dari sebuah sudut atau garis yang tipis, Eagle memberikan hasil yang kurang memuaskan.
Derek Liauw Kie Fa, atau lebih dikenal dengan *nickname* "Kreed", mengembangkan sebuah algoritma *pixel art scaling* berbasis Eagle, dengan tambahan berupa algoritma *pattern recognition*. Dalam 2xSaI, satuan pixel pembentuk gambar disebut sebagai texel (*texture element*) karena pembacaan informasi pixel ini juga akan menyimpan pola yang terdapat pada pixel-pixel di sekitarnya. Pola-pola tertentu akan menyatakan bentukan tekstur tertentu, dan hasil penerjemahan dari pola ini akan menghasilkan cara penggambaran pixel keluaran yang berbeda-beda.
Pada Eagle, pembacaan sebuah pixel disertai dengan pembacaan 8 pixel yang menyertainya. Pada 2xSaI, pembacaan pixel ini diperluas. Dari sebuah pixel dan 8 pixel yang mengelilinginya, dibentuk matriks 3x3. Kemudian dilakukan lagi pembacaan pixel di luar matriks tersebut dan disimpan pada index minus matriks 4x4.

Tabel 1. Pembacaan sebuah pixel pada algoritma 2xSaI dan matriks penyimpanannya [3]

	-1	0	1	2
-1	I	E	F	J
0	G	A	B	K
1	H	C	D	L
2	M	N	O	P

Pada tabel di atas, pembacaan pixel terjadi pada pixel D. Pada 2xSaI, pembacaan pixel secara berkelompok ini dinyatakan sebagai sebuah grup texel. Selanjutnya informasi texel ini akan dibandingkan dengan berbagai jenis *pattern* untuk menentukan bagaimana pixel keluaran akan digambar. Algoritma *pattern recognition* yang ada di dalam 2xSaI ini cukup rumit, sehingga tidak akan dibahas lebih lanjut di sini.

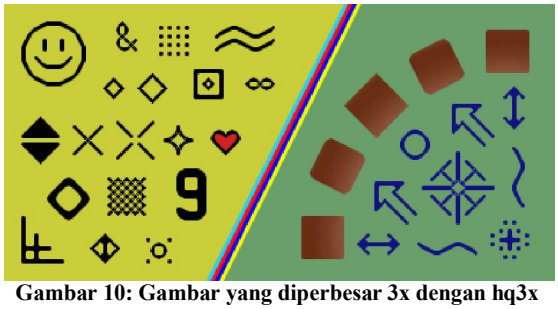
Baik 2xSaI maupun Eagle kini telah dikembangkan untuk menghasilkan gambar keluaran yang lebih halus. Varian-varian ini dikenal dengan nama Super 2xSaI dan Super Eagle. Super 2xSaI dan Super Eagle banyak digunakan di kalangan pengembang *emulator*. Beberapa yang terkenal di antaranya adalah ZSNES (*emulator* Super NES) dan ePSXe (*emulator* Sony PlayStation).

Gambar 8: Sebuah gambar 2-dimensi isometris, tanpa 2xSaI (kiri) dan dengan 2xSaI (kanan)

2.5 Varian-varian hq*n*x

Algoritma hqx dikembangkan oleh Maxim Stepin pertama kali untuk *emulator* ZSNES [4]. hqx terdiri dari 3 varian, yaitu hq2x, hq3x, dan hq4x. Masing-masing menyatakan proporsi perbesaran yang dilakukan. Apabila algoritma 2xSaI berfokus pada *pattern* yang terdapat pada gambar untuk menghindari kesalahan, maka algoritma hqx ini berfokus pada pembacaan warna dan gradasi untuk menghasilkan gambar yang lebih *eye-candy*.
hqx bekerja dengan membaca sebuah pixel P, kemudian membaca 8 pixel yang ada di sekitarnya. Pixel-pixel di sekitar ini kemudian diberi tanda yang menyatakan apakah isi warna di dalamnya dekat atau jauh dari nilai pixel P yang dibaca. hqx ini menyimpan sebuah tabel referensi yang digunakan untuk menentukan proporsi pixel masukan terhadap pixel keluaran.
Karena bentuk dari pixel keluaran ditentukan oleh penanda warna pada pixel-pixel di sekitar pixel yang sedang dibaca, terkadang algoritma ini memberikan keluaran berupa *pattern* yang sedikit salah. Namun kualitas warna yang dihasilkan dan kehalusan gambarnya memberikan hasil yang sangat memuaskan.

Gambar 9: Gambar yang diperbesar 3x dengan *nearest-neighbor interpolation*



Gambar 10: Gambar yang diperbesar 3x dengan hq3x

3. KESIMPULAN

Perbesaran gambar digital 2-dimensi secara biasa dapat menimbulkan *alias* karena adanya informasi-informasi pixel yang hilang. Secara umum, cara untuk menghindari terjadinya *alias* ini adalah dengan melakukan pembacaan tidak hanya terhadap satu pixel saja, tapi dengan ikut menyertakan pembacaan pixel-pixel yang ada di sekitar pixel tersebut untuk menentukan isi pixel-pixel keluarannya. Ini dilakukan dengan mengaplikasikan *pixel art scaling algorithm.*

Hal-hal yang menjadi fokus dalam pengembangan *pixel art scaling algorithm* antara lain adalah kehalusan gambar, ketajaman garis, serta ketepatan tekstur dan warna. Terdapat algoritma yang dikembangkan secara khusus untuk menangani keakuratan tekstur dan warna, dan terdapat juga algoritma yang dikembangkan secara khusus untuk mempertahankan kehalusan gambar dan ketajaman garis.

Efek balik dari penggunaan *pixel art scaling algorithm* ini adalah meningkatnya beban yang diperlukan untuk memroses suatu gambar. Pada algoritma-algoritma yang rumit, beban yang diperlukan untuk memroses suatu gambar dapat meningkat hingga belasan kali lebih berat.

REFERENSI

[1] Munir, Renaldi, *Diktat Kuliah IF2153 Matematika Diskrit Edisi Keempat,* Program Studi Teknik Informatika Institut Teknologi Bandung, Bandung, 2006

[2] AdvanceMAME Development Team, *Scale2x*, http://scale2x.sourceforge.net, 20/12/2009 – 20:00

[3] Ryan A. Nunn, *2xSaI Algorithm*, http://www.si-gamer.net, 20/12/2009 – 20:30

[4] Wikimedia Foundation, Inc. (anonymous), *Pixel Art Scaling Algorithm,* http://en.wikipedia.org, 20/12/2009 – 20:30

[5] Wikimedia Foundation, Inc. (anonymous), *Nearest-Neighbor Interpolation,* http://en.wikipedia.org, 20/12/2009 – 22:30

[6] Derek Liauw Kie Fa (a.k.a. Kreed), *Kreed's 2xSaI*, http://elektron.its.tudelft.nl, 21/12/2009 – 01:59

Filename: Makalah 2009 (Pixel Art Processing).docx
Directory: C:\Documents and Settings\Galaxy Rogue\My Documents\The Real IF\Strukdis
Template: C:\Documents and Settings\Galaxy Rogue\Application Data\Microsoft\Templates\Normal.dotm
Title: Panduan Penyerahan Makalah Ilmiah dengan Ukuran Huruf untuk Judul 16 Point Dicetak Tebal
Subject:
Author: DENI EKA PRASETYA
Keywords:
Comments:
Creation Date: 12/21/2009 2:44:00 AM
Change Number: 86
Last Saved On: 12/21/2009 7:50:00 AM
Last Saved By: Galaxy Rogue
Total Editing Time: 301 Minutes
Last Printed On: 12/21/2009 7:51:00 AM
As of Last Complete Printing
Number of Pages: 6
Number of Words: 2,460 (approx.)
Number of Characters: 14,026 (approx.)