# BAB II

# LANDASAN TEORI

## 2.1. Adaptor Tampilan

Display adaptor atau adaptor tampilan adalah perangkat keras tambahan yang ada di dalam komputer yang dapat digunakan untuk menampilkan citra grafis pada layar monitor selain untuk menampilkan text. Ada bermacam-macam jenis adaptor tampilan, antara lain: Hercules, MDA, CGA, EGA, VGA. Setiap jenis adaptor tampilan tersebut mempunyai kemampuan yang berbeda-beda, terutama dalam hal kemampuan menampilkan, baik dalam mode teks maupun mode grafis.

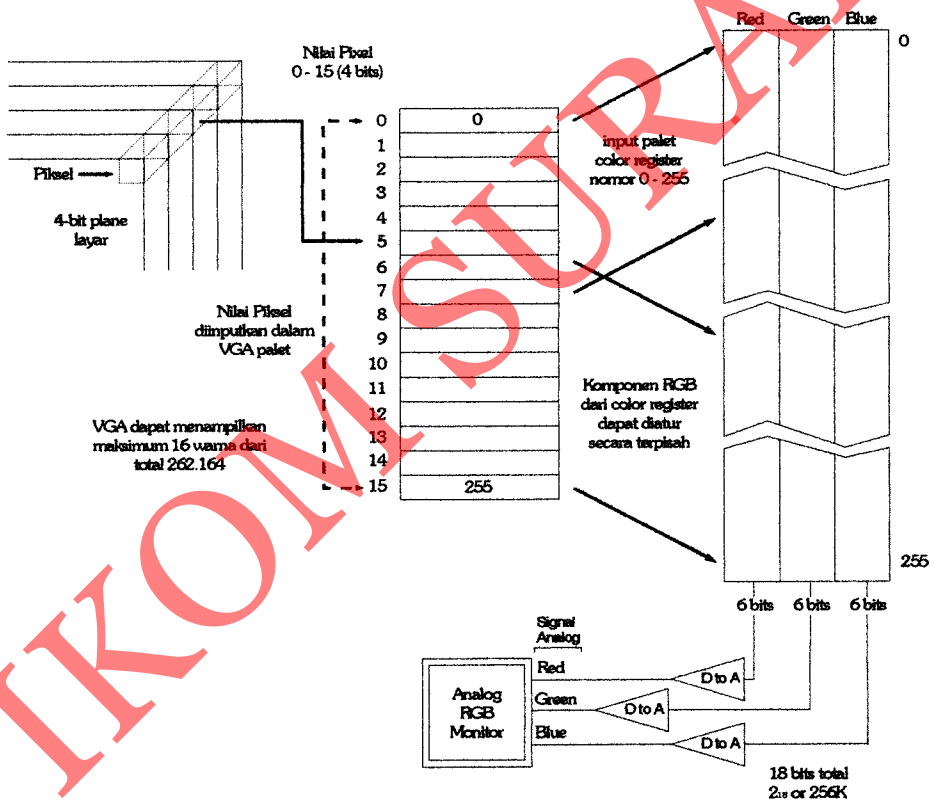
Tabel 2.1. Kemampuan adaptor tampilan

| Resolusi | Jumlah Warna | CGA | EGA | VGA |
|---|---|---|---|---|
| 320 x 200 | 4 | * | * | * |
| 640 x 200 | 2 | * | * | * |
| 320 x 200 | 16 | | * | * |
| 320 X 200 | 256 | | | * |
| 640 X 200 | 16 | | * | * |
| 640 X 350 | 2 | | * | * |
| 640 X 350 | 4 | | * | |
| 640 X 350 | 16 | | * | * |
| 640 X 480 | 2 | | | * |
| 640 X 480 | 16 | | | * |
| Palette warna dapat diatur | | | * | * |
| Register warna dapat diatur | | | | * |

Pada tugas akhir ini bukan hanya adaptor tampilan yang mendukung mode grafis saja yang dibutuhkan, tetapi juga adaptor tampilan yang dapat digunakan untuk menampilkan citra berwarna ke layar monitor. Untuk menampilkan citra berwarna ke layar monitor dibutuhkan suatu adaptor tampilan tertentu yang mendukung baik dalam hal mode dan resolusi. Selain itu dibutuhkan juga adaptor

tampilan yang palette warna dan register warnanya dapat diatur sesuai dengan warna-warna yang akan ditampilkan. Seperti tampak pada tabel 2.1, maka hanya adaptor tampilan jenis VGA saja yang dapat digunakan untuk menampilkan citra grafis pada layar monitor.

### 2.1.1. Palette




Gambar 2.1. Konfigurasi palette VGA

Dalam bidang komputer grafik, palette adalah koleksi dari warna-warna yang dapat dipilih untuk ditampilkan pada layar monitor warna secara serentak. Beberapa

jenis adaptor tampilan mempunyai cara yang berbeda-beda untuk menampilkan warna-warna pada layar. Selain itu cara untuk menampilkan warna-warna tersebut juga dipengaruhi oleh mode grafik yang dipergunakan pada adaptor tampilan tersebut. Cara untuk menampilkan warna pada adaptor tampilan jenis VGA dengan resolusi 640x480 piksel 16 warna, adalah seperti yang tampak pada gambar 2.1.

Adaptor tampilan jenis VGA pada mode grafik 640x480 piksel 16 warna menggunakan register palette sebanyak 16, yang dimulai dari 0 sampai 15. Masukan dalam register palette tersebut berisi pointer yang menunjuk ke salah satu dari register 256 warna. Informasi warna yang berasal dari register palette tersebut berupa sinyal dijital. Kemudian informasi warna yang berupa sinyal dijital tersebut diteruskan ke tabel warna DAC (Digital to Analog Converter) untuk dikonversi menjadi sinyal analog. Setelah informasi warna berubah menjadi sinyal analog, kemudian dikirimkan ke monitor warna untuk ditampilkan.

Untuk dapat menampilkan citra berwarna dengan warna yang tepat, perlu dibentuk palette yang mewakili warna dari citra yang akan ditampilkan. Pembentukan palette tersebut harus sesuai dengan warna yang dimiliki oleh citra yang akan ditampilkan, oleh karena itu pembentukan palette tersebut harus berdasarkan pada warna-warna yang terdapat didalam citra. Palette yang dibentuk ini kemudian diaktifkan sehingga citra berwarna dapat ditampilkan secara tepat.

Pada adaptor tampilan jenis VGA, informasi warna dari register palette dilewatkan ke tabel warna DAC untuk menghasilkan sinyal analog yang kemudian dikirimkan ke monitor VGA. Dengan DAC ini sinyal warna dijital dikonversi menjadi sinyal warna analog untuk dikirimkan ke monitor. Konversi ini diperlukan karena monitor VGA hanya mengerti sinyal analog dan bukan sinyal dijital.

Tabel warna DAC mempunyai 256 register, dimana tiap-tiap register tersebut

menyimpan informasi-informasi untuk setiap warna yang dipilih dari keseluruhan VGA palette yaitu lebih dari 262.000 warna. Jumlah warna ini merupakan hasil dari pengkodean warna 18-bit ($2^{18}$ = 262.144). Setiap masukan ke dalam tabel warna DAC terdiri dari 3 buah 6-bit nilai warna; dimana setiap komponen-komponen warna merah, hijau, dan biru yang masing-masing mempunyai lebar data sebesar 6-bit.

### 2.1.2. Video BIOS

ROM BIOS PC mempunyai banyak kemampuan yang dapat digunakan untuk berbagai macam keperluan layar tampilan. Semua kemampuan tersebut dikelompokkan sebagai fungsi-fungsi yang terdapat dalam interrupt 10h (fungsi interrupt video).

Pada mulanya fungsi-fungsi yang ada pada interrupt 10h hanya dapat digunakan untuk adaptor tampilan jenis MDA dan CGA. BIOS asli tidak mendukung adaptor EGA dan VGA atau kemampuan-kemampuan tambahannya baik pada mode teks maupun grafik. Untuk mengatasi hal itu adaptor tampilan jenis EGA dan VGA menyertakan fungsi-fungsi BIOS tambahan milik mereka sendiri pada chip ROM. Fungsi-fungsi tambahan ini aktif (dapat diakses) setelah sistem komputer melakukan proses *booting*.

Fungsi-fungsi tambahan ini kemudian berinteraksi dengan interrupt 10h untuk menambah fungsi-fungsi *interrupt* 10h yang ada pada BIOS dengan fungsi-fungsi yang dapat digunakan untuk mengakses adaptor tampilan jenis EGA dan VGA. Sehingga baik adaptor tampilan EGA maupun VGA dapat diakses melalui fungsi-fungsi yang ada pada interrupt 10h.

### 2.1.3. Fungsi-fungsi pada interrupt 10h

Fungsi-fungsi yang ada pada interrupt 10h adalah fungsi-fungsi yang digunakan untuk mengakses adaptor tampilan. Adapun fungsi-fungsi tersebut antara lain:

**a. Fungsi 00h**

Set video mode.

Input: AH=00h

AL=EGA/VGA video mode

AL=18: 640x480 grafik, 16 warna (khusus VGA)

Output: Tidak ada

Keterangan: Jika bit 7 dari register AL di set ketika fungsi ini dipanggil, maka isi dari video RAM tidak dihapus ketika ada perubahan mode video.

**b. Fungsi 0Ch**

Menulis piksel ke layar monitor.

Input: AH=0Ch

AL=Nilai warna

BX=Halaman video

DX=Baris layar

CX=Kolom layar

Output: Tidak ada

**c. Fungsi 0Dh**

Membaca piksel dari layar monitor.

Input: AH=0Dh

BH=Halaman video

DX=Baris layar

CX=Kolom layar

Output: AL=Nilai warna piksel

**d. Fungsi 0Fh**

Membaca mode default.

Input: AL=0Fh

Output: AL= Mode

**e. Fungsi 1Ah**

Memeriksa keberadaan dan jenis adaptor tampilan dan monitor yang terpasang.

Input: AH= 1Ah

Al = 0;

Output: Al = Mode

### 2.1.4. Bahasa C

Dalam tugas akhir ini digunakan bahasa pemrograman C dengan kompiler Turbo C versi 2.0. Bahasa C sering disebut sebagai bahasa tingkat menengah. Sebutan bahasa tingkat menengah tersebut bukan berarti negatif. Tidak juga berarti bahwa bahasa C mempunyai kemampuan yang kurang, sukar digunakan, atau kurang dapat dipakai untuk pengembangan dibandingkan dengan bahasa tingkat tinggi lainnya, seperti bahasa BASIC atau Pascal. Bukan pula bahwa bahasa C

tersebut mirip dengan bahasa Assembly dengan segala kesulitan-kesulitannya. Bahasa C disebut sebagai bahasa tingkat menengah karena bahasa C menggabungkan elemen-elemen dari bahasa tingkat tinggi dengan fungsi-fungsi dari bahasa Assembly. Gambar berikut memperlihatkan bahasa C dalam spektrum dari bahasa-bahasa pemrograman.

Tabel 2.2. Spektrum bahasa pemograman

| | |
|---|---|
| Bahasa Tingkat Tinggi | Ada |
| | Modula-2 |
| | Pascal |
| | COBOL |
| | FORTRAN |
| | BASIC |
| Bahasa Tingkat Menengah | C |
| | Forth |
| Bahasa Tingkat Rendah | Macro Assembler |
| | Assembler |

Selain itu bahasa C unggul dalam hal kecepatan, fleksibilitas, *user-friendly interface*, adanya fasilitas *debugger*, dan fasilitas untuk pembuatan *project*. Adanya fasilitas untuk pembuatan *project* ini mendukung kecepatan proses pembuatan program aplikasi dan kecepatan baik dalam pencarian kesalahan maupun pembetulannya.

Dalam Turbo C, *project* merupakan gabungan dari beberapa file program yang membentuk program aplikasi. Setiap *project* berhubungan dengan sebuah file *project*, yang menentukan file-file yang menjadi bagian dari *project* tersebut. Di dalam file *project* tersebut, terdapat nama-nama file program yang digabung untuk membentuk program aplikasi.

Alasan utama yang membuat bahasa C dipilih sebagai dasar implementasi

dalam tugas akhir ini adalah pustaka fungsi grafik yang disediakan bahasa C lengkap dan mudah digunakan. Pustaka grafik melakukan fungsi-fungsi yang penting seperti mendeteksi secara otomatis perangkat keras adaptor tampilan yang terpasang, menginisialisasi *driver* dengan benar untuk perangkat keras adaptor tampilan yang terpasang, memiliki fungsi-fungsi untuk melakukan manipulasi teks dan grafik pada *graphics output device*, dan lain sebagainya.

Untuk memakai fasilitas grafik yang disediakan oleh Turbo C, file pustaka grafik, yaitu graphics.lib harus di-*link* dengan kode sumber yang dihasilkan oleh kompiler turbo C. File graphics.lib ini berisi kode yang mengerjakan semua fungsi-fungsi grafik. Selain itu pada saat eksekusi *driver* grafik untuk adaptor tampilan dan semua file font yang dipakai harus tersedia. Karena dalam tugas akhir ini memakai adaptor tampilan jenis VGA, maka hanya file *driver* egavga.bgi saja yang harus selalu tersedia. Berikut ini beberapa fungsi-fungsi grafik yang tersedia dalam file pustaka grafik C.

**a. initgraph**

Inisialisasi mode grafik.

Deklarasi:

void far initgraph( int far *gdriver, int far *gmode, char far *driverpath );

Keterangan:

*gdriver adalah sebuah nilai integer atau konstanta yang merupakan nilai dari driver grafik yang akan digunakan.

*gmode adalah sebuah nilai integer atau konstanta yang menyatakan mode grafik. Jika diisi dengan konstanta DETECT, maka fungsi initgraph akan mengubah menjadi mode grafik yang tertinggi yang ada untuk *driver* yang tersedia.

driverpath digunakan untuk menentukan direktori file *driver* grafik dan file font berada.

Jika initgraph digunakan dengan mode autodetect, maka initgraph akan memanggil fungsi detectgraph untuk memilih driver grafik dan mode grafik. Fungsi initgraph ini akan mengembalikan nilai yang dilewatkan dari variabel internal graphresult. Dimana graphresult ini akan berisi nilai 0 jika fungsi initgraph bekerja dengan sukses atau kode kesalahan jika terjadi kesalahan pada saat fungsi initgraph berjalan.

**b. detectgraph**

Menentukan driver grafik dan mode gafik yang akan digunakan dengan melakukan pengecekan ke perangkat keras.

Deklarasi:

void far detectgraph(int far *gdriver, int far *gmode);

Keterangan:

*gdriver adalah sebuah nilai integer atau konstanta yang menyatakan driver grafik yang digunakan.

*gmode menyatakan mode grafik. Jika diisi dengan konstanta DETECT, maka fungsi initgraph akan mengubah menjadi mode grafik yang tertinggi yang ada untuk *driver* yang tersedia.

Fungsi detectgraph mendeteksi adaptor tampilan dan memilih mode tertinggi untuk adaptor tampilan yang tepasang. Jika tidak ada adaptor tampilan yang terdeteksi, maka detectgraph akan mengisi gdriver dengan nilai grNotDetected, dan graphresult akan mengembalikan nilai grNotDetected.

**c. putpixel**

menulis sebuah piksel pada layar monitor dengan warna color pada lokasi x,y.

Deklarasi:

void far putpixel(int x, int y, int color);

Keterangan:

x adalah kolom layar monitor.

y adalah baris layar monitor.

color adalah nilai warna piksel yang ditulis.

**d. getpixel**

Membaca warna piksel yang ada di layar monitor pada posisi x,y.

Deklarasi:

unsigned far getpixel(int x, int y);

Keterangan:

x adalah kolom layar monitor.

y adalah baris layar monitor.

Fungsi getpixel menghasilkan nilai warna dari piksel yang dibaca.

**e. setpalette**

Mengubah sebuah nilai warna dalam palette warna.

Deklarasi:

void far setpalette(int colornum, int color);

Keterangan:

Fungsi setpalette mengubah warna ke colornum dalam palette warna dengan warna color. Perubahan pada palette warna akan segera tampak di layar. Setiap perubahan palette warna dilakukan, semua warna yang tampak di layar akan

berubah ke nilai warna yang baru.

**f. setallpalette**

Mengubah seluruh palette warna.

Deklarasi:

void far setallpalette(struct palettetype far *palette);

Keterangan:

Fungsi setallpalette mengubah palette yang ada menjadi nilai-nilai yang diberikan melalui palettetype structure *palette. Fungsi setallpalette dapat mengubah sebagian atau keseluruhan palette warna dalam palette VGA.

**g. getpalette**

Membaca informasi palette yang ada.

Deklarasi:

void far getpalette(struct palettetype far *palette);

Keterangan:

Fungsi getpalette mengisi palettetype structure *palette dengan informasi mengenai ukuran dan warna-warna palette yang ada.

**h. getdefaultpalette**

Membaca definisi struktur palette.

Deklarasi:

struct palettetype *far getdefaultpalette(void);

Keterangan:

Fungsi getdefault palette membaca palettetype structure yang berisi palette yang diinisialisasi oleh fungsi initgraph.

Fungsi-fungsi yang terdapat pada pustaka grafik C seperti yang disebutkan di atas hanyalah beberapa fungsi yang cukup penting untuk disajikan disini.

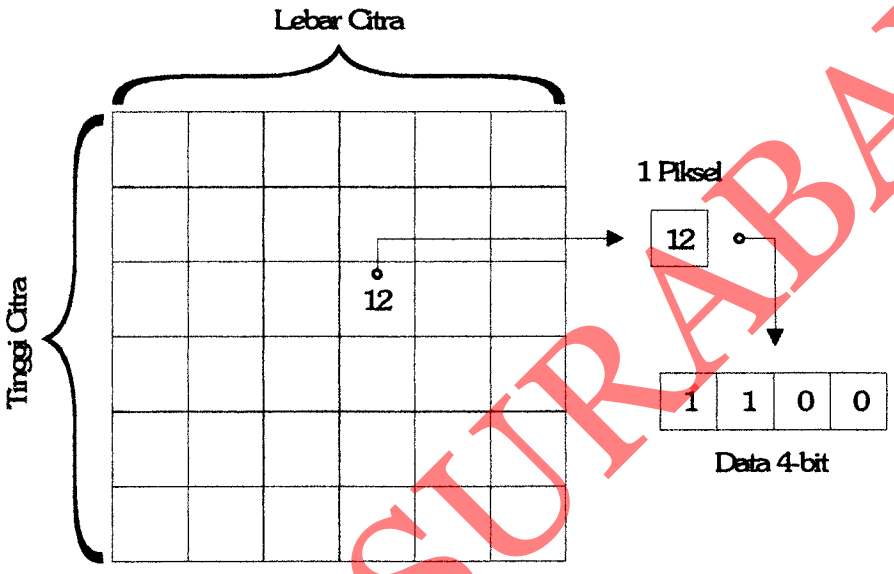
## 2.2. File Bitmap

File Bitmap adalah file yang digunakan untuk menyimpan data citra. Disebut bitmap karena file ini menggunakan metode penyimpanan yang sama dengan metode untuk menampilkan citra pada layar monitor, yaitu sebagai matriks atau array dua dimensi dengan elemennya berupa titik-titik. Dimana titik-titik ini disebut dengan *pixel* (piksel) yang merupakan singkatan dari *picture element*. Dalam komputer dijital informasi visual yang menyusun citra tersebut disimpan dalam satu atau lebih bit dalam memori komputer. Oleh karena itu menampilkan citra dengan metode ini berarti memetakan bit-bit dalam memori komputer ke layar monitor sehingga file jenis ini disebut dengan bitmap atau dapat diartikan sebagai peta bit.

Banyaknya bit yang dibutuhkan untuk menyimpan informasi mengenai piksel tersebut, tergantung dari jumlah maksimum warna yang dimiliki oleh file tersebut. Pada citra hitam putih hanya dibutuhkan satu bit saja untuk menyimpan informasi mengenai piksel pada memori komputer. Sedangkan untuk menyimpan informasi piksel pada citra yang memiliki jumlah warna sebanyak 16 dibutuhkan tempat sebanyak 4 bit ( $2^4 = 16$).

Seperti tampak pada gambar 2.2., sebuah citra digambarkan sebagai sebuah matriks. Setiap elemen dari matriks tersebut merupakan piksel-piksel yang membentuk citra. Jika citra yang diwakili oleh matriks tersebut merupakan citra yang memiliki warna maksimum 16 warna, maka setiap elemen matriks yang merupakan piksel-piksel pendukung citra tersebut akan diwakili oleh data dengan lebar empat bit dalam bentuk bilangan biner. Jika sebuah piksel mempunyai nilai

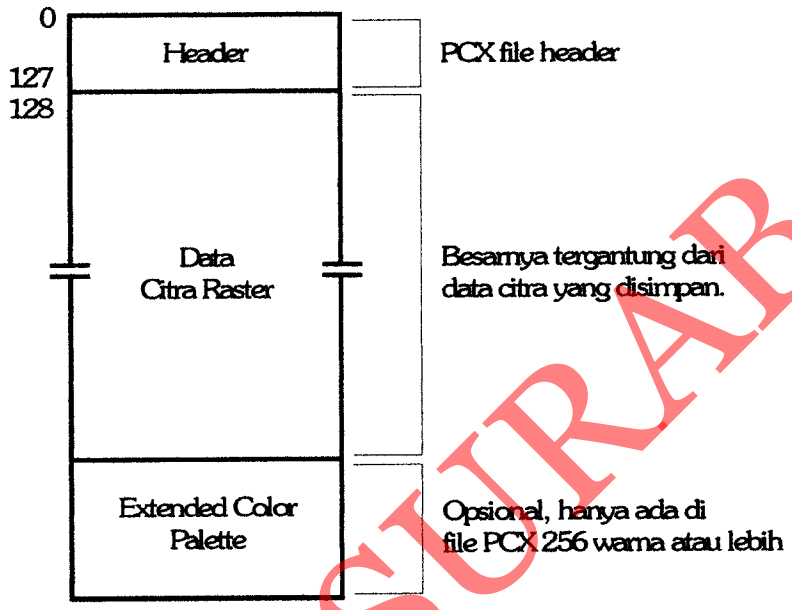
warna 12, maka dalam data dengan lebar empat bit tersebut, akan berupa sebuah bilangan biner dengan nilai 1100.




Gambar 2.2. Citra Dijital

Pada tugas akhir ini digunakan file bitmap dengan format PCX yang merupakan format asli *paint program* produksi Zsoft, yaitu PC Paintbrush. File bitmap dengan format PCX ini digunakan karena file ini mendukung mode 16 warna dan cukup dikenal di kalangan pengguna komputer.

### 2.2.1. Format file PCX




Gambar 2.3. Struktur file PCX

Struktur file PCX terdiri atas 3 (tiga) bagian, yaitu: header, data raster, dan Extended Color Palette. Header pada file PCX mempunyai ukuran sebesar 128 byte. Sedangkan ukuran data raster besarnya berbanding lurus dengan besarnya data citra yang disimpan dan berbanding terbalik dengan kompresi yang dapat dilakukan untuk data citra tersebut. Extended Color Palette pada file PCX sifatnya opsional. Extended Color Palette ini hanya ada pada file PCX 256 warna atau lebih, sedangkan pada file PCX yang kurang dari 256 warna Extended Color Palette tersebut tidak ada karena memang tidak digunakan.

**a. Header**

Dalam file PCX, header merupakan bagian penting yang berisi informasi yang digunakan untuk membaca data raster yang ada pada file tersebut sekaligus untuk menampilkan pada layar monitor. Susunan dari header file PCX adalah seperti yang tampak pada tabel 2.3 berikut ini.

Tabel 2.3. Struktur header file PCX

| Byte | Ukuran | Field | Keterangan |
|---|---|---|---|
| 0 | 1 | Manufacturer | Selalu 10 atau 0Ah |
| 1 | 1 | Version | 0 = Versi 2.5<br>2 = Versi 2.8 dengan palette<br>3 = Versi 2.8 tanpa palette<br>5 = Versi 3 (high color atau true color) |
| 2 | 1 | Encoding | Selalu 1 |
| 3 | 1 | BitsPerPixel | Jumlah bit per pixel dalam setiap plane |
| 4 | 2 | XMin | Batas tepi kiri gambar |
| 6 | 2 | YMin | Batas tepi atas gambar |
| 8 | 2 | XMax | Batas tepi kanan gambar |
| 10 | 2 | YMax | Batas tepibawah gmbar |
| 12 | 2 | HRes | Resolusi Horizontal dalam dpi |
| 14 | 2 | VRes | Resolusi Vertikal dalam dpi |
| 16 | 48 | ColorMap | Palette warna untuk EGA atau VGA 16 warna |
| 64 | 1 | Reserved | Dapat diabaikan ketika pembacaan.<br>Diisi 0 ketika penulisan. |
| 65 | 1 | NPlanes | Jumlah color planes |
| 66 | 2 | BytePerLine | Jumlah byte untuk setiap baris dalam satu plane |
| 68 | 2 | PaletteType | 1 = Warna atau Hitam Putih<br>2 = grayscale |
| 70 | 58 | Unused | Ruang kosong. Dapat diabaikan |

Sebelum kita dapat membaca data citra yang terdapat pada bagian data raster di dalam file PCX, maka terlebih dulu harus dibaca header file PCX. Header pada file PCX berisi informasi mengenai data citra yang di simpan dalam file tersebut. Struktur header file PCX selalu berurutan seperti yang tampak pada tabel 2.3. Diawali dengan field Manufacturer, kemudian Version, Encoding, BitsPerPixel, XMin, YMin,

XMax, YMax, HRes, VRes, ColorMap, Reserved, NPlanes, BytePerLine, PaletteType dan yang terakhir adalah field Unused.

Field Manufacturer yang merupakan identitas dari file PCX selalu berisi 10 atau 0Ah. Bila isi dari field Manufacturer tersebut bukan 10 atau 0Ah maka file tersebut bukan merupakan file PCX.

Field Version berisi informasi mengenai versi dari file PCX tersebut. Ada 4 (empat) versi file PCX, yaitu versi 2.5 yang ditandai dengan 0, versi 2.8 dengan palette yang ditandai dengan 2, versi 2.8 tanpa palette yang ditandai dengan 3 dan versi 3.0 yang ditandai dengan 5 pada field Version. Field Version pada dasarnya hanya digunakan untuk menentukan penggunaan jenis palette warna pada saat menampilkan gambar, yaitu antara palette warna yang ada pada field ColorMap pada bagian header file atau Extended Color Palette yang disimpan pada bagian akhir dari file PCX. Dalam file PCX 16 warna tidak ada Extended Color Palette. Dalam file PCX 16 warna ini cukup hanya menggunakan palette yang ada pada field ColorMap dalam header file PCX, maka pemeriksaan field Version cukup dilakukan dengan memeriksa apakah terdapat palette warna pada file PCX tersebut. File PCX yang memiliki palette warna adalah file PCX yang pada field Version-nya berisi nilai selain 3, yaitu file PCX yang tidak termasuk file PCX versi 2.8 tanpa palette.

Field Encoding akan selalu berisi nilai 1, yang menandai metode kompresi data yang digunakan untuk kompresi data citra yang disimpan pada bagian raster data. Nilai 1 pada field ini menandakan bahwa kompresi data menggunakan algoritma RLE (Run Length Encoding).

BitsPerPixel adalah field yang berisi informasi mengenai jumlah bit per piksel. Jumlah bit per piksel dalam file PCX tergantung dari jumlah warna maksimum yang dapat

ditampilkan oleh citra yang disimpan dalam file PCX. Untuk file PCX dengan 256 warna field BitsPerPixel ini akan berisi nilai 8 ($2^8 = 256$). Sedangkan untuk file PCX 16 warna field BitsPerPixel ini bukan berisi nilai 4 seperti yang seharusnya ($2^4 = 16$), melainkan berisi nilai 1 (satu). Karena untuk file PCX 16 warna, pemeriksaan jumlah warna tidak hanya dilakukan berdasarkan field BitsPerPixel saja, melainkan bersama-sama dengan field NPlanes. Field BitsPerPixel ini mempunyai ukuran sebesar 1 byte.

XMin, YMin, XMax dan YMax merupakan ukuran atau dimensi data citra yang disimpan pada file PCX tersebut.

HRes dan VRes merupakan informasi mengenai resolusi dari *device* yang membuat atau merekam file PCX tersebut.

ColorMap adalah field yang berisi palette warna untuk file PCX 16 warna atau kurang. Palette warna yang mempunyai ukuran sebesar 48 byte ini, terdiri dari 3 komponen warna Red, Green dan Blue. Sebuah warna piksel pada layar monitor dibentuk berdasarkan prosentase dari ketiga komponen warna tersebut. Masing-masing komponen warna Red, Green, dan Blue, mempunyai ukuran sebesar 16 byte. Dengan ukuran sebesar 16 byte untuk masing-masing komponen warna Red, Green dan Blue, maka ColorMap dapat dipakai untuk menyimpan palette warna sebanyak 16 warna pilihan yang disusun dari penggabungan ketiga komponen warna Red, Green dan Blue.

Field Reserved dapat diabaikan pada waktu proses pembacaan, sedangkan waktu proses perekaman field Reserved ini harus diisi dengan 0.

Field NPlanes berisi informasi mengenai jumlah color plane. Color plane dalam file adaptor tampilan jenis VGA adalah halaman memori, dimana dalam adaptor tampilan jenis VGA ini terdapat halaman memori sebanyak empat. Keempat color

plane tersebut, masing-masing menyimpan sebuah bit informasi warna yang secara berurutan Intensitas, Red, Green, dan Blue. Dengan menampilkan keempat color plane tersebut, akan dapat ditampilkan citra dengan 16 warna secara lengkap.
Field BytesPerLine berisi mengenai informasi jumlah byte per baris dalam setiap color plane.
Field PaletteType adalah informasi data citra yang disimpan dalam file untuk ditampilkan dalam *gray scale* atau warna. Field ini akan berisi nilai 1 untuk *gray scale* dan nilai 2 untuk warna.

**b. Data Raster**

Bagian data raster merupakan bagian yang digunakan untuk menyimpan data citra. Ukuran dari bagian ini berbanding lurus dengan besarnya data citra yang disimpan dan berbanding terbalik dengan kompresi data yang dapat dilakukan terhadap data citra yang disimpan. Besarnya data citra dipengaruhi oleh ukuran atau resolusi dari citra yang disimpan serta jumlah warna maksimum yang dimiliki oleh citra tersebut. Semakin besar ukuran atau resolusi citra, akan semakin besar pula ruang yang dibutuhkan untuk menyimpan citra ke dalam media penyimpanan. Semakin banyak warna yang dimiliki oleh citra tersebut, maka akan semakin banyak pula jumlah bit yang dibutuhkan untuk mewakili setiap piksel pembentuk citra tersebut. Semakin banyak jumlah bit yang dibutuhkan untuk mewakili setiap piksel pada citra, akan semakin banyak pula ruang yang dibutuhkan untuk menyimpan setiap piksel ke dalam media penyimpanan.

**c. RLE (Run Length Encoding)**

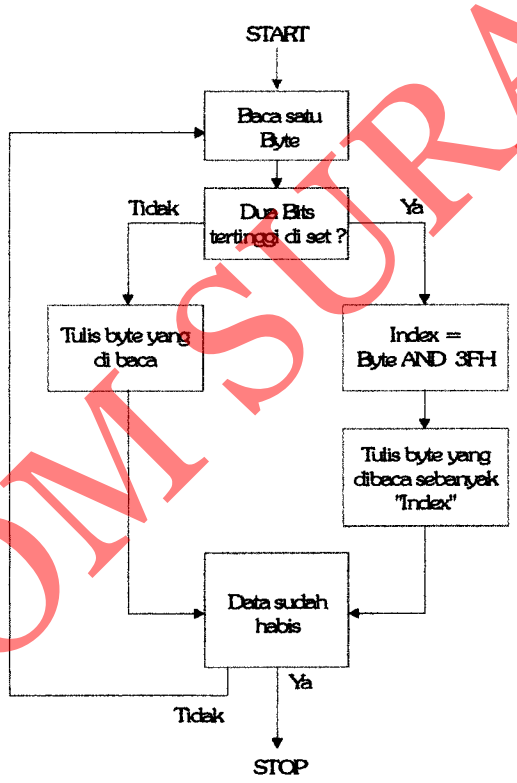
Pada file PCX kompresi data dilakukan dengan algoritma RLE (Run Length Encoding) yang ditandai dengan nilai 1 pada field Encoding pada header file.

Algoritma RLE adalah sebuah algoritma kompresi data yang melakukan kompresi dengan cara mengkodekan data yang diulang secara berurutan menjadi sebuah data sebesar 2 byte. Byte pertama digunakan untuk menyimpan tanda perulangan dan jumlah perulangan, dimana tanda perulangan ini menempati 2 bit tertinggi sedangkan jumlah perulangan menempati 6 bit terendah dari byte pertama. Sedangkan byte yang kedua digunakan untuk menyimpan data yang diulang. Kompresi data dengan algoritma RLE ini hanya dapat dilakukan untuk data yang diulang maksimum sebanyak 63 kali, karena bagian perulangan hanya menempati ruang sebesar 6 bit. Sehingga untuk data yang diulang lebih dari 63 kali, akan dilakukan pengkodean lebih dari satu kali.


Gambar 2.4. Pengkodean data byte dengan algoritma RLE

Untuk melakukan kompresi data dengan algoritma RLE ini, maka data yang akan dikenai kompresi, dibaca satu persatu, byte per-byte. Jika ada data yang diulang dan lebih kecil atau sama dengan 63, maka akan dilakukan pengkodean terhadap

data perulangan tersebut. Pengkodean dilakukan dengan mengisi 2 bit tertinggi dari byte pertama dengan 1, dan 6 bit berikutnya dengan jumlah perulangan yang ada. Kemudian byte data yang diulang diletakkan pada byte yang kedua. Hasil pengkodean ini kemudian disimpan ke dalam media penyimpanan. Jika byte data yang dibaca bukan merupakan suatu data perulangan, maka byte tersebut ditulis apa adanya ke dalam media penyimpanan.




Gambar 2.5. Diagram alir pembacaan data yang terkompresi dengan algoritma RLE

Tanda perulangan pada kompresi data dengan algoritma RLE ini merupakan 2 bit tertinggi pada byte pertama yang diisi dengan nilai 1, oleh karena itu jika

terdapat suatu byte data yang 2 bit tertingginya mengandung nilai 1 (11XXXXXX), maka byte data tersebut akan mendapat perlakuan khusus, yaitu dianggap sebagai suatu byte data yang diulang sebanyak 1 (satu) kali. Untuk adanya perlakuan khusus ini bukan terjadi kompresi data, melainkan suatu pemekaran data dari 1 (satu) byte menjadi 2 (dua) byte.

Sedangkan untuk melakukan pembacaan terhadap data yang dikompresi dengan algoritma RLE, data yang dikompresi dibaca satu persatu, byte per-byte. Jika pada satu byte yang dibaca 2 bit tertingginya berisi nilai 1, maka byte yang dibaca tersebut merupakan suatu byte kode perulangan data. Untuk itu dilakukan duplikasi terhadap byte berikutnya sebanyak nilai yang disimpan pada 6 bit terendah dari byte pertama.

## 2.3. Pengolahan Citra

Citra adalah gambar yang disimpan dalam media penyimpanan data. Citra tersusun atas titik-titik kecil yang membentuk suatu array dua dimensi. Titik-titik kecil ini disebut piksel (*pixel = picture element*) yang merupakan bagian yang terkecil dari citra itu sendiri. Jika suatu data citra di simpan dalam media penyimpanan, maka data mengenai piksel itulah yang disimpan. Besarnya ukuran penyimpanan data citra tergantung dari resolusi dan jumlah warna maksimum yang membentuk citra tersebut.

Pengolahan Citra adalah ilmu pengetahuan yang membahas citra dan data citra. Pengolahan Citra dapat juga diartikan sebagai suatu bentuk khusus dari *signal processing* dua dimensi yang digunakan untuk mendapatkan informasi mengenai citra. Secara umum Pengolahan Citra dapat digolongkan menjadi empat golongan, antara lain:

a. ***Point Processes***, yang memproses nilai piksel dalam suatu citra berdasarkan hanya pada nilai asli piksel dan kemungkinan pemindahan lokasi piksel tersebut di dalam citra yang dikenai proses.
b. ***Area Processes***, yang memproses nilai piksel berdasarkan pada nilai asli piksel dan nilai piksel yang mengelilinginya.
c. ***Frame Processes***, yang memproses nilai piksel di dalam suatu citra berdasarkan pada nilai piksel yang ada pada satu atau lebih data citra tambahan.
d. ***Geometric Processes***, yang melakukan proses pengaturan atau penempatan piksel dalam suatu citra berdasarkan pada transformasi geometrik. Proses memperbesar dan memperkecil citra termasuk dalam golongan ini.

Proses geometri pada citra adalah suatu proses yang mengubah posisi dan atau mengatur piksel dalam suatu citra yang didasarkan atas transformasi geometri. Transformasi geometri tidak mengubah nilai intensitas piksel melainkan hanya mengubah posisi piksel. Dalam hal ini nilai intensitas piksel yang ada pada citra sumber dipindah atau dipetakan ke lokasi yang baru yaitu citra tujuan.

Yang dimaksud dengan memperbesar atau memperkecil citra disini adalah memperbesar atau memperkecil ukuran atau resolusi citra. Dimana dalam proses ini suatu citra digunakan sebagai masukan yang kemudian diproses dan akan menghasilkan keluaran berupa citra juga. Citra hasil proses akan memiliki ukuran atau resolusi yang sudah berubah menjadi lebih besar untuk proses memperbesar, dan menjadi lebih kecil untuk proses memperkecil.

Penerapan proses geometri pada citra kadangkala menyebabkan hilangnya hubungan pemetaan piksel antara citra sumber dan citra tujuan. Misalkan dalam proses memperbesar citra. Jika citra diperbesar, maka satu piksel yang ada di citra sumber akan dipetakan kepada beberapa piksel yang ada di citra tujuan. Hal ini

terjadi sebaliknya pada proses memperkecil citra. Jika suatu citra diperkecil, maka beberapa piksel yang ada di citra sumber akan dipetakan kepada satu piksel yang ada di citra tujuan.

Untuk memastikan bahwa setiap piksel yang ada di citra tujuan berisi sebuah nilai dari hasil pemetaan piksel dari citra sumber ke citra tujuan tersebut, maka pemetaan dilakukan berdasarkan setiap ruang piksel yang ada di citra tujuan. Proses ini dapat disebut sebagai pemetaan balik.

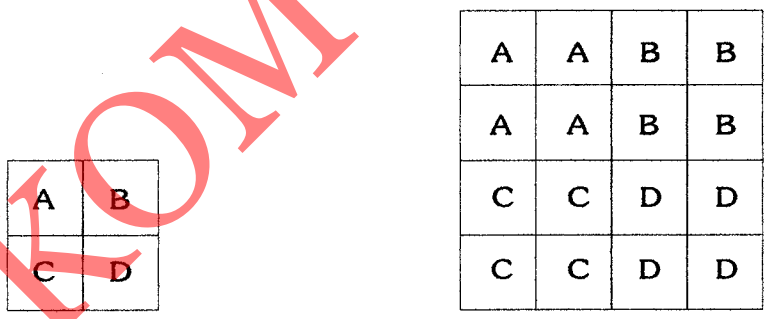
Meski dengan diterapkannya pemetaan balik ini dapat menjamin bahwa setiap piksel yang ada di citra tujuan memiliki sebuah nilai dari hasil pemetaan tersebut, namun pemetaan balik ini juga menimbulkan permasalahan lain yaitu timbulnya posisi piksel berupa bilangan pecahan. Hal ini terjadi ketika dilakukan perhitungan untuk mencari posisi piksel yang dicari pada citra sumber. Perhitungan untuk mencari posisi piksel yang dicari pada citra sumber ini dilakukan untuk mengetahui piksel-piksel pendukung yang terdapat di citra sumber. Piksel pendukung ini merupakan piksel pendukung untuk mencari sebuah nilai piksel baru. Perhitungan untuk mencari posisi piksel yang dicari pada citra sumber ini dilakukan dengan cara membagi posisi piksel yang dicari dengan faktor skala. Dari hasil perhitungan ini akan dapat menghasilkan nilai berupa bilangan pecahan. Dalam dunia nyata posisi piksel harus berupa bilangan integer atau bilangan bulat yang sesuai dengan baris dan kolom dari sebuah tampilan. Tidak ada baris dan kolom yang berupa bilangan pecahan.

### 2.3.1. Memperbesar citra

Memperbesar citra dapat dilakukan dengan beberapa metode, diantaranya yaitu metode replikasi dan interpolasi. Pada metode replikasi jika citra diperbesar

dengan faktor skala *n*, maka setiap piksel yang ada pada citra tersebut di perbesar menjadi *n*-kali (*n* = faktor skala). Perbesaran piksel ini dilakukan dengan cara menggandakan jumlah piksel, dari satu buah piksel pada citra sumber menjadi *n* piksel pada citra tujuan. Atau dengan kata lain perbesaran piksel ini dilakukan dengan cara memetakan satu piksel yang ada di satu lokasi dari citra sumber menjadi *n* piksel yang ada di *n* lokasi pada citra tujuan. Dengan cara ini, maka proses perbesaran citra dapat dilakukan dengan cukup mudah dan cepat, tetapi juga memiliki kelemahan. Kelemahan dari proses ini yaitu citra hasil proses perbesaran menjadi tampak bergeirigi dan kasar.

Misalkan sebuah citra dengan resolusi 2 x 2 akan diperbesar menggunakan metode replikasi dengan faktor perbesaran 2. Dari proses perbesaran ini akan menghasilkan citra dengan resolusi 4 x 4, yang hasilnya seperti tampak pada gambar berikut ini:




Gambar 2.6. Memperbesar citra dengan metode replikasi

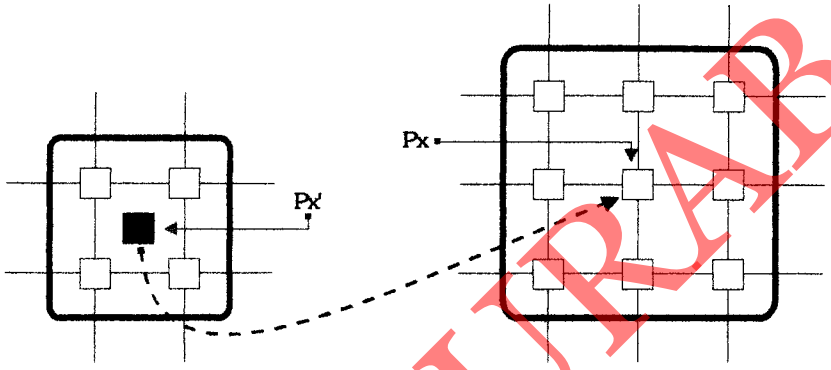
Metode yang kedua yang dapat digunakan adalah metode interpolasi. Dimana interpolasi yang digunakan disini adalah interpolasi linier. Interpolasi yang

diterapkan untuk piksel disini adalah sama dengan interpolasi yang digunakan untuk menghitung suatu nilai logaritma diantara tabel logaritma yang ada. Perbedaannya adalah jika interpolasi yang digunakan untuk menghitung nilai suatu logaritma merupakan perhitungan satu dimensi, sedangkan untuk interpolasi yang diterapkan pada piksel adalah interpolasi yang diterapkan secara dua dimensi, yaitu secara vertikal dan horizontal. Penerapan metode interpolasi secara dua dimensi pada proses memperbesar dan memperkecil citra ini dilakukan agar metode interpolasi dapat dikerjakan dengan efektif.

Dengan metode interpolasi ini maka proses yang dilakukan dapat menghasikan citra dengan mutu yang lebih baik dari pada menggunakan metode replikasi. Selain itu metode interpolasi dapat menyelesaikan masalah posisi piksel yang berupa bilangan pecahan sebagai akibat dari hasil perhitungan untuk mencari posisi piksel yang dicari pada citra sumber. Metode interpolasi digunakan untuk menghitung nilai suatu piksel baru yang terletak diantara piksel-piksel lain, yang merupakan piksel-piksel pendukung, yang nilainya sudah diketahui dan dengan jarak antara piksel-piksel tersebut diketahui juga. Interpolasi linier menganggap bahwa piksel-piksel pendukung yang berada disekitar piksel yang nilainya dicari, berpengaruh secara langsung atau secara linier sesuai dengan jarak masing-masing piksel dari piksel yang nilainya dicari. Dengan kata lain piksel yang mempunyai jarak terdekat akan memiliki pengaruh yang terbesar dan sebaliknya pada piksel yang memiliki jarak paling jauh dari piksel yang dicari.

Interpolasi yang diterapkan untuk mencari sebuah nilai piksel pada citra tujuan adalah interpolasi yang dilakukan secara dua dimensi. Interpolasi yang dilakukan secara dua dimensi disini sebenarnya adalah interpolasi biasa yang dilakukan sebanyak 3 kali. Interpolasi yang pertama adalah interpolasi secara hori-

zontal untuk piksel A dan B, dan interpolasi yang kedua adalah interpolasi secara horizontal untuk piksel C dan D, sedangkan interpolasi yang ketiga adalah interpolasi secara vertikal untuk hasil interpolasi horizontal A dan B dan hasil interpolasi horizontal C dan D.




Gambar 2.7. Pemetaan dari citra sumber ke citra tujuan

Untuk menghitung nilai piksel Px, maka harus diketahui terlebih dulu posisi piksel Px' yang berada di citra sumber. Piksel Px' merupakan piksel yang mewakili piksel Px pada citra sumber yang digunakan untuk mencari nilai piksel Px. Setelah posisi piksel Px' diketahui baru dicari perubahan jarak dari setiap piksel pendukung yang berada di sekeliling piksel Px' dengan piksel Px' yang dcari. Dalam prakteknya, tidak semua perubahan jarak dari setiap piksel tersebut perlu diketahui, tetapi cukup hanya perubahan jarak dari piksel pada pojok kiri atas dari piksel Px' saja yang perlu diketahui. Rumusan untuk perhitungan tersebut untuk mencari perubahan jarak dari piksel-piksel tersebut adalah sebagai berikut:

Jarak Kolom = Kolom Px' – Kolom A

Jarak Baris = Baris Px' – Baris A

Dimana Jarak Kolom dan Jarak Baris merupakan perubahan posisi piksel Px' dalam citra sumber yang dikurangi dengan posisi kolom dan baris dari posisi piksel terdekatnya, dalam hal ini adalah A. Jarak ini harus diketahui terlebih dahulu agar perhitungan interpolasi dapat dilakukan dengan benar. Posisi piksel Px' disini merupakan posisi Px di dalam citra sumber, dimana untuk baris Px' dapat diketahui dari perhitungan baris piksel Px pada citra tujuan dibagi faktor skala vertikal, sedangkan kolom Px' dapat diketahui dari hasil perhitungan kolom piksel Px pada citra tujuan dibagi faktor skala horizontal.

Setelah perubahan jarak piksel Px' diketahui, baru dapat dihitung interpolasi untuk mencari nilai Px. Rumusan interpolasi untuk menghitung nilai piksel Px adalah sebagai berikut:

IAB = Jarak Kolom * (Nilai piksel B – Nilai piksel A) + Nilai piksel A

ICD = Jarak Kolom * (Nilai piksel D – Nilai piksel C) + Nilai piksel C

Px = Jarak Baris * (ICD - IAB) + IAB

IAB adalah hasil perhitungan interpolasi horizontal antara piksel pendukung A dan B, dan ICD adalah hasil perhitungan interpolasi horizontal antara piksel pendukung C dan D, sedangkan perhitungan interpolasi vertikal dilakukan antara hasil interpolasi horizontal A dan B dengan interpolasi horizontal C dan D. Interpolasi vertikal ini adalah interpolasi yang ketiga yang menghasilkan nilai piksel Px yang dicari yang diwakili oleh piksel Px'.

Untuk memperjelas keterangan tersebut di atas, berikut ini disajikan contoh proses perhitungan dari proses memperbesar citra. Misalkan untuk sebuah citra dengan resolusi 2 x 2, diperbesar dengan metode interpolasi dengan faktor skala 2. Dari sini dapat diketahui resolusi atau ukuran dari citra tujuan dengan mengalikan resolusi citra sumber dengan faktor skala.

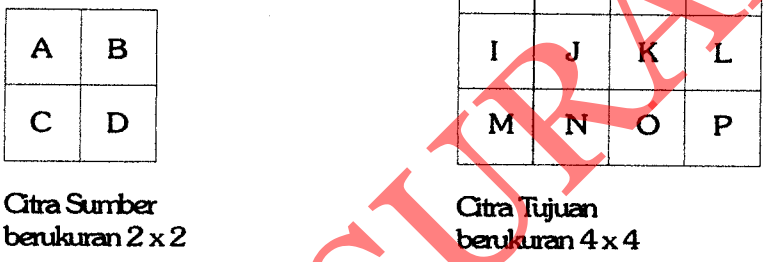
Maka dapat dihitung resolusi citra tujuan sebesar:

Jumlah Baris Citra Tujuan = faktor skala x resolusi vertikal

= 2 x 2 = 4

Jumlah Kolom Citra Tujuan = faktor skala x resolusi horizontal

= 2 x 2 = 4

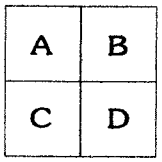



Citra Sumber berukuran 2 x 2

Citra Tujuan berukuran 4 x 4

Gambar 2.8. Memperbesar citra dengan metode interpolasi

Sedangkan untuk mengisi nilai-nilai piksel pada citra tujuan ($A_{tuj} - P_{tuj}$), digunakan metode interpolasi dengan menggunakan rumusan seperti yang telah disebutkan di atas. Misal citra sumber yang berukuran 2 x 2 berupa citra seperti pada gambar 2.9 berikut ini.

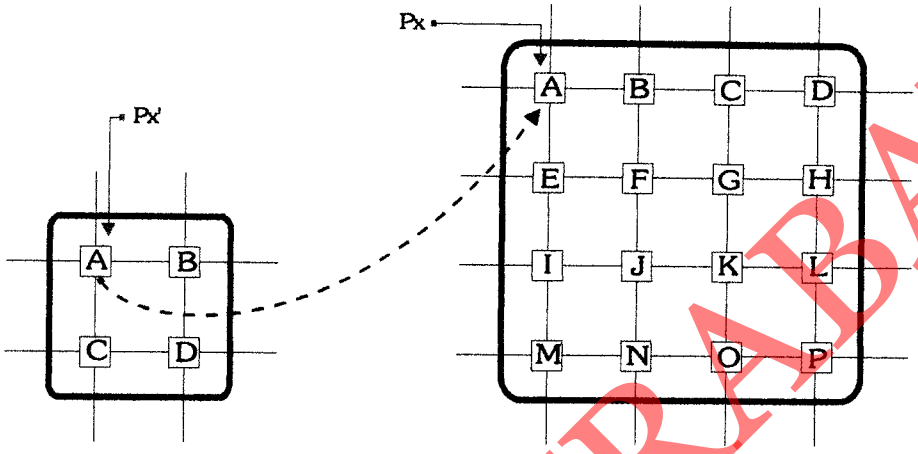



Citra Sumber berukuran 2 x 2

Gambar 2.9. Citra sumber berukuran 2 x 2

Dan perhitungan selengkapnya untuk mencari nilai-nilai piksel pada citra tujuan

adalah sebagai berikut:




Gambar 2.10. Pemetaan untuk piksel $A_{tuj}$

**a. Piksel $A_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $A_{sum}$ = 1 – 1 = 0

Jarak Baris = Baris Px' – Baris $A_{sum}$ = 1 – 1 = 0

IAB = Jarak Kolom * (B –A) + A

= 0 * (4 – 1) + 1

= 1

ICD = Jarak Kolom * (D –C) + C

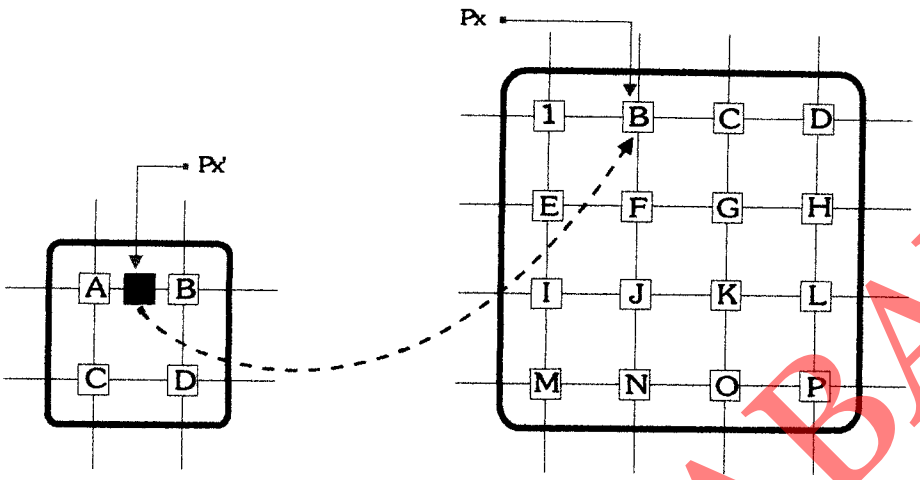
= 0 * (9 – 6) + 6

= 6

$A_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (6 – 1) + 1

= 1




Gambar 2.11. Pemetaan untuk piksel $B_{tuj}$

**b. Piksel $B_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $B_{sum}$ = 2 – 1,5 = 0,5

Jarak Baris = Baris Px' – Baris $B_{sum}$ = 1 – 1 = 0

IAB = Jarak Kolom * (B –A) + A

= 0,5 * (4 – 1) + 1

= 2,5

ICD = Jarak Kolom * (D –C) + C

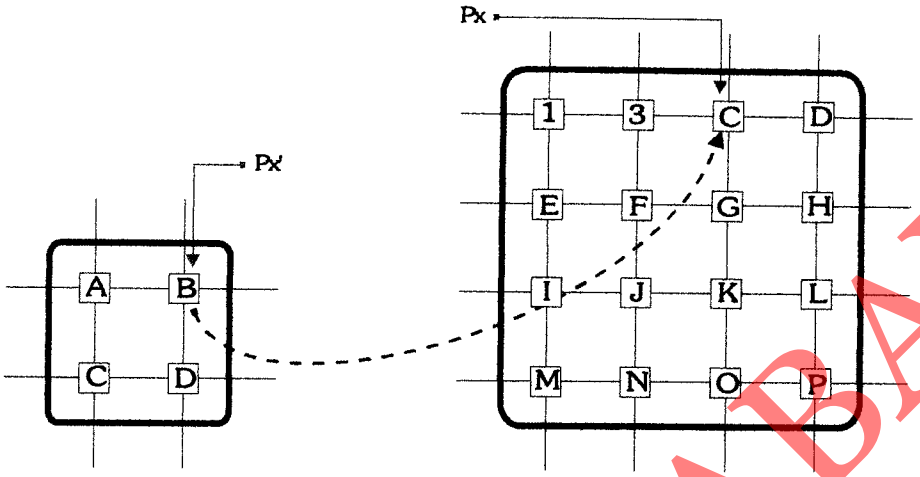
= 0,5 * (9 – 6) + 6

= 7,5

$B_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (7,5 – 2,5) + 2,5

= 2,5 » 3




Gambar 2.12. Pemetaan untuk piksel $C_{tuj}$

**c. Piksel $C_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $C_{sum}$ = 2 – 2 = 0

Jarak Baris = Baris Px' – Baris $C_{sum}$ = 1 – 1 = 0

IAB = Jarak Kolom * (B –A) + A

= 0 * (0 – 4) + 4

= 4

ICD = Jarak Kolom * (D –C) + C

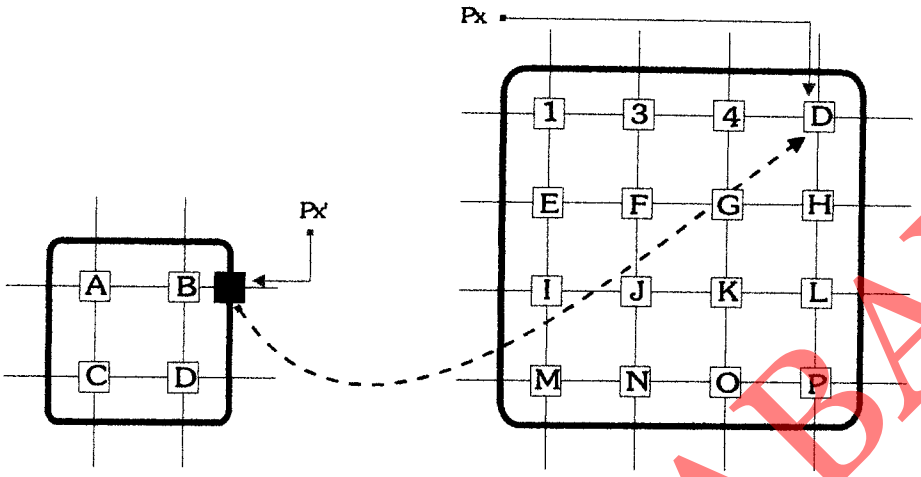
= 0 * (0 – 9) + 9

= 9

$C_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (9 – 4) + 4

= 4




Gambar 2.13. Pemetaan untuk piksel $D_{tuj}$

**d. Piksel $D_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $D_{sum}$ = 2,5 – 2 = 0,5

Jarak Baris = Baris Px' – Baris $D_{sum}$ = 1 – 1 = 0

IAB = Jarak Kolom * (B –A) + A

= 0,5 * (0 – 4) + 4

= 2

ICD = Jarak Kolom * (D –C) + C

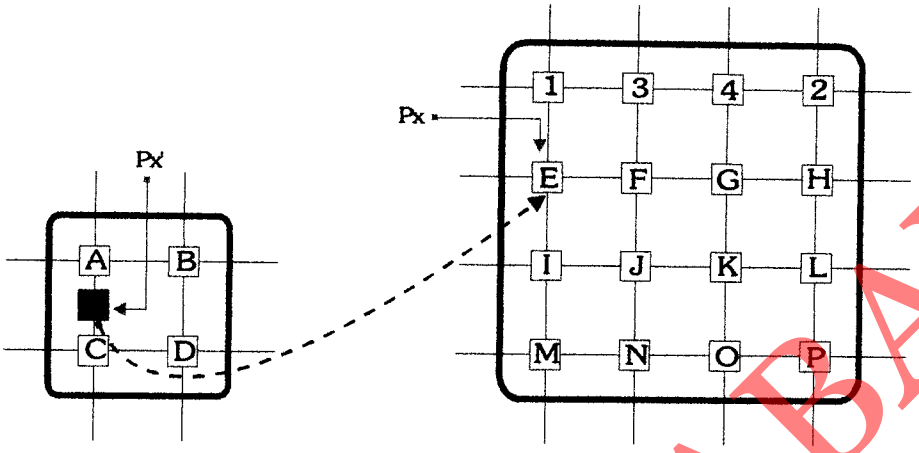
= 0,5 * (0 – 9) + 9

= 4,5

$D_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (4,5 – 2) + 2

= 2




Gambar 2.14. Pemetaan untuk piksel $E_{tuj}$

**e. Piksel $E_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $E_{sum}$ = 1 – 1 = 0

Jarak Baris = Baris Px' – Baris $E_{sum}$ = 1,5 – 1 = 0,5

IAB = Jarak Kolom * (B –A) + A

= 0 * (4 – 1) + 1

= 1

ICD = Jarak Kolom * (D –C) + C

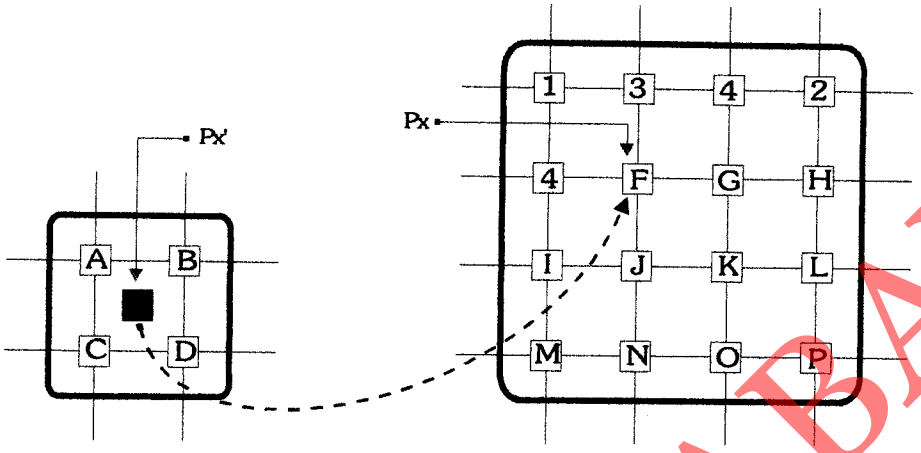
= 0 * (9 – 6) + 6

= 6

$E_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0,5 * (6 – 1) + 1

= 3,5 » 4




Gambar 2.15. Pemetaan untuk piksel $F_{tuj}$

**f. Piksel $F_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $F_{sum}$ = 1,5 – 1 = 0,5

Jarak Baris = Baris Px' – Baris $F_{sum}$ = 1,5 – 1 = 0,5

IAB = Jarak Kolom * (B –A) + A

= 0,5 * (4 – 1) + 1

= 2,5

ICD = Jarak Kolom * (D –C) + C

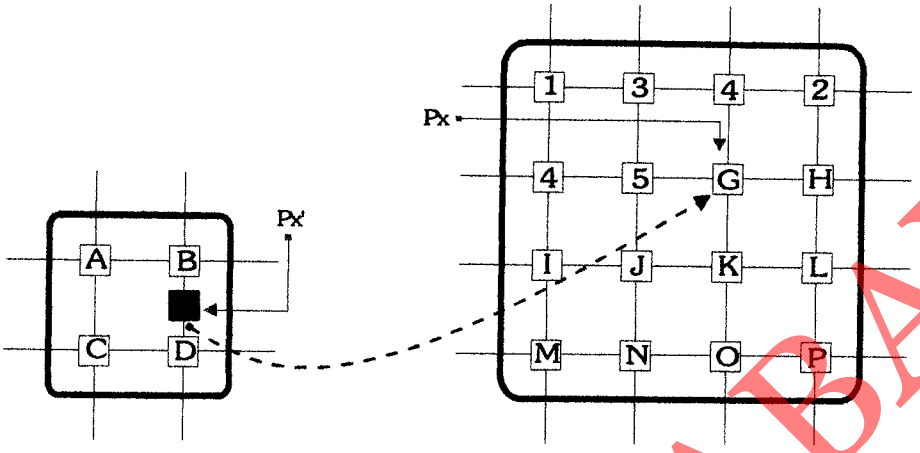
= 0,5 * (9 – 6) + 6

= 7,5

$F_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0,5 * (7,5 – 2,5) + 2,5

= 5




Gambar 2.16. Pemetaan untuk piksel $G_{tuj}$

**g. Piksel $G_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $G_{sum}$ = 2 – 2 = 0

Jarak Baris = Baris Px' – Baris $G_{sum}$ = 1,5 – 1 = 0,5

IAB = Jarak Kolom * (B –A) + A

= 0 * (0 – 4) + 4

= 4

ICD = Jarak Kolom * (D –C) + C

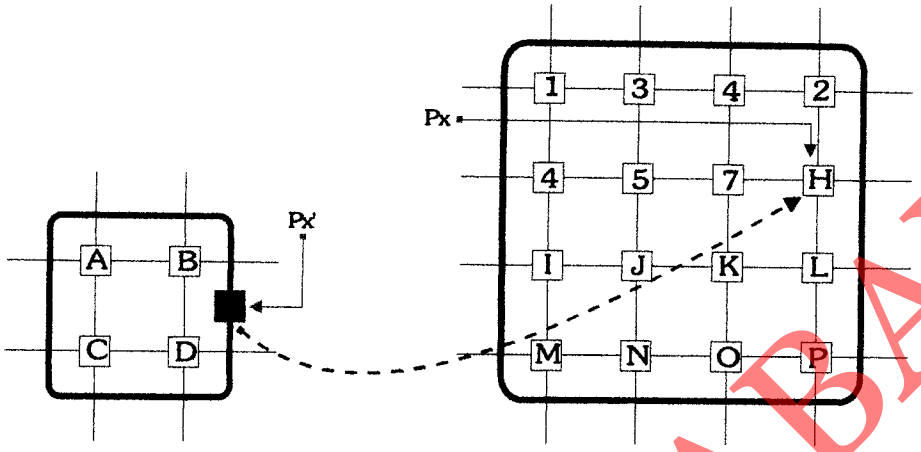
= 0 * (0 – 9) + 9

= 9

$G_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0,5 * (9 – 4) + 4

= 6,5 » 7




Gambar 2.17. Pemetaan untuk piksel $H_{tuj}$

**h. Piksel $H_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $H_{sum}$ = 2,5 – 2 = 0,5

Jarak Baris = Baris Px' – Baris $H_{sum}$ = 1,5 – 1 = 0,5

IAB = Jarak Kolom * (B –A) + A

= 0,5 * (0 – 4) + 4

= 2

ICD = Jarak Kolom * (D –C) + C

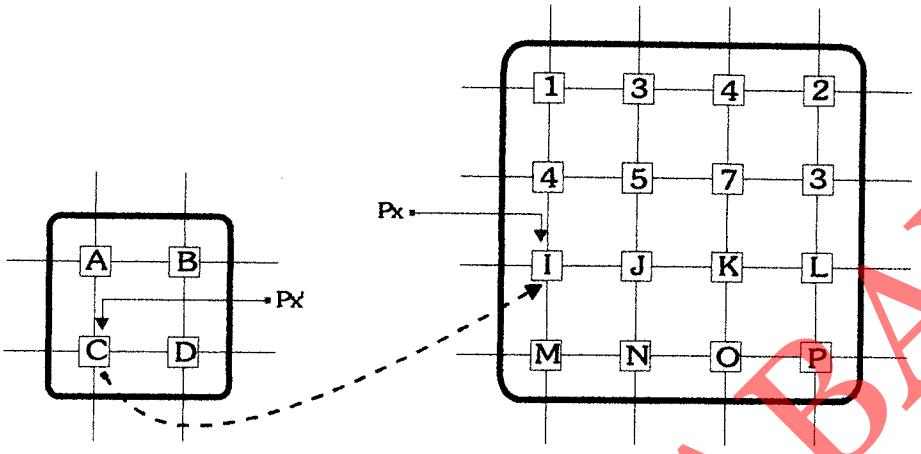
= 0,5 * (9 – 6) + 6

= 4,5

$H_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0,5 * (4,5 – 2) + 2

= 3,25 » 3




Gambar 2.18. Pemetaan untuk piksel $I_{tuj}$

**i. Piksel $I_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $I_{sum}$ = 1 – 1 = 0

Jarak Baris = Baris Px' – Baris $I_{sum}$ = 2 – 2 = 0

IAB = Jarak Kolom * (B –A) + A

= 0 * (0 – 6) + 6

= 6

ICD = Jarak Kolom * (D –C) + C

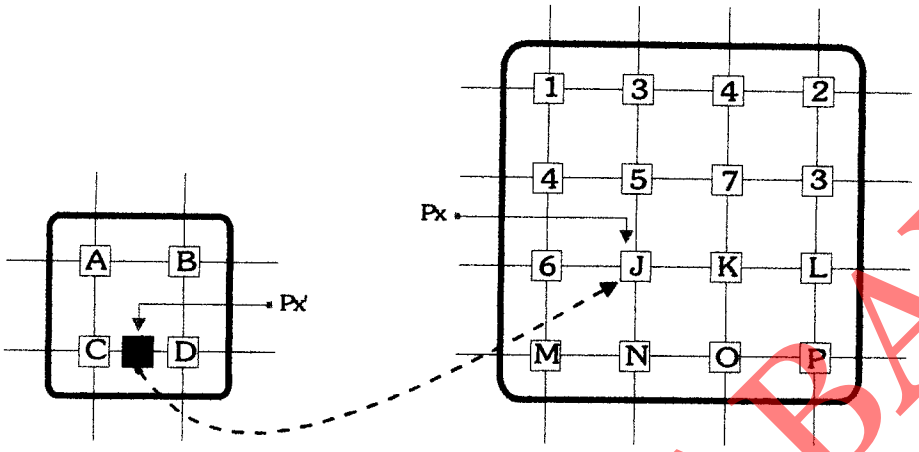
= 0 * (0 – 9) + 9

= 9

$I_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (9 – 6) + 6

= 6




Gambar 2.19. Pemetaan untuk piksel $J_{tuj}$

**j. Piksel $J_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $J_{sum}$ = 1,5 – 1 = 0,5

Jarak Baris = Baris Px' – Baris $J_{sum}$ = 2 – 2 = 0

IAB = Jarak Kolom * (B –A) + A

= 0,5 * (9 – 6) + 6

= 7,5

ICD = Jarak Kolom * (D –C) + C

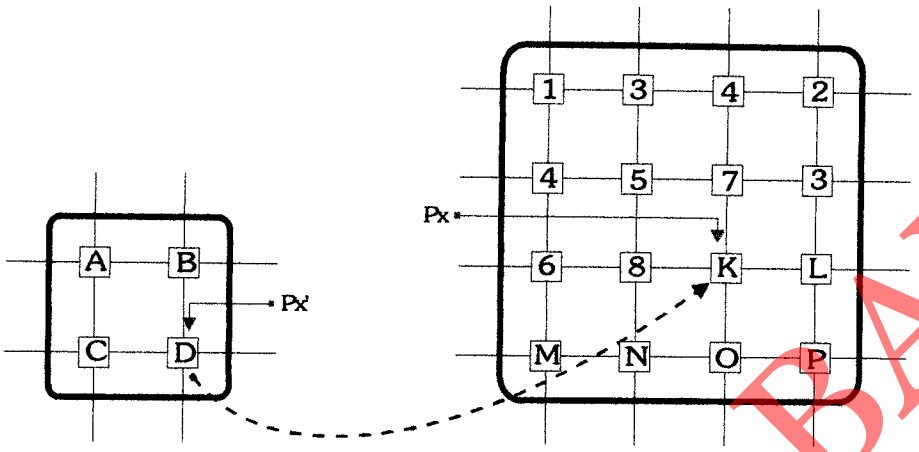
= 0,5 * (0 – 0) + 0

= 0

$J_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (0 – 7,5) + 7,5

= 7,5 » 8




Gambar 2.20. Pemetaan untuk piksel $K_{tuj}$

**k. Piksel $K_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $K_{sum}$ = 2 – 2 = 0

Jarak Baris = Baris Px' – Baris $K_{sum}$ = 2 – 2 = 0

IAB = Jarak Kolom * (B –A) + A

= 0 * (0 – 9) + 9

= 9

ICD = Jarak Kolom * (D –C) + C

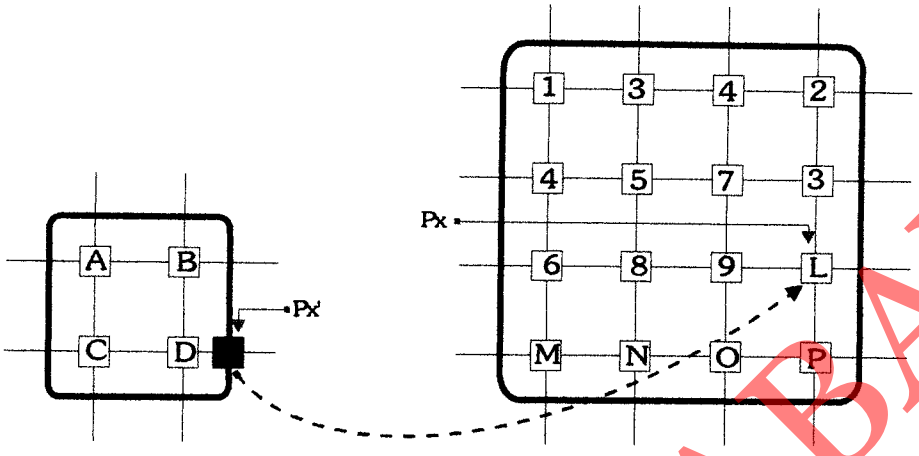
= 0 * (0 – 0) + 0

= 0

$K_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (0 – 9) + 9

= 9




Gambar 2.21. Pemetaan untuk piksel $L_{tuj}$

**1. Piksel $L_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $L_{sum}$ = 2,5 – 2 = 0,5

Jarak Baris = Baris Px' – Baris $L_{sum}$ = 2 – 2 = 0

IAB = Jarak Kolom * (B –A) + A

= 0,5 * (0 – 9) + 9

= 4,5

ICD = Jarak Kolom * (D –C) + C

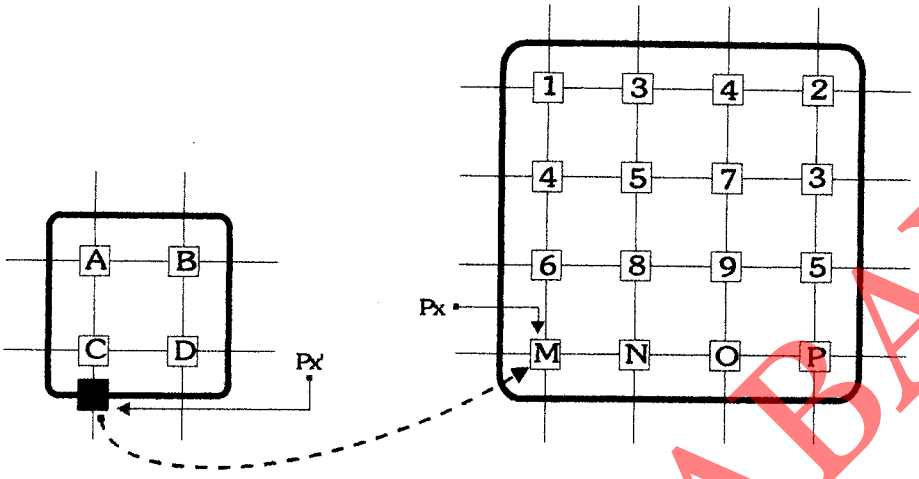
= 0,5 * (0 – 0) + 0

= 0

$L_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (0 – 4,5) + 4,5

= 4,5 » 5




Gambar 2.22. Pemetaan untuk piksel $M_{tuj}$

**m. Piksel $M_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $M_{sum}$ = 1 – 1 = 0

Jarak Baris = Baris Px' – Baris $M_{sum}$ = 2,5 – 2 = 0,5

IAB = Jarak Kolom * (B –A) + A

= 0 * (9 – 6) + 6

= 6

ICD = Jarak Kolom * (D –C) + C

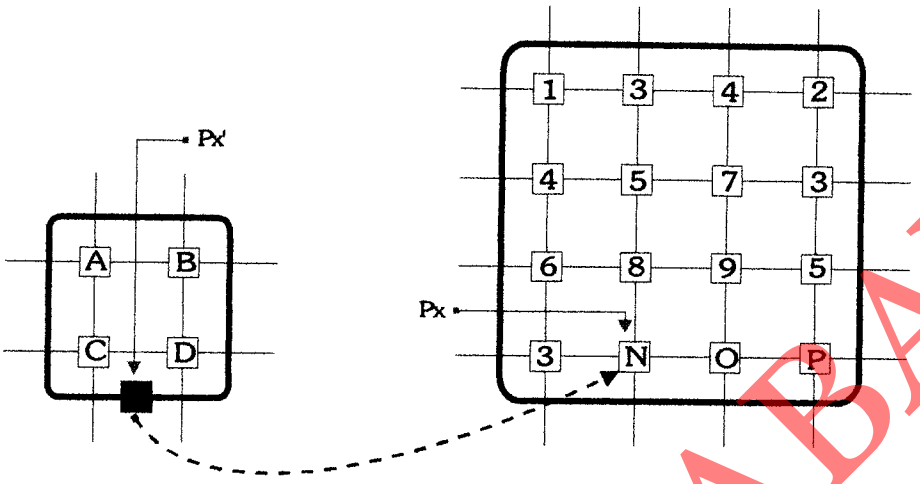
= 0 * (0 – 0) + 0

= 0

$M_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0,5 * (0 – 6) + 6

= 3




Gambar 2.23. Pemetaan untuk piksel $N_{tuj}$

**n. Piksel $N_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $N_{sum}$ = 1,5 – 1 = 0,5

Jarak Baris = Baris Px' – Baris $N_{sum}$ = 2,5 – 2 = 0,5

IAB = Jarak Kolom * (B –A) + A

= 0,5 * (9 – 6) + 6

= 7,5

ICD = Jarak Kolom * (D –C) + C

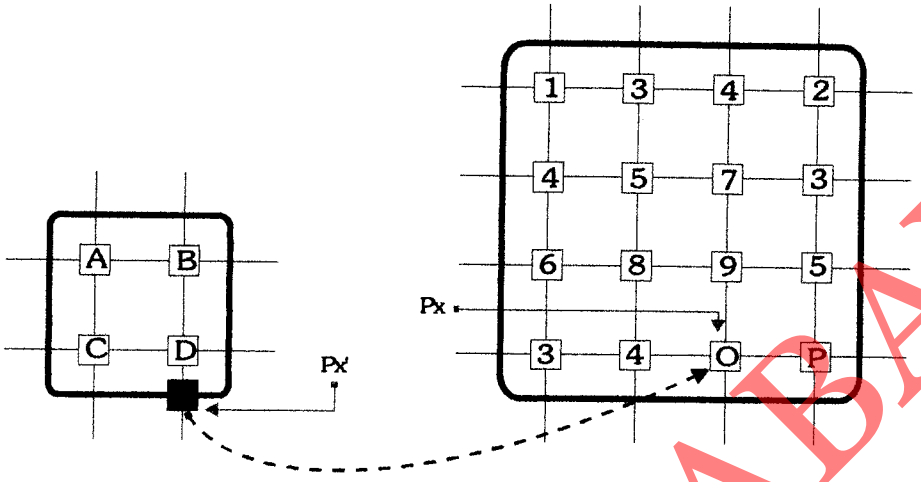
= 0,5 * (0 – 0) + 0

= 0

$N_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0,5 * (0 – 7,5) + 7,5

= 3,75 » 4




Gambar 2.24. Pemetaan untuk piksel $O_{tuj}$

**o. Piksel $O_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $O_{sum}$ = 2 – 2 = 0

Jarak Baris = Baris Px' – Baris $O_{sum}$ = 2,5 – 2 = 0,5

IAB = Jarak Kolom * (B –A) + A

= 0 * (0 – 9) + 9

= 9

ICD = Jarak Kolom * (D –C) + C

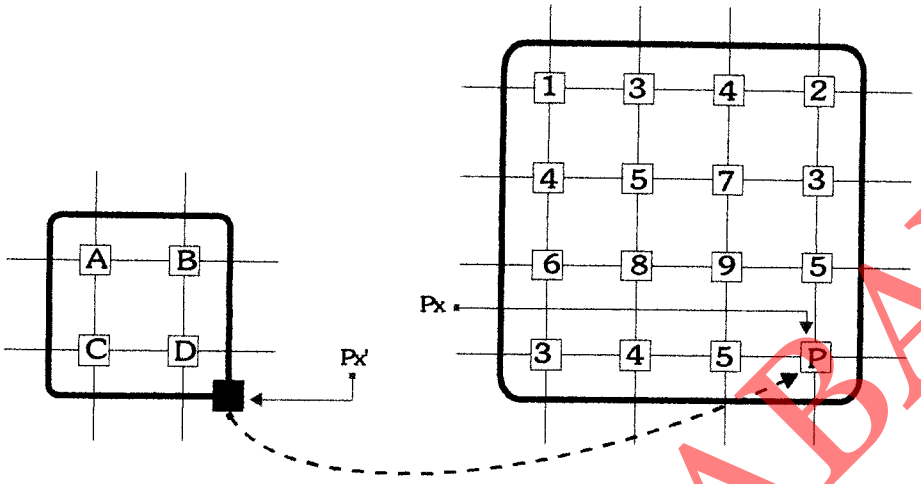
= 0 * (0 – 0) + 0

= 0

$O_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0,5 * (0 – 9) + 9

= 4,5 » 5




Gambar 2.25. Pemetaan untuk piksel $P_{tuj}$

**p. Piksel $P_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $P_{sum}$ = 2,5 – 2 = 0,5

Jarak Baris = Baris Px' – Baris $P_{sum}$ = 2,5 – 2 = 0,5

IAB = Jarak Kolom * (B –A) + A

= 0,5 * (0 – 9) + 9

= 4,5

ICD = Jarak Kolom * (D –C) + C

= 0,5 * (0 – 0) + 0

= 0

$P_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0,5 * (0 – 4,5) + 4,5

= 2,25 » 2

Sehingga citra tujuan menjadi seperti tampak pada gambar d bawah ini:

| | | | |
|---|---|---|---|
| 1 | 3 | 4 | 2 |
| 4 | 5 | 7 | 3 |
| 6 | 8 | 9 | 5 |
| 3 | 4 | 5 | 2 |

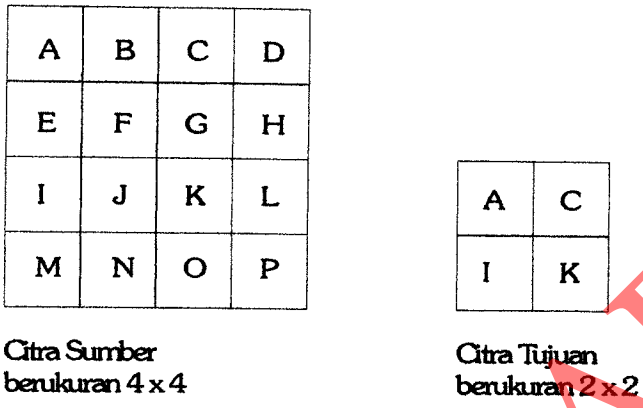
Citra Tujuan berukuran 4 x 4

Gambar 2.26. Citra ukuran 4 x 4 hasil proses perbesaran

### 2.3.2. Memperkecil citra

Proses memperkecil citra pada dasarnya adalah proses kebalikan dari proses memperbesar citra. Dalam hal ini jika pada proses memperbesar citra, piksel dari citra sumber dipetakan kepada banyak piksel di citra tujuan, maka dalam proses memperkecil, beberapa piksel dari citra sumber akan dipetakan kepada satu piksel pada citra tujuan. Metode yang digunakan dalam proses memperbesar citra dapat digunakan dalam proses memperkecil citra, tetapi dengan cara yang terbalik.

Untuk proses memperkecil citra dapat dilakukan dengan beberapa metode, diantaranya adalah dengan mengambil setiap piksel ke-*i* ( i = indeks citra tujuan) dibagi *n* (*n* = faktor skala) dari citra sumber yang diperkecil dengan faktor skala *n*. Kemudian piksel yang diambil dari citra sumber tersebut diletakkan pada citra tujuan. Metode ini merupakan metode replikasi pada proses memperbesar citra yang dilakukan dengan cara yang terbalik. Untuk selanjutnya metode yang dipakai dalam proses memperkecil citra ini akan disebut dengan metode replikasi.




Gambar 2.27. Memperkecil citra dengan metode replikasi

Misalkan sebuah citra dengan resolusi 4 x 4 akan diperkecil menggunakan metode replikasi dengan faktor 0,5. Dari proses memperkecil citra ini akan menghasilkan citra dengan resolusi 2 x 2. Maka citra tujuan akan memiliki indeks mulai dari 0 sampai dengan 1. Sehingga hasil dari proses memperkecil akan menjadi seperti pada gambar berikut ini:

Metode lain yang dapat digunakan untuk memperkecil citra adalah metode yang merupakan kebalikan dari metode interpolasi, yang untuk selanjutnya akan disebut dengan metode interpolasi saja. Untuk proses memperkecil citra dengan metode interpolasi ini rumusan interpolasi adalah sama seperti yang dipakai dalam proses memperbesar citra. Rumus untuk menghitung perubahan jarak dari piksel Px' yang dicari dengan piksel-piksel pendukung yang berada di sekitar piksel Px' yang dicari:

Jarak Kolom = Kolom Px' – Kolom A

Jarak Baris = Baris Px' – Baris A

Setelah perubahan jarak antara piksel-piksel tersebut diketahui, maka perlu dilakukan juga interpolasi secara dua dimensi. Sedangkan rumusan untuk interpolasi dua dimensi tersebut sama dengan rumusan perhitungan interpolasi pada proses memperbesar citra yaitu:

IAB = Jarak Kolom * (Nilai piksel B – Nilai piksel A) + Nilai piksel A

ICD = Jarak Kolom * (Nilai piksel D – Nilai piksel C) + Nilai piksel C

Px = Jarak Baris * (ICD - IAB) + IAB

IAB adalah hasil perhitungan interpolasi horizontal antara piksel pendukung A dan B, dan ICD adalah hasil perhitungan interpolasi horizontal antara piksel pendukung C dan D, sedangkan perhitungan interpolasi vertikal dilakukan antara hasil interpolasi horizontal A dan B dengan interpolasi horizontal C dan D. Interpolasi vertikal ini adalah interpolasi yang ketiga yang menghasilkan nilai piksel Px yang dicari yang diwakili oleh piksel Px'.

Untuk lebih jelasnya, berikut ini disajikan contoh proses perhitungan untuk proses memperkecil citra. Misalkan sebuah citra dengan resolusi 4x4 diperkecil dengan faktor skala 0,5 dengan menggunakan metode interpolasi.

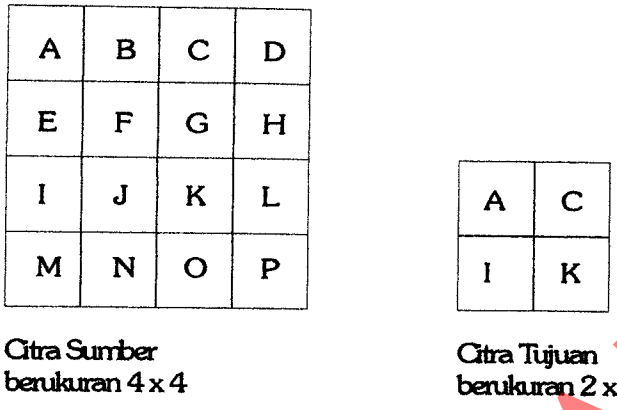
Maka dapat dihitung resolusi citra tujuan sebesar:

Jumlah Baris Citra Tujuan = faktor skala x resolusi vertikal

= 4 x 0,5 = 2

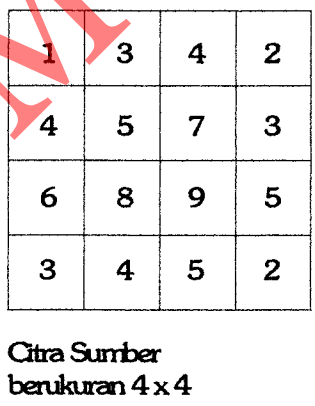
Jumlah Kolom Citra Tujuan = faktor skala x resolusi horizontal

= 4 x 0,5 = 2




Gambar 2.28. Memperkecil citra dengan metode interpolasi

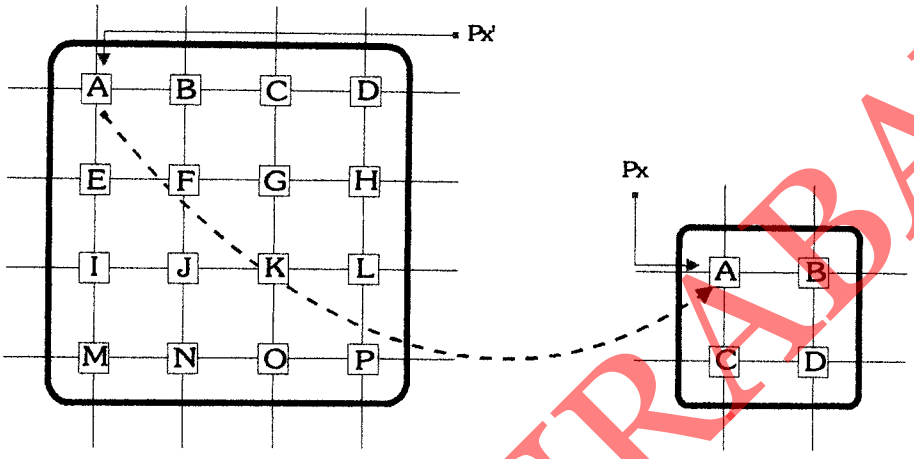
Sedangkan untuk mengisi nilai-nilai piksel pada citra tujuan ($A_{tuj}$ – $D_{tuj}$), dapat digunakan metode interpolasi dua dimensi dengan menggunakan rumusan seperti tersebut di atas. Misalkan sebuah citra sumber dengan ukuran 4 x 4 seperti pada gambar 2.29.




Gambar 2.29. Citra sumber berukuran 4 x 4

Dan perhitungan selengkapnya untuk mencari nilai setiap piksel pada citra sumber

adalah sebagai berikut.




Gambar 2.30. Pemetaan untuk piksel $A_{tuj}$

**a. Piksel $A_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $A_{sum}$ = 2 – 2 = 0

Jarak Baris = Baris Px' – Baris $A_{sum}$ = 2 – 2 = 0

IAB = Jarak Kolom * (B –A) + A

= 0 * (4 – 1) + 1

= 1

ICD = Jarak Kolom * (D –C) + C

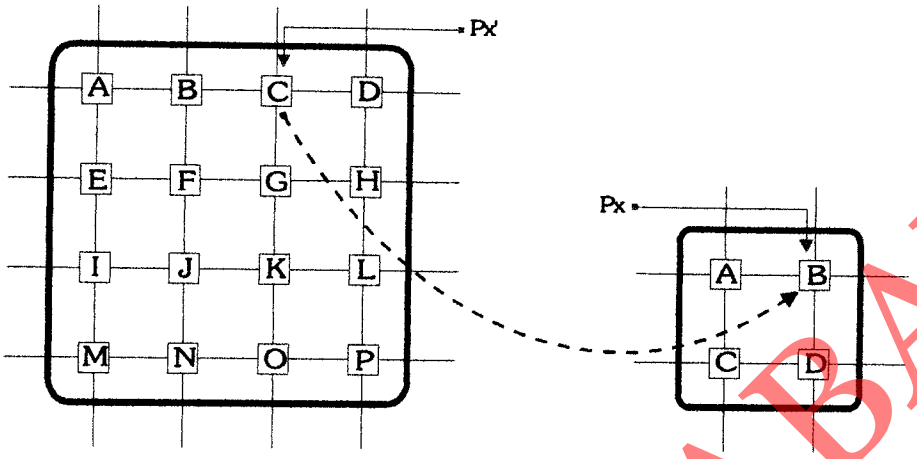
= 0 * (9 – 6) + 6

= 6

$A_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (6 – 1) + 1

= 1




Gambar 2.31. Pemetaan untuk piksel $B_{tuj}$

**b. Piksel $B_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $B_{sum}$ = 4 – 4 = 0

Jarak Baris = Baris Px' – Baris $B_{sum}$ = 2 – 2 = 0

IAB = Jarak Kolom * (B –A) + A

= 0 * (0 – 4) + 4

= 4

ICD = Jarak Kolom * (D –C) + C

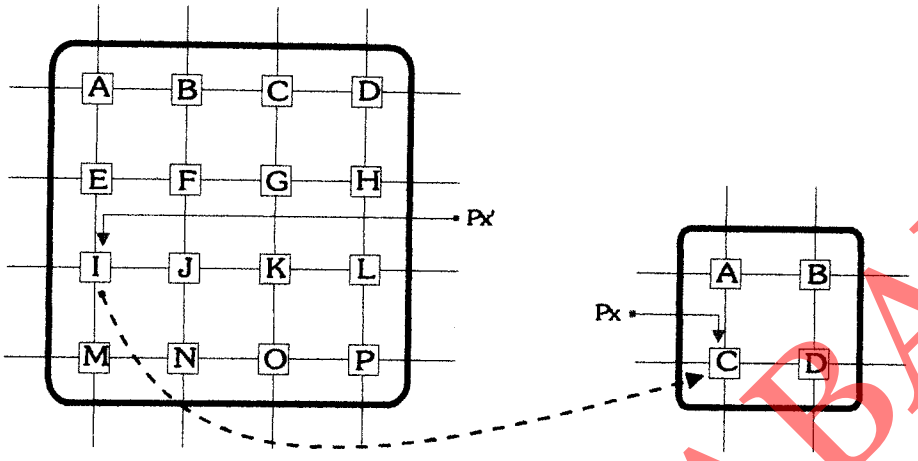
= 0 * (0 – 9) + 9

= 9

$B_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (9 – 4) + 4

= 4




Gambar 2.32. Pemetaan untuk piksel $C_{tuj}$

**c. Piksel $C_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $C_{sum}$ = 2 – 2 = 0

Jarak Baris = Baris Px' – Baris $C_{sum}$ = 4 – 4 = 0

IAB = Jarak Kolom * (B –A) + A

= 0 * (9 – 6) + 6

= 6

ICD = Jarak Kolom * (D –C) + C

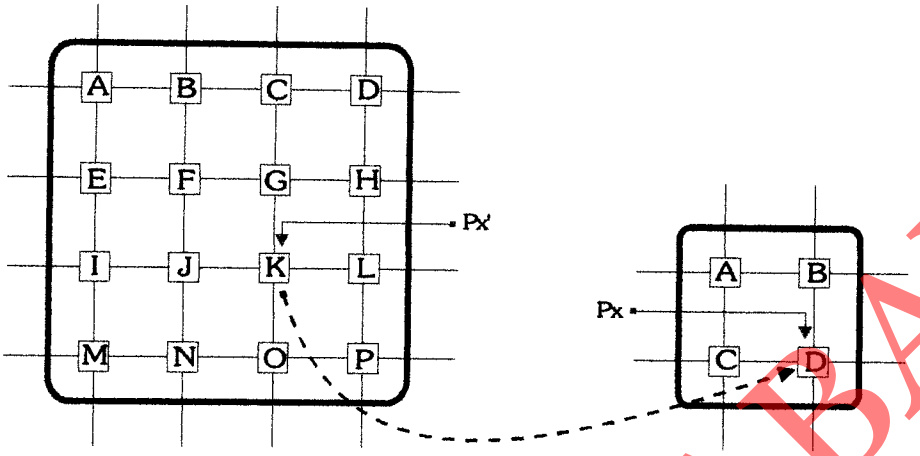
= 0 * (0 – 0) + 0

= 0

$C_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (0 – 6) + 6

= 6




Gambar 2.33. Pemetaan untuk piksel $D_{tuj}$

**d. Piksel $D_{tuj}$**

Jarak Kolom = Kolom Px' – Kolom $D_{sum}$ = 4 – 4 = 0

Jarak Baris = Baris Px' – Baris $D_{sum}$ = 4 – 4 = 0

IAB = Jarak Kolom * (B –A) + A

= 0 * (0 – 9) + 9

= 9

ICD = Jarak Kolom * (D –C) + C

= 0 * (0 – 0) + 0

= 0

$D_{tuj}$ = Jarak Baris * (ICD - IAB) + IAB

= 0 * (0 – 9) + 9

= 9

Sehingga citra tujuan hasil proses memperkecil citra akan menjadi seperti yang tampak pada gambar 2.34 berikut ini.

| | |
|---|---|
| 1 | 4 |
| 6 | 9 |

Gambar 2.34. Citra ukuran 2 x 2 hasil proses memperkecil