Peniruan Gaya *Pop Art* dengan Teknik Pengolahan Citra

Muhammad Rifat Abiwardani 13519205 (*Author*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: abi.wardani85@gmail.com

***Abstract*—Gaya seni pop art merupakan gaya dengan palet warna yang mencolok dan tingkat detail yang berada pada ambang minimalis dan realistis. Program ini akan menggunakan metode image processing untuk mengubah citra asal foto menjadi citra hasil seolah-olah merupakan ilustrasi pop art. Program terdiri dari tiga bagian: deteksi latar belakang, perubahan warna, dan *image enhancement*. Pendeteksian latar belakang menggunakan teknik pengolahan tepi berbasis konvolusi, perubahan warna menggunakan kuantisasi warna, dan perbaikan citra pada tahap pasca-pemrosesan menggunakan penapis luas serta eliminasi derau.**

Keywords—Pop art, konvolusi, deteksi tepi, kuantisasi, perbaikan citra

I. PENDAHULUAN

Sejak era 1950-an, gaya seni *Pop Art* menyebar di dunia Barat sebagai bidang seni yang identik dengan non-tradisionalisme. Gaya *Pop Art* digunakan secara luas pada media popular seperti komik, pengiklanan, dan *branding*. Awalnya berasal dari seniman-seniman Britania Raya dan Amerika, *Pop Art* kini dikenal di seluruh dunia sebagai salah satu gaya seni yang simbolik dengan kehidupan modern. Selain gayanya yang mencolok dan penuh unsur-unsur budaya popular, *Pop Art* juga dikenal sebagai gaya seni yang memanfaatkan teknologi dan mekanisasi untuk menghasilkan karya seninya. Hal tersebut dikarenakan *Pop Art* pada puncak eranya banyak digunakan pada produk komersial, sehingga media tersebut harus diproduksi secara massal dan dengan biaya yang murah. Teknologi percetakan dan digitalisasi mendorong keberadaan *Pop Art* dalam budaya seni, dan kini, teknologi media seni digital dan manipulasi citra menjadi rumah berikutnya untuk *Pop Art*.

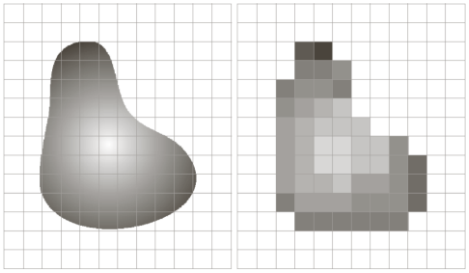
Gambar 1.1. "Michael Jackson" karya Andy Warhol
(Sumber: Smithsonian Magazine www.smithsonianmag.com/)

Era digital menunjang keberadaan *Pop Art* dalam berbagai cara. Dengan digitalisasi seni, kini seniman mampu menghasilkan karyanya tanpa menggunakan cat dan kertas, melainkan dapat langsung menggambar pada perangkat keras digital yang khusus didesain untuk seni, seperti tablet gambar dan pena digital. Bersama dengan aplikasi melukis, era digital telah membuka suatu media baru untuk berkarya. Namun selain itu, manipulasi gambar juga hadir sebagai bidang pengolahan citra berbasis komputer. Bidang pengolahan citra melibatkan mengubah suatu citra masukan menjadi citra baru, dengan berbagai hasil yang hanya dicapai menggunakan pemrosesan data dan angka. Bidang pengolahan citra memungkinkan pengguna untuk melakukan berbagai hal, seperti menghilangkan derau pada gambar, mempertajam detil, maupun mengubah tampilan objek pada gambar. Pada makalah ini, teknik pengolahan citra dimanfaatkan untuk menghasilkan sebuah gambar yang dapat meniru gaya *Pop Art* tersebut. Khususnya, makalah ini menyusun sebuah *pipeline* dalam MATLAB untuk mengubah citra input menjadi citra dalam gaya tiruan *Pop Art*.

II. LANDASAN TEORI

A. *Citra Digital*

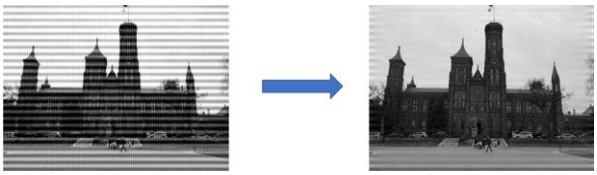
Citra digital merupakan representasi citra atau gambar dalam medium digital. Citra yang direpresentasi merupakan sekumpulan sinyal kontinu (umumnya berbasis cahaya) yang analog dengan stimuli yang ditangkap oleh sistem visual manusia. Citra kontinu tersebut direpresentasikan pada medium digital (yang bersifat diskrit) melalui proses *sampling* dan kuantisasi. Proses *sampling* dilakukan untuk membagi ruang kontinu proyeksi citra menjadi elemen-elemen ruang diskrit, yang juga dikenal sebagai *pixel*. Proses kuantisasi mengubah nilai intensitas input kontinu menjadi nilai-nilai intensitas (dan nilai-nilai warna untuk citra berwarna) yang diskrit. [1]



Gambar 2.1. Citra asli (kiri) dan citra digital (kanan) (Sumber: Dr. George Bebis, *Image Formation and Representation*)

B. *Pengolahan Citra*

Pengolahan citra merupakan kegiatan mengolah atau memproses citra sehingga memperoleh citra baru. Umumnya, pengolahan dilakukan untuk meningkatkan kualitas citra agar lebih sesuai dilihat oleh mata manusia. Proses ini dapat membantu manusia untuk lebih mudah menginterpretasi informasi yang dikandung dalam citra tersebut. Beberapa aspek citra yang umum diperbaiki adalah derau, kontras, ketajaman, kekaburan, dan distorsi. Pengolahan citra juga mencakup proses-proses yang memungkinkan analisis lanjut citra tersebut, seperti deteksi objek, pengelompokan elemen, maupun pengukuran. Berikut beberapa contoh kegunaan pengolahan citra. [1]



Gambar 2.2. Penghilangan derau (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2022-2023/01-Pengantar-Pengolahan-Citra-Bag1-2022.pdf)

Gambar 2.3. Gambar *blur* (kiri) dan hasil reduksi *blur* (kanan) (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2022-2023/01-Pengantar-Pengolahan-Citra-Bag1-2022.pdf)

C. *Konvolusi*

Salah satu metode pengolahan citra utama adalah penapisan, atau *filtering*. Penapisan merupakan operasi pada tingkat lokal citra yang mengubah nilai pixel dalam citra berdasarkan nilai-nilai pixel tetangganya. Dengan itu, perubahan yang diharapkan dicapai melalui pengolahan citra berkaitan kuat dengan properti atau karakteristik citra yang dapat diamati pada aras lokal, seperti adanya derau dan ketajaman tepi-tepi objek. Operator atau proses yang digunakan untuk melakukan penapisan adalah operator konvolusi. [2]

Secara formal, konvolusi antara dua fungsi f(x) dan g(x) didefinisikan sebagai berikut, dimana tanda * merupakan operator konvolusi. [2]

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(a)g(x-a)da$$

Eq. 2.1. Definisi operator konvolusi

Dalam konvolusi, f(x) dapat dianggap sebagai sinyal sumber, sementara g(x) adalah kernel atau mask konvolusi yang digunakan. Konsep dibalik konvolusi adalah mempertimbangkan setiap jendela pada sumbu input, menghitung nilai hasil evaluasi sinyal sumber terhadap kernel, kemudian menggeser jendela tersebut. [2]

Jika f(x) dan g(x) merupakan fungsi diskrit, dengan sumbu x juga diskrit, maka operator konvolusi diskrit dapat didefinisikan sebagai berikut. [2]

$$f(x) * g(x) = \sum_{a=-\infty}^{\infty} f(a)g(x-a)$$

Eq. 2.2. Definisi operator konvolusi diskrit

Karena dalam pengolahan citra, konvolusi dilakukan pada sinyal dua dimensi, maka konvolusi dilakukan pada kedua dimensi tersebut, dan kernel yang digunakan juga berada dalam dua dimensi. Maka untuk fungsi sinyal citra digital f(x, y) dan kernel konvolusi g(x, y), persamaan konvolusi dapat ditulis sebagai berikut. [2]

$$f(x,y) * g(x,y) = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} f(a,b)g(x-a,y-b)$$

Eq. 2.3. Definisi operator konvolusi dua dimensi diskrit

Dalam konteks pengolahan citra, kernel konvolusi umumnya merupakan matriks MxM, dimana M merupakan bilangan ganjil, dengan M=3 paling sering digunakan. Dengan menukar dimensi pada perhitungan konvolusi, operasi dapat dianggap sebagai perhitungan *dot product* pada setiap jendela MxM citra denagn kernel konvolusi, dan membangkitkan citra baru dari hasil *dot product* tersebut.

D. *Citra Biner*

Citra biner merupakan citra tak berwarna dengan dua derajat keabuan: 0 dan 1, atau hitam dan putih. Hitam dapat dinotasi dengan 0 dan putih dengan 1, ataupun sebaliknya, bergantung pada konvensi yang digunakan untuk definisi citra biner. Citra biner digunakan untuk menggambarkan fitur suatu citra. Fitur dapat berupa tepi-tepi objek, *mask* objek dan latar belakang, ataupun pola yang hadir dalam citra. Konversi dari citra *grayscale* menjadi citra biner menggunakan proses *thresholding* atau pengambangan. [3]

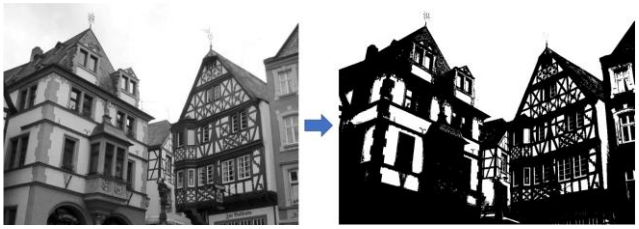
E. *Thresholding*

Proses *thresholding* atau pengambangan merupakan operasi untuk mengubah citra *grayscale* menjadi citra biner. Operasi *thresholding* mengelompokkan pixel dalam citra menjadi dua kelompok berdasarkan suatu nilai ambang. Semua pixel dalam citra dengan nilai kurang dari atau sama dengan suatu nilai ambang dijadikan satu kelompok, dan semua pixel dengan nilai lebih dari nilai ambang dijadikan satu kelompok. Tergantung konvensi, kelompok pixel dengan nilai kurang dari sama dengan nilai ambang dapat ditukar dengan 0, dan kelompok pixel dengan nilai lebih dari nilai ambang ditukar dengan 1, atau sebaliknya. [3]

Penentuan nilai ambang dapat dilakukan secara global, secara lokal, maupun secara adaptif. Untuk pengambangan secara global, nilai ambang ditentukan sama untuk setiap pixel. Umumnya, nilai ini merupakan nilai yang paling bagus membagi citra dengan histogram bimodal; yakni, intensitas cahaya pixel-pixel dalam citra *grayscale* terkelompok sekitar dua pusat. Untuk pengambangan secara lokal, nilai ambang ditentukan berdasarkan nilai tetangga suatu pixel. Sebagai contoh, untuk jendela 5x5, nilai pixel tengah dapat dijadikan 0 apabila nilainya kurang dari atau sama dengan rata-rata pixel dalam jendela 5x5 tersebut. Secara umum, penentuan kelompok dapat ditentukan dengan membandingkan nilai pixel tengah dengan hasil perhitungan konvolusi suatu kernel dengan matriks tetangga pixel tersebut. Untuk pengambangan secara adaptif, nilai ambang dapat diubah secara dinamis berdasarkan karakteristik lokal pixel, ataupun aturan lain. Dengan itu, pengambangan dapat dilakukan dengan mempertimbangkan kecacatan citra seperti kondisi cahaya yang tidak rata. [3]

F. *Metode Otsu*

Metode Otsu digunakan untuk mencari nilai optimum untuk melakukan pengambangan secara global. Metode Otsu berusaha memaksimalkan variansi antar kelas, sehingga *thresholding* membagi pixel menjadi dua kelompok yang berbeda. Metode ini pertama mencatat distribusi frekuensi setiap tingkat *graylevel* dalam citra, misalkan terdapat L tingkat, dan menghitung proporsi setiap tingkat. Kemudian diambil suatu nilai k antara 0 hingga $L-1$, dan ditinjau kesesuaian nilai k untuk menjadi nilai ambang global. Pertama dihitung proporsi pixel dengan tingkat keabuan kurang dari sama dengan k sebagai P_1, dan dihitung proporsi pixel dengan tingkat keabuan lebih dari k sebagai P_2. Setelah itu dihitung rata-rata tingkat keabuan dalam masing kelompok sebagai m_1 dan m_2, dan diperoleh tingkat keabuan rata-rata global sebagai $m_g = P_1 m_1 + P_2 m_2$. Menggunakan m_g, dihitung variansi global untuk nilai ambang k sebagai ${\sigma_g}^2 = \sum_{i=0}^{L-1}(i - m_g)^2 \cdot p_i$. Juga menggunakan m_g, dihitung variansi antar kelas untuk nilai ambang k sebagai ${\sigma_b}^2 = \frac{(m_g P_1 - m)^2}{P_1(1-P_1)}$. Untuk mengevaluasi kesesuaian k sebagai nilai ambang, dilihat nilai $\frac{{\sigma_b}^2}{{\sigma_g}^2}$ terbesar. [4]



Gambar 2.4. Contoh hasil pengambangan dengan Metode Otsu (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2021-2022/22-Segmentasi-Citra-Bagian1-2022.pdf)

G. *Pendeteksian Tepi*

Dalam citra digital, tepi didefinisikan sebagai peningkatan atau penurunan nilai keabuan yang tajam. Dalam kata lain, tepi merupakan perubahan nilai keabuan yang besar dalam jarak spasial yang pendek. Karena citra digital berada dalam dua dimensi, maka dapat ditetapkan konvensi arah bagi definisi tepi, yakni sebuah tepi memiliki arah sesuai dengan arah perubahan nilai keabuan. [5]

Pendeteksian tepi merupakan sekumpulan metode untuk memperoleh informasi tepi dari suatu citra digital. Umumnya, informasi ini diperoleh melalui operasi aras lokal, dimana pixel tepi dipahami sebagai pixel yang memiliki perubahan nilai tinggi di sekitarnya. Pendeteksian tepi umumnya dapat dibagi menjadi dua bagian: evaluasi kondisi tepi dan pengambangan. Pertama, suatu nilai dihitung yang dapat mengindikasi apakah suatu pixel merupakan pixel tepi atau bukan. Nilai tersebut dapat dihitung menggunakan berbagai kernel deteksi tepi seperti Sobel, Prewitt, atau Roberts. Selain menggunakan kernel konvolusi, penilaian tepi dapat dilakukan dengan menghitung *simple standard deviation* dari pixel-pixel tetangga suatu pixel. Kemudian, tahap pengambangan menentukan apakah pixel tersebut dianggap pixel tepi atau tidak, berdasarkan hasil perhitungan penilaian tepi. Pengambangan dapat menggunakan metode pengambangan global, maupun secara adaptif. [5]

Gambar 2.5. Citra *grayscale* asli (kiri) dan hasil pendeteksian tepi (kanan) (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2021-2022/18-Pendeteksian-Tepi-Bagian1-2022.pdf)

H. *Kuantisasi*

Dalam konversi citra digital, kuantisasi dipahami sebagai metode diskritisasi untuk mengubah nilai intensitas cahaya kontinu menjadi nilai diskrit. Dalam kuantisasi, umumnya didefinisikan terlebih dahulu jumlah tingkat intensitas target, untuk kemudian memuatkan nilai-nilai kontinu dalam setiap rentang diskrit. Kuantisasi dapat dilakukan secara uniform, sehingga berusaha mempertahankan distribusi intensitas cahaya

citra asli. Sebagai konvensi, umumnya citra digital menggunakan jumlah tingkat intensitas yang merupakan perpangkatan dari 2 seperti 32, 64, 256, dan sebagainya. Semakin tinggi jumlah tingkat intensitas, semakin baik citra digital dalam mempertahankan tampak citra asli. Dalam pemetaan uniform, citra dengan rentang nilai awal $[0, M]$ dibagi menjadi K buah rentang yang berjarak sama, dan setiap sub-rentang dalam $[0, M]$ dipetakan ke suatu nilai antara 0 dan $K - 1$. Dengan itu, setiap sub-rentang memiliki jarak M/K. [1]

Selain pemetaan uniform, penentuan nilai hasil pemetaan dapat ditentukan melalui klustering. Klustering merupakan metode untuk membagi sekumpulan nilai menjadi kelas-kelas yang meminimalisasi suatu fungsi objektif, seperti variansi antar kelas dan variansi dalam kelas. Salah satu metode klustering adalah *k-means clustering*, yang mengelompokkan nilai-nilai data menjadi kluster-kluster yang meminimalisasi jarak dari rata-rata setiap kluster. Metode *k-means clustering* diawali dengan pembagian *k* kluster awal secara acak, kemudian menghitung nilai rata-rata setiap kluster. Untuk setiap nilai dalam data, dihitung jarak titik data tersebut ke nilai rata-rata kluster. Untuk nilai intensitas keabuan, jarak tersebut dapat dihitung sebagai akar dari selisih nilai titik data dan nilai rata-rata kluster. Kemudian seluruh titik data dikelompokkan ulang berdasarkan jarak terdekat ke rata-rata kluster, dan proses iterasi diulang hingga pengelompokkan berhenti berubah (kondisi konvergen). [1]

III. METODE

Pipeline pemrosesan citra awal menjadi citra akhir terdiri dari tiga tahap. Pertama, dilakukan deteksi latar belakang untuk memisahkan objek dengan sekitarannya. Kedua, dilakukan perubahan skema warna untuk mengubah citra foto menjadi gaya target *Pop Art*. Ketiga, dilakukan pasca-pemrosesan berupa perbaikan citra untuk merapihkan citra akhir agar lebih menyerupai gaya *Pop Art*.

A. Pendeteksian Latar Belakang

Pemisahan objek dan latar belakang dilakukan menggunakan gabungan teknik konvolusi, pendeteksian tepi, pengambangan, serta penapisan luas. Tahap ini menerima citra digital berwarna, dan mengembalikan dua mask biner yang mengindikasikan objek dan latar belakang. Untuk contoh kasus, digunakan citra berikut, yang memiliki latar belakang yang mudah dipisah karena warnanya yang seragam dan berbeda dari objek utama.

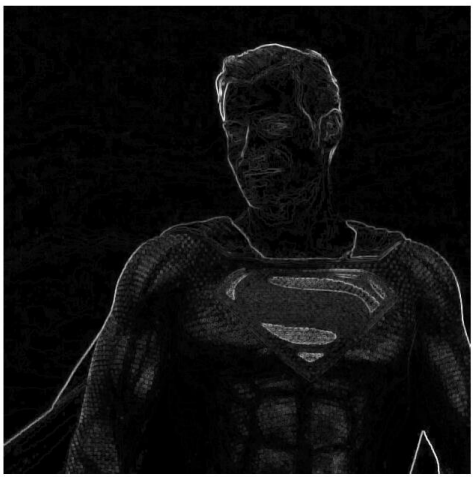
Gambar 3.1. Citra uji *Superman*

Pertama, dilakukan pendeteksian tepi menggunakan konvolusi. Untuk mencapai efek *fuzzy*, tidak digunakan kernel deteksi umum seperti Sobel maupun Prewitt, melainkan menghitung nilai standar deviasi setiap jendela pixel. Dalam pengembangan *pipeline*, digunakan ukuran jendela 3x3. Berikut kode program MATLAB untuk melakukan konvolusi tersebut.

```
function scores = BgEval(img, window)
    dim = size(size(img));

    if (dim(2) == 3)
        img = rgb2gray(img);
    end

    [Y, X] = size(img);
    res = img;
    scores = zeros(Y, X);

    for i = 1:Y
        top = max(1, i-window);
        bottom = min(Y, i+window);
        for j = 1:X
            left = max(1, j-window);
            right = min(X, j+window);

            mat = zeros(bottom-top+1, right-left+1);

            for p = top:bottom
                for q = left:right
                    mat(p-top+1, q-left+1) = img(p, q);
                end
            end

            score = calcStd(mat);
            scores(i, j) = score;
        end
    end

    scores = scores/max(max(scores));
end

function score = calcStd(mat)
```

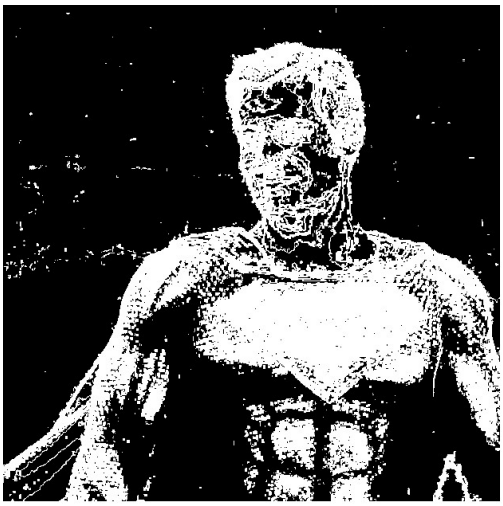
```
        arr = reshape(mat.', 1, []);
        score = std(arr);
end
```

Berikut hasil fungsi BgEval pada citra *Superman*.



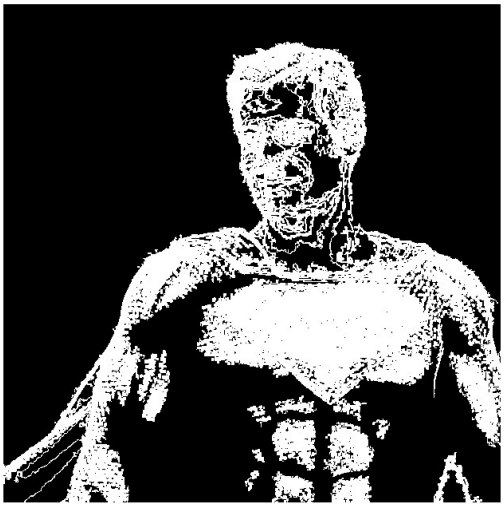
Gambar 3.2. Pendeteksian tepi citra uji *Superman*

Kemudian dilakukan pengambangan secara global pada hasil konvolusi, menggunakan nilai ambang 0.05. Nilai 0.05 dipilih melalui percobaan demi menghapus semua titik pada daerah tanpa banyak perubahan pada tingkat lokal. Berikut hasil pengambangan tersebut.



Gambar 3.3. Pengambangan mask tepi citra uji *Superman*

Pada gambar hasil pengambangan, dapat dilihat bahwa tetap terdapat titik-titik terisolasi dimana program mendeteksi pixel sebagai pixel tepi. Demi memperoleh mask latar belakang yang bersih, titik-titik tersebut harus dihilangkan. Metode yang digunakan untuk menghapus titik-titik tersebut adalah penapisan luas. Penapisan luas mendeteksi setiap komponen terhubung dalam citra biner, kemudian berdasarkan suatu nilai ambang luas, memilih untuk menghapus atau menyimpan komponen terhubung tersebut. Proses ini dilakukan menggunakan fungsi *bwareaopen* bawaan modul Image Processing MATLAB, dan nilai ambang yang digunakan adalah 100. Berikut hasil penapisan luas pada mask tepi.

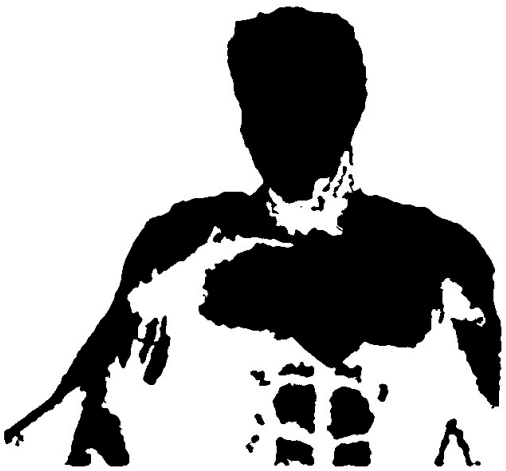


Gambar 3.4. Penapisan luas tepi citra uji *Superman*

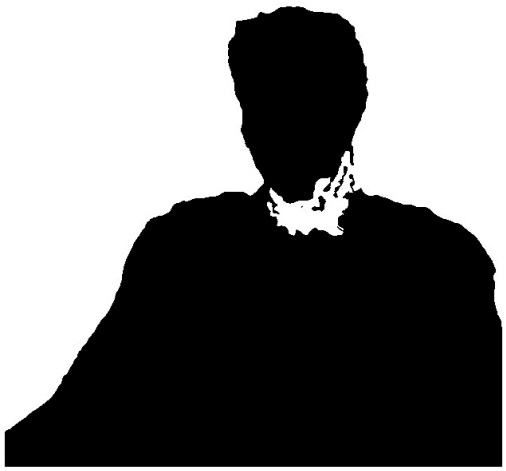
Setelah menghilangkan komponen-komponen terhubung dengan luas kurang dari 100, mask tepi dirapihkan lebih lanjut menggunakan *image smoothing*. Kernel *smoothing* yang digunakan adalah kernel uniform, dan dilakukan dalam dua tahap. Pertama, dilakukan *smoothing* dengan kernel kecil, kemudian dilakukan *smoothing* dengan kernel besar. Proses *smoothing* dengan kernel besar diaugmentasi proses pemeriksaan konsistensi informasi tepi. Yakni, dipastikan bahwa konvolusi tidak menyebabkan garis tepi untuk menjadi putus. Pemeriksaan tersebut dilakukan menggunakan fungsi *bwconncomp* bawaan modul Image Processing MATLAB, yang menghitung dan merangkum informasi komponen terhubung dalam suatu citra. Untuk memastikan bahwa tidak ada garis yang putus, dibandingkan setiap jendela MxM antara mask tepi sebelum dan setelah *smoothing* dengan kernel besar, dan jika jumlah komponen bertambah, maka terdapat komponen yang dipisah menjadi dua atau lebih bagian dari mask tepi awal. Jendela yang memenuhi kriteria tersebut dikembalikan ke mask tepi awal, sementara jendela yang lolos disimpan perubahannya. Berikut hasil *smoothing* dan *filtering* tersebut.

Gambar 3.5. *Smoothing* dan *filtering* mask tepi citra uji *Superman*

Setelah merapihkan informasi tepi, perlu diekstraksi informasi objek dan latar belakang. Didefinisikan bahwa untuk citra yang akan dikonversi menjadi gaya *Pop Art*, objek adalah bagian citra dengan aktivitas tinggi, atau bagian penuh pixel tepi. Sebaliknya, latar belakang adalah bagian citra dengan aktivitas rendah, atau bagian tanpa pixel tepi. Maka dari itu, perlu diteliti setiap komponen terhubung yang mungkin merupakan latar belakang, dan dipilih komponen yang berisi pixel non-tepi paling banyak. Selain itu, didefinisikan bahwa letak objek dalam citra akan cenderung dekat dengan pusat citra, maka latar belakang akan cenderung berada jauh dari pusat. Dengan dua kriteria tersebut, dilakukan pemilihan daerah latar belakang dengan bantuan fungsi *bwlabel* bawaan modul Image Processing MATLAB.



Gambar 3.6. Komponen-komponen calon latar belakang (dalam putih) dari citra uji *Superman*



Gambar 3.7. Mask calon latar belakang (dalam putih) dan mask calon objek (dalam hitam) dari citra uji *Superman*

Setelah mendapatkan mask calon objek dan latar belakang, maka dapat digunakan fungsi *grabcut* untuk membagi kedua daerah dan menelusuri batasnya. Fungsi *grabcut* digunakan bersama fungsi *superpixel* dan *poly2mask* untuk mendefinisikan daerah-daerah kasar pada citra. Berikut hasil *grabcut* dengan mask calon latar belakang.

Gambar 3.8. Mask latar belakang dari citra uji *Superman*

Berikut hasil *masking* pada citra asli.

Gambar 3.9. Citra objek (kiri) dan citra latar belakang (kanan) dari citra uji *Superman*

Setelah memisahkan kedua daerah, dapat dilakukan kuantisasi warna. Kuantisasi warna dilakukan untuk mendapatkan palet terbatas yang karakteristik pada karya *Pop Art*. Pada citra objek, kuantisasi dilakukan dengan target 16 warna, sementara pada latar belakang, kuantisasi dilakukan dengan target 2 warna (warna latar belakang dan warna hitam). Kuantisasi dilakukan dengan fungsi *rgb2ind* bawaan modul Image Processing MATLAB. Berikut hasil kuantisasi warna pada masing bagian citra.

Gambar 3.10. Hasil kuantisasi citra objek (kiri) dan citra latar belakang (kanan) dari citra uji *Superman*

Meskipun sudah diperoleh citra hasil kuantisasi terpisah, jika dilihat lebih dekat, terdapat daerah-daerah pada citra objek

dimana hasil kuantisasi menghasilkan pixel-pixel yang berderau. Pixel derau dipahami sebagai pixel yang memiliki nilai berbeda dari setiap tetangganya, dan setiap tetangganya memiliki nilai yang sama dengan yang lainnya. Berikut contoh daerah berderau tersebut.

Gambar 3.11. Contoh derau pada hasil kuantisasi citra objek dari citra uji *Superman*

Agar hasil metode penghilangan derau masih berada dalam skema warna hasil kuantisasi, maka pendeteksian pixel derau dan perbaikannya harus dilakukan berdasarkan informasi warna yang ada. Jika satu pixel memiliki nilai berbeda dengan setiap pixel sekitarnya, dan setiap pixel sekitarnya memiliki nilai yang saling sama, maka nilai pixel tengah tersebut ditukar dengan nilai pixel tetangganya. Proses tersebut dilakukan dengan ukuran jendela 3x3 dengan fungsi *RemoveIslands* berikut.

```
function res = RemoveIslands(lmap, window)
    [Y, X] = size(lmap);
    res = lmap;

    for i = 1:Y
        for j = 1:X
            i0 = max(1, i-window);
            i1 = min(Y, i+window);
            j0 = max(1, j-window);
            j1 = min(X, j+window);

            mat = lmap(i0:i1, j0:j1);

            vals = transpose(unique(mat));
            [~, n] = size(vals);
            if n == 2
                hist = histcounts(mat, n);
                if hist(1) == 1
                    if lmap(i, j) == vals(1)
                        res(i, j) = vals(2);
                    end
                elseif hist(2) == 1
                    if lmap(i, j) == vals(2)
                        res(i, j) = vals(1);
                    end
                end
            end
        end
    end
end
```

Berikut citra gabungan setelah dilalui fungsi *RemoveIslands* tersebut.

Gambar 3.12. Hasil penghilangan derau dari hasil kuantisasi citra uji *Superman*

Program keseluruhan *pipeline* dirangkum dalam potongan kode berikut.

```
function res = Im2VectorStyle(img)
    [ind, map] = rgb2ind(img, 64, "nodither");

    scores = BgEval(ind2rgb(ind, map), 1);

    b_scores = (scores >= 0.05);
    filtered = bwareaopen(b_scores, 100);

    windowSize1 = 3;
    kernelSmall = ones(windowSize1) / windowSize1 ^ 2;
    blurred_mask = conv2(single(filtered), kernelSmall, 'same');
```

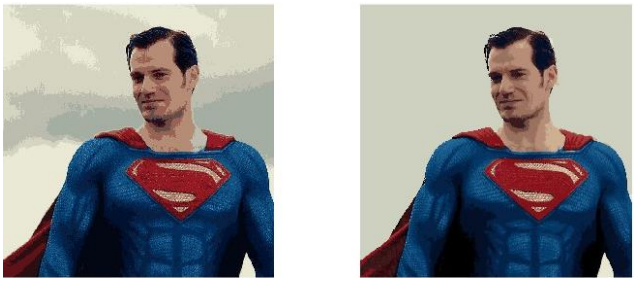
```
        smooth_mask = blurred_mask > 0.5; % Rethreshold
        res_mask = smooth_mask;

        for k = 1:1
            [Y, X] = size(res_mask);

            windowSize2 = 7;
            kernelBig = ones(windowSize2) / windowSize2 ^ 2;

            blurred_mask = conv2(single(res_mask), kernelBig, 'same');
            temp_mask = blurred_mask > 0.5;
            res_mask = temp_mask;
            [Y1, X1] = size(temp_mask);
            res2_mask = zeros(Y1, X1);

            for i = 1:Y-windowSize2-1
                for j = 1:X-windowSize2-1
                    i1 = min(i+windowSize2+1, Y);
                    j1 = min(j+windowSize2+1, X);
                    CC1 = bwconncomp(smooth_mask(i:i1, j:j1) + temp_mask(i:i1, j:j1));
                    CC2 = bwconncomp(temp_mask(i:i1, j:j1));

                    if (CC2.NumObjects > CC1.NumObjects)
                        res_mask(i:i1, j:j1) = (res_mask(i:i1, j:j1) + smooth_mask(i:i1, j:j1)) > 0;
                        res2_mask(i:i1, j:j1) = smooth_mask(i:i1, j:j1);
                    end
                end
            end
        end

        filled_mask = imfill(res_mask, "holes");

        bg_heuristic_mask = 1-filled_mask;
```

```
        [labels, n] = bwlabel(bg_heuristic_mask);
        [Y, X] = size(bg_heuristic_mask);
        Ym = Y/2;
        Xm = X/2;

        dists = zeros(1, n);

        for i = 1:n
            [rows, cols] = find(labels==i);
            coords = [rows cols];
            [l, ~] = size(coords);
            sum = 0;

            for j = 1:l
                sum = sum + (coords(j, 1)-Ym)^2+(coords(j, 2)-Xm)^2;
            end
            sum = sum/sqrt(l);
            dists(i) = sum;
        end

        [~, sort_idx] = sort(dists, 'descend');
        mask_copy = zeros(Y, X);
        mask_copy(labels==sort_idx(1)) = 1;

        gaussfilt = fspecial('gaussian', [7 7], 0.5);
        smooth_img = imfilter(img, gaussfilt, 'same');

        L = superpixels(smooth_img, 1000);
        h1 = drawpolygon('Position', [0, 0; Y+1, 0; Y+1, X+1; 0, X+1]);
        roiPoints = h1.Position;
        roi = poly2mask(roiPoints(:,1),roiPoints(:,2),size(L,1),size(L,2));
        cut_mask = grabcut(smooth_img, L, roi, smooth_mask, mask_copy);
        foreground_img = img;
        foreground_img(repmat(~cut_mask,[1 1 3])) = 0;
        background_img = img;
        background_img(repmat(cut_mask,[1 1 3])) = 0;
```

```
        [fg_res,            fg_map]            =
rgb2ind(foreground_img, 16, "nodither");
        [bg_res,            bg_map]            =
rgb2ind(background_img, 2, "nodither");


        fg_res = RemoveIslands(fg_res, 1);


        fg_ind = ind2rgb(fg_res, fg_map);
        bg_ind = ind2rgb(bg_res, bg_map);
        res= bg_ind+fg_ind;
end
```

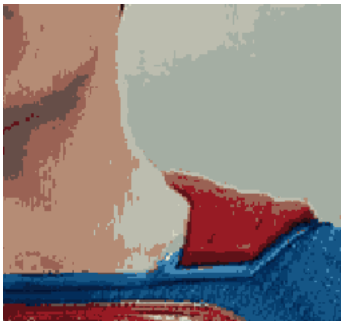
IV. Analisis

Berikut perbandingan antara hasil *pipeline* dengan jika hanya kuantisasi dilakukan pada keseluruhan citra.



Gambar 4.1. Hasil dengan hanya kuantisasi (kiri) and hasil *pipeline* (kanan)

Dari Gambar 4.1., dilihat bahwa tanpa pemisahan latar belakang, warna dari pixel objek dapat bercampur dengan pixel latar belakang, sebagaimana terlihat pada leher *Superman*. Berikut potongan *close-up* dari fenomena tersebut.



Gambar 4.2. *Close-up* leher *Superman*

Selain itu, tanpa proses *smoothing*, terdapat banyak pixel-pixel derau pada citra hasil kuantisasi saja. Derau yang tinggi tidak sesuai dengan gaya *Pop Art*, dan hal tersebut tidak diinginkan dalam hasil *pipeline*. Kehadiran derau tersebut dapat dilihat pula dalam Gambar 4.2., dimana daerah satu warna berpindah ke warna lain cenderung berderau. Bandingkan dengan daerah serupa pada Gambar 4.3., yang transisi warnanya jauh lebih rapih dan tidak ada pixel objek yang bercampur dengan pixel latar belakang.

Gambar 4.3. *Close-up* perbandingan leher *Superman*

V. Kesimpulan

Dengan *pipeline* yang dikembangkan dalam makalah ini, disimpulkan bahwa konversi citra menjadi gaya *Pop Art* berhasil dilakukan. Konversi tersebut dicapai melalui tiga proses: pendeteksian latar belakang, kuantisasi warna, dan perbaikan citra. Terlihat pula bahwa hasil *pipeline* lebih bagus secara kualitatif dibandingkan jika hanya kuantisasi warna yang dilakukan untuk meniru gaya *Pop Art*.

References

[1] Munir, Rinaldi. 2022. Pembentukan Citra dan Digitalisasi Citra. https://informatika.stei.itb.ac.id/~rinaldi.munir/

[2] Munir, Rinaldi. 2022. Konvolusi. https://informatika.stei.itb.ac.id/~rinaldi.munir/

[3] Munir, Rinaldi. 2022. Citra Biner. https://informatika.stei.itb.ac.id/~rinaldi.munir/

[4] Munir, Rinaldi. 2022. Segmentasi Citra Bagian 1. https://informatika.stei.itb.ac.id/~rinaldi.munir/

[5] Munir, Rinaldi. 2022. Pendeteksian Tepi (edge detection) Bagian 1. https://informatika.stei.itb.ac.id/~rinaldi.munir/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Desember 2022

Muhammad Rifat Abiwardani 13519205