BAB 3 ANALISIS DAN PERANCANGAN SISTEM

Bab ini memaparkan analisis masalah yang diatasi berserta pendekatan dan alur kerja dari perangkat lunak yang dikembangkan, mengimplementasikan metode yang digunakan dan hasil yang akan ditampilkan.

3.1 Analisis Masalah

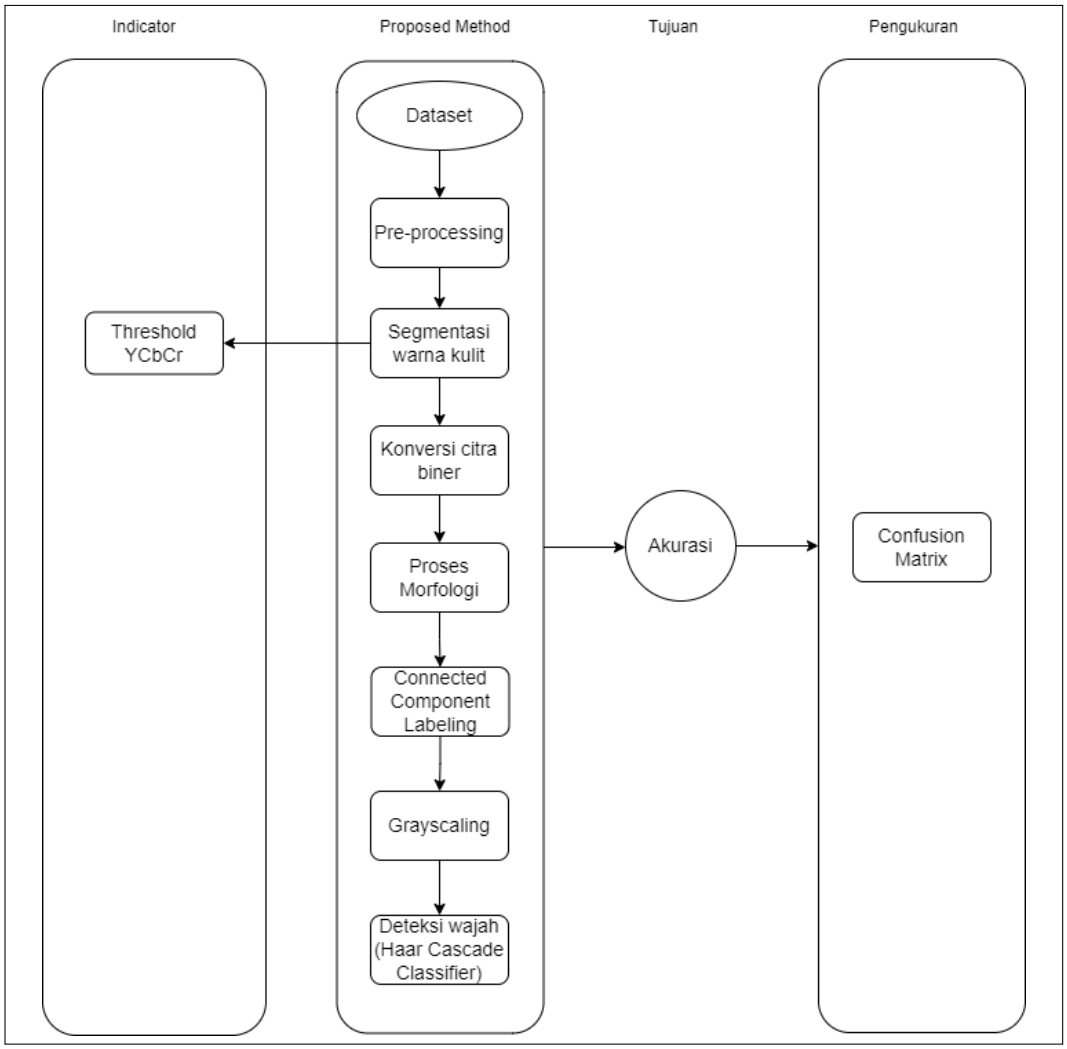
Di bab 1 telah dijelaskan bahwa deteksi wajah memiliki beberapa masalah. Masalah yang dihadapi itu berupa iluminasi, halangan, *pose* wajah, orientasi dan ekspresi. Iluminasi merupakan masalah akan pencahayaan pada citra. Halangan dapat berupa penggunaan aksesoris pada wajah, seperti kacamata, kumis dan jenggot. *Pose* wajah merupakan masalah akan posisi wajah yang tidak menghadap lurus. Orientasi berupa rotasi pada citra input atau gambar. Ekspresi merupakan masalah emosi pada wajah, seperti senang, sedih, marah, jijik dan kecewa. Deteksi wajah menjadi langkah pertama untuk menggunakan wajah sebagai fitur utama untuk dikembangkan. Maka dari itu, diperlukan sebuah metode yang dapat mendeteksi wajah dengan cepat dan memiliki akurasi yang baik.

Penelitian ini menggunakan metode *Color Space Segmentation* dan *Haar Cascade Classifier* untuk mendeteksi wajah. *Dataset* yang akan digunakan yaitu citra yang diambil dari *Bao Face Dataset* dan *Head Pose Image Dataset*. Pada *Bao Face Dataset* terdapat citra yang memiliki satu wajah dan foto grup yang terdiri dari banyak wajah. Sedangkan pada *Head Pose Image Dataset* terdapat citra yang memiliki satu wajah dengan berbagai *pose*. *Pose* yang dimaksud dapat berupa wajah menghadap lurus, atas, bawah, kiri dan kanan.

Format citra yang dimasukkan berupa JPG atau JPEG dengan citra RGB. Citra yang diinput tidak hanya berisi wajah manusia, namun ada juga yang berisi latar belakang dan bagian tubuh lainnya. Hasil dari sistem akan menunjukkan wajah yang telah terdeteksi.

3.2 Kerangka Pemikiran

Berikut ini adalah kerangka pemikiran dari metode yang diusulkan untuk melakukan deteksi wajah :



Gambar 3.1 Kerangka Pemikiran

Seperti pada gambar 3.1, penelitian ini akan melakukan deteksi wajah berdasarkan metode *Color Space Segmentation* dan *Haar Cascade Classifier*. Metode yang diusulkan untuk melakukan deteksi wajah diuraikan sebagai berikut:

1. Indikator: berisikan parameter dan faktor yang dapat mempengaruhi hasil proses metode dalam penelitian. Pada tahap segmentasi warna kulit, faktor yang mempengaruhi yaitu *threshold* dimana nilai piksel dalam citra ditentukan apakah termasuk kulit atau bukan kulit.

2. Metode yang diusulkan pada penelitian ini yaitu :

   (a) Pre-processing : di tahap ini akan dilakukan konversi warna dari ruang warna RGB menjadi YCbCr.

   (b) Segmentasi warna kulit : di tahap ini akan dilakukan pemisahan area kulit

dan bukan kulit berdasarkan *threshold* yang sudah ditetapkan pada persamaan 2.3.

(c) Konversi citra biner : di tahap ini akan dilakukan binarisasi citra. Warna pada citra akan dibagi menjadi hitam dan putih.

(d) Proses morfologi : di tahap ini akan dilakukan pengurangan *noise* dengan proses morfologi *opening*.

(e) *Connected Component Labeling* : di tahap ini akan dilakukan pemisahan objek yang berdekatan dengan diberikan label untuk setiap objek. *Connectivity* yang digunakan adalah 8 agar jangkauan yang lebih luas sehingga objek yang berdekatan akan lebih terhubung.

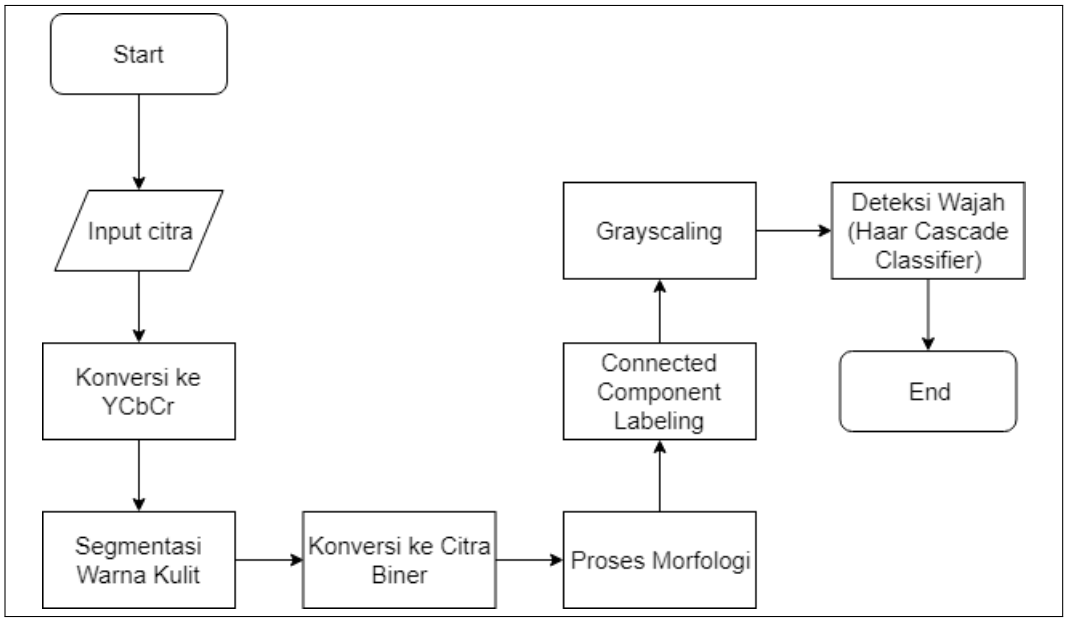
(f) *Grayscaling* : di tahap ini citra sebelumnya akan dikembalikan ke semula. Kemudian citra akan dipotong berdasarkan koordinat yang sudah didapatkan dari tahap sebelumnya. Citra hasil potongan tersebut akan diubah ke citra *grayscale* untuk mempermudah proses deteksi wajah.

(g) Deteksi wajah : di tahap ini akan dilakukan deteksi wajah menggunakan metode *Haar Cascade Classifier*.

3. Tujuan dari penelitian ini yaitu untuk menguji akurasi yang didapatkan dari metode-metode yang digunakan dalam sistem ini dan diterapkan pada aplikasi.

4. Pengukuran yang digunakan dalam penelitian ini yaitu *confusion matrix* untuk menghitung akurasi yang didapatkan. Pengukuran ini memiliki 4 nilai dalam persentase yaitu *precision, recall, accuracy* dan *f-measure*.

3.3 Analisis Urutan Proses Global



Gambar 3.2 *Flowchart* Sistem Deteksi Wajah

Berikut ini adalah uraian dari *flowchart* pada gambar 3.2 yang dilakukan dalam penelitian ini:

Proses dimulai dari masukan citra yang akan dideteksi wajahnya. Kemudian masukan citra akan dikonversi ke *color space* YCbCr. Setelah dikonversi warnanya, akan dilakukan proses segmentasi warna kulit untuk memisahkan daerah kulit dan bukan kulit. *Threshold* yang digunakan akan disesuaikan berdasarkan persamaan 2.3.

Selanjutnya, akan dilakukan proses binarisasi citra yang akan mengubah daerah yang bukan kulit menjadi warna hitam dan daerah yang merupakan kulit akan diubah menjadi warna putih. Untuk mengurangi *noise*, akan dilakukan proses morfologi. Setelah *noise* berhasil dikurangi, akan dilanjutkan proses *Connected Component Labeling* dengan pemberian label terhadap objek yang saling berhubungan lalu dipisahkan menjadi objek-objek yang lebih kecil.

Setelah diberikan label pada objek yang saling berhubungan, citra akan diubah ke semula. Citra akan dipotong berdasarkan koordinat yang sudah didapatkan dari proses sebelumnya. Citra hasil potongan tersebut akan diubah ke citra *grayscale*. Kemudian dilakukan proses deteksi wajah menggunakan *Haar Cascade Classifier*. Hasil luaran yang akan didapatkan yaitu wajah yang telah terdeteksi.

3.3.1 *Data Sampling* Citra

Di tahap ini, sistem akan dimasukkan citra dengan format JPG atau JPEG. Citra pada *Bao Face Dataset* dan *Head Pose Image Dataset* berupa citra dengan ruang warna RGB. *Bao Face Dataset* memiliki citra dengan kondisi pencahayaan dan posisi wajah yang beragam. *Head Pose Image Dataset* memiliki citra dengan wajah menghadap -90 hingga +90 derajat. Nilai minus (-) mengartikan wajah menghadap bawah atau kiri sedangkan nilai positif (+) mengartikan wajah menghadap atas atau kanan.

Pada penelitian ini, citra wajah yang diuji menghadap -60 hingga +60 dan jumlah masukan citra masing-masing *dataset* untuk diuji yaitu *Bao Face Dataset* sebanyak 30 citra (termasuk citra 1 wajah dan banyak wajah) dan *Head Pose Image Dataset* juga sebanyak 30 citra. Sehingga total jumlah masukan citra dari kedua *dataset* sebanyak 60 citra. Untuk mengetahui berapa kemiringan citra pada *Head Pose Image Dataset*, dapat dilihat nama *file* pada setiap citra dengan penjelasan sebagai berikut.

person[id][series][number][tilt][pan]

Keterangan	:
id	: nomor identitas (01, ..., 15)
series	: nomor seri (1,2)
number	: nomor *file* pada *directory* (00,01,...,92)
tilt	: menghadap *vertical* (-90, -60, -30, -15, 0, +15, +30, +60, +90)
pan	: menghadap *horizontal* (-90, -75, -60, -45, -30, -15, 0, +15, +30, +45, +60, +75, +90)

Berikut ini uraian contoh *data sampling* citra yang didapatkan dari *Bao Face Dataset* dan *Head Pose Image Dataset*.

Gambar 3.3 *Data sampling* citra dari *Bao Face Dataset* dengan 1 wajah (kiri) dan 9 wajah (kanan)

Gambar 3.4 *Data sampling* citra satu wajah dari *Head Pose Image Dataset* dengan wajah menghadap 30 derajat ke kiri (citra kiri) dan kanan (citra kanan)

Pada Gambar 3.4, wajah citra kiri menghadap *vertical* ke atas (+) sebesar 0° dan *horizontal* ke kiri (-) sebesar 30°, maka penamaan *[tilt]* akan menjadi +0 dan *[pan]* menjadi -30.

Sedangkan citra kanan pada Gambar 3.4, wajah menghadap vertical ke atas (+) sebesar 0° dan *horizontal* ke kanan (+) sebesar 30°, maka penamaan *[tilt]* akan menjadi +0 dan *[pan]* menjadi +30.

Berikut Gambar 3.5 adalah contoh pemberian nama file pada citra Gambar 3.4.

Gambar 3.5 Contoh penamaan *file* citra dari *Head Pose Image Dataset*

3.3.2 Konversi ke Ruang Warna YCbCr

Citra yang sudah dimasukkan ke dalam sistem, selanjutnya akan dilakukan tahap *pre-processing*. Teknik ini akan mengubah citra *input* menjadi ruang warna YCbCr. Untuk mengubah ruang warna menggunakan persamaan 2.2.

Gambar 3.6 Citra hasil *pre-processing* menggunakan ruang warna YCbCr

3.3.3 Segmentasi Warna Kulit

Setelah proses *pre-processing*, dilakukan segmentasi warna kulit. Pertama, perlu dicari daerah kulit dengan ruang warna YCbCr. Dengan menggunakan *threshold* menggunakan persamaan 2.3 yaitu Y>80, 100<Cb<150, 150<Cr<200 maka ruang warna YCbCr akan mendapatkan area yang merupakan warna kulit dari citra. Selanjutnya, warna kulit tersebut akan diubah menjadi biner dengan proses binarisasi agar segmentasi lebih terlihat.

Gambar 3.7 Citra hasil proses binarisasi

3.3.4 Proses Morfologi

Citra yang sudah disegmentasi melalui citra biner akan memiliki beberapa *noise* karena warna objek mungkin ada yang mirip dengan kulit manusia ikut tersegmentasi. Objek yang ikut tersegmentasi dapat dihilangkan dengan menggunakan proses morfologi yaitu *opening*. Proses *opening* merupakan gabungan dari proses erosi dan dilasi. Pada citra yang mengandung wajah, proses *opening* dilakukan dimana proses erosi akan mengurangi bagian kecil hasil segmentasi yang tidak diinginkan (*noise*), kemudian proses dilasi akan membantu memulihkan area wajah yang terkikis akibat proses erosi. *Kernel* yang akan digunakan pada proses ini yaitu 3x3. Karena jika *kernel* yang digunakan terlalu kecil, maka *noise* yang berkurang juga sedikit. Namun jika *kernel* yang digunakan besar, maka fitur penting pada wajah seperti mata, hidung, mulut dan telinga dapat menghilang.

Gambar 3.8 Perbandingan citra biner sebelum proses *opening* (kiri) dan setelah dilakukan proses *opening* (kanan)

3.3.5 *Connected Component Labeling*

Proses ini akan mengelompokkan titik-titik yang saling berdekatan menjadi objek yang diberi label. *Connected Component Labeling* ini menggunakan 8 *connectivity* karena memiliki ketetanggaan lebih luas sehingga objek yang berdekatan akan lebih terhubung. Setiap objek yang telah terdeteksi akan dipisahkan menjadi objek-objek kecil.

Ketentuan untuk *Connected Component Labeling* menggunakan *8-connectivity* sebagai berikut.

1. Cek apakah piksel saat ini memiliki nilai 1 (*foreground*). Jika ya, tandai dengan label. Jika tidak, periksa ketentuan berikutnya.
2. Cek apakah piksel di atas, kanan dan kanan diagonal ke bawah dari piksel saat ini memiliki nilai yang sama dengan piksel saat ini tetapi tidak dengan label yang sama. Jika ya, wilayah yang sama harus digabungkan. Jika tidak, periksa ketentuan berikutnya.
3. Cek apakah piksel di sebelah kanan memiliki nilai berbeda, juga di atas dan atas diagonal ke kanan memiliki nilai yang sama dengan piksel saat ini. Jika ya, berikan label piksel di atas ke piksel saat ini (dihubungkan). Jika tidak, cek kondisi berikutnya.
4. Cek apakah tetangga piksel di atas, atas diagonal ke kanan dan sebelah kanan memiliki nilai piksel yang berbeda dari piksel saat ini. Jika ya, buat id label baru dan jadikan piksel tersebut sebagai acuan untuk cek kondisi berikutnya seperti nomor 1.

Gambar 3.9 Pemberian label warna pada wajah

3.3.6 *Grayscaling*

Citra yang telah melalui proses *Connected Component Labeling* akan didapatkan kandidat wajah dengan warna atau nomor berbeda dan akan diketahui koordinat dari kandidat wajah tersebut. Citra kemudian diubah ke semula kemudian dipotong berdasarkan koordinat yang sudah didapatkan. Citra hasil potongan tersebut akan diubah ke citra *grayscale* dengan persamaan 2.1 untuk mempermudah proses deteksi wajah. Berikut ini gambar 3.10 menunjukkan citra hasil proses tahap *grayscaling*.

Gambar 3.10 Citra Grayscale

3.3.7 Deteksi Wajah menggunakan *Haar Cascade Classifier*

Citra *grayscale* kemudian digunakan untuk tahap selanjutnya yaitu deteksi wajah menggunakan *Haar Cascade Classifier*. Metode ini memiliki 4 tahap dalam prosesnya untuk mendeteksi wajah sebagai berikut.

1. Memilih fitur yang akan digunakan pada wajah. Fitur tersebut terdiri dari area gelap dan terang. Macam-macam bentuk fitur dapat dilihat pada gambar 2.7 dan penerapan fitur pada wajah dapat dilihat pada gambar 2.8.

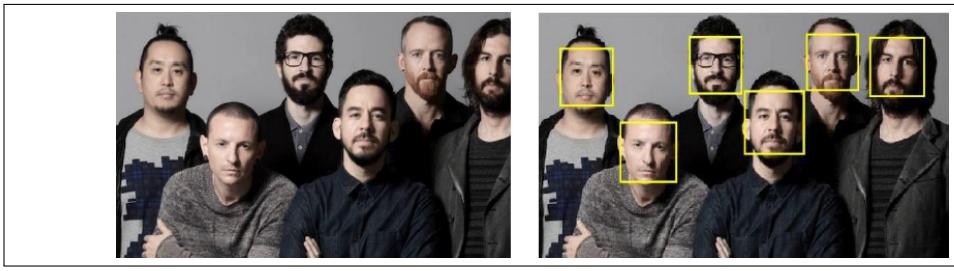
2. Untuk menghitung nilai fitur tersebut, digunakan perhitungan *integral image* dengan persamaan 2.9. *Integral image* ini dapat mempercepat perhitungan fitur dengan satu kali *scan*. Nilai fitur ini bisa disebut juga sebagai *threshold* yang akan dibandingkan dengan nilai fitur selanjutnya. Jika setelah dihitung selisih nilai piksel pada daerah gelap dan terang ternyata di atas nilai *threshold*, maka daerah tersebut dinyatakan memiliki fitur.

3. Klasifikasi *adaboost* digunakan untuk membangun *strong classifier* dari *weak classifier* yaitu fitur yang dipilih. Pada penelitian ini, *dataset* yang digunakan memiliki wajah sehingga setiap citra diberi bobot awal yang sama dengan persamaan 2.10. Kemudian untuk setiap fitur, dihitung nilai *error rate* dengan persamaan 2.11. Kemudian pilih kandidat *weak classifier* dengan nilai *error rate* yang paling sedikit. Setelah itu lakukan pemberian bobot ulang (*update* bobot) dengan persamaan 2.12. Untuk mendapatkan *strong classifier*, dilakukan perhitungan menggunakan persamaan 2.13. *Strong classifier* ini merupakan gabungan dari semua *weak classifier* yang didapatkan dari setiap tahap *adaboost*.

4. Penggunaan *cascade classifier* yang merupakan penyaringan *sub-window* dengan membentuk tingkatan penyaringan berdasarkan hasil klasifikasi *adaboost*. Pada *cascade classifier*, citra masukan dari setiap tingkatan merupakan *output* dari tingkatan sebelumnya. Semua citra yang berhasil melewati klasifikasi pertama, akan dilanjutkan ke klasifikasi kedua dan seterusnya.

   Sub-window ini berisikan fitur (hitam dan putih) seperti gambar 2.7 yang akan bergerak dari posisi x,y (0,0) pada citra. *Sub-window* memiliki ukuran 20x20 sebagai inisialisasi. Untuk total tahap penyaringan pada *cascade classifier* akan dicoba menggunakan 22 tahapan (tahap 0 hingga 21) yang nantinya akan disesuaikan kembali saat proses pengujian. Jika jumlah tahapan terlalu banyak, proses penyaringan akan memakan waktu yang sangat lama.

   Nilai *threshold* setiap tahap klasifikasi pada *cascade classifier* kemudian dibandingkan dengan perhitungan nilai fitur. Jika nilai fitur yang dihitung lebih kecil dari nilai *threshold*, maka fitur tersebut dianggap bukan wajah. Tapi jika nilai fitur tersebut lebih besar dari *threshold*, maka *sub-window* akan bergerak ke tahap klasifikasi selanjutnya. *Sub-window* ini akan memasuki tahap demi tahap hingga selesai. Jika suatu *sub-window* gagal melewati satu tahap tertentu, maka *sub-window* ini akan dieliminasi dan dianggap bukan wajah. Sebaliknya, jika *sub-window* berhasil melewati semua tahap, maka *sub-window* ini dianggap wajah. Area wajah yang ditangkap akan ditandai dengan bentuk *rectangle* seperti pada gambar 3.11 berikut ini.



Gambar 3.11 Contoh wajah pada citra belum terdeteksi (kiri) dan wajah pada citra sudah terdeteksi (kanan)

3.4 Analisis Manual

Pada bagian ini akan dilakukan analisis tahapan proses dengan melakukan perhitungan manual.

3.4.1 Mengubah Ruang Warna RGB ke YCbCr

Sebuah citra yang dimasukkan dengan ruang warna RGB akan diubah menjadi ruang warna YCbCr. Berikut ini tabel 3.1 merupakan contoh sebuah citra dengan nilai RGB :

Tabel 3.1 Nilai citra RGB 10x10 piksel

R	80	135	60	99	74	123	175	87	179	163
G	120	77	75	34	66	142	139	68	123	126
B	95	69	100	20	80	76	122	73	142	88
	53	145	220	70	69	97	142	94	89	92
	74	130	150	68	77	74	68	92	75	80
	88	98	31	110	99	88	67	78	69	76
	62	143	99	82	132	75	89	88	97	95
	83	112	108	187	175	33	75	76	86	76
	11	35	118	119	73	17	73	33	73	70
	128	170	229	234	153	89	145	167	132	134
	44	150	225	200	89	68	83	145	170	110
	66	123	90	142	67	42	92	155	78	97
	89	187	140	73	140	90	134	83	124	178
	100	162	93	71	92	65	116	73	74	67
	87	120	67	68	75	70	91	52	80	42
	201	220	162	187	154	265	95	129	177	280
	167	164	157	145	66	165	89	112	133	178
	198	129	150	132	45	142	88	117	95	154
	114	172	190	78	126	163	190	231	180	99
	121	154	88	55	98	150	111	189	122	87
	117	132	70	50	77	93	75	164	52	43
	173	240	176	256	199	93	143	97	88	120
	134	213	143	113	73	66	155	73	74	72
	89	189	167	183	88	77	114	71	33	52
	132	173	266	190	216	114	193	176	189	278
	82	142	188	118	173	184	154	64	143	153
	62	110	94	104	140	132	62	72	133	122
	90	83	190	78	168	140	188	146	190	78
	88	70	139	23	147	58	69	120	132	43
	77	61	124	11	135	34	37	62	165	32

Contoh pada tabel 3.1 menunjukkan contoh citra RGB 4x4 piksel dengan nilai R (*Red*) = 80 , G (*Green*) = 120 , B (*Blue*) = 95.

Dengan rumus pada persamaan 2.2 , maka dapat dilakukan perhitungan sebagai berikut :

$$Y = 16 + (0.257 * 80) + (0.504 * 120) + (0.098 * 95) = 106.35$$
$$Cb = 128 + (-0.148 * 80) + (-0.291 * 120) + (0.439 * 95) = 122.945$$
$$Cr = 128 + (0.439 * 80) + (-0.368 * 120) + (-0.071 * 95) = 112.215$$

Berdasarkan perhitungan YCbCr, maka dihasilkan nilai YCbCr seperti tabel 3.2, 3.3, 3.4 berikut ini :

Tabel 3.2 Hasil konversi nilai Y

Y									
106.35	96.265	79.02	60.539	76.122	126.627	142.987	79.785	137.911	130.019
75.541	128.389	151.178	79.042	82.243	86.849	93.332	94.17	83.435	87.412
74.844	112.629	107.439	142.984	145.278	53.573	83.827	80.154	91.427	85.579
77.54	147.344	197.073	190.854	106.743	77.261	104.113	147.189	143.248	115.384
97.799	157.467	105.418	77.209	105.698	78.75	117.82	79.219	93.004	99.63
171.229	167.838	151.462	150.075	93.252	181.181	93.895	117.067	137.831	192.764
117.748	150.756	116.042	68.666	105.32	142.605	128.124	186.695	128.844	89.505
136.719	203.554	149.67	156.678	112.559	80.711	142.043	84.679	79.146	88.224
97.328	142.809	188.326	134.494	172.424	150.97	149.293	100.544	149.679	176.514
91.028	78.589	147.038	48.716	146.494	84.544	102.718	120.078	147.528	60.854

Tabel 3.3 Hasil konversi nilai Cb

Cb									
122.945	115.904	141.195	112.234	132.962	101.838	115.209	127.383	128.053	105.842
137.254	111.732	65.399	146.142	138.842	130.742	116.609	121.558	123.294	124.468
99.5	89.609	133.722	113.688	89.586	114.76	125.05	107.347	120.665	122.554
125.226	113.187	68.143	97.506	108.87	113.478	122.775	129.134	93.236	118.741
123.921	105.862	109.63	126.387	113.433	126.495	114.361	117.301	123.234	100.597
136.577	104.347	124.187	116.077	105.757	103.103	126.673	127.679	104.806	102.368
127.28	115.678	105.002	122.401	114.637	101.053	100.504	110.809	88.686	106.908
102.473	113.468	133.652	137.566	115.937	128.833	111.777	123.57	107.929	112.116
111.82	109.364	75.19	111.198	107.149	115.532	81.84	114.936	116.802	95.891
122.875	122.125	113.867	114.592	119.624	105.328	96.34	98.69	133.903	117.991

Tabel 3.4 Hasil konversi nilai Cr

Cr									
112.215	154.03	119.64	157.529	130.518	124.345	145.011	135.986	151.235	146.941
117.787	136.857	167.179	125.896	122.926	137.103	160.557	129.872	134.572	133.552
123.893	147.076	123.339	86.733	116.365	147.574	134.288	136.321	133.752	136.767
163.314	138.697	139.341	147.044	157.658	139.065	154.579	136.948	117.85	139.459
124.094	141.957	150.479	129.091	150.279	138.62	137.677	133.881	149.524	178.504
140.725	155.069	130.692	147.361	168.123	173.533	130.705	135.108	150.014	174.482
125.211	137.464	174.056	138.452	141.783	137.754	165.237	148.213	158.432	136.392
148.316	141.557	140.783	185.807	182.249	139.072	125.643	138.678	137.057	150.492
151.37	143.881	168.916	160.602	149.22	100.962	151.653	176.6	148.904	185.076
129.659	134.346	151.454	152.997	138.071	165.702	182.513	143.532	151.119	144.146

3.4.2 Segmentasi Ruang Warna

Proses segmentasi akan memisahkan piksel mana saja yang merupakan area kulit dan bukan kulit sesuai dengan threshold yang telah ditetapkan pada persamaan 2.3 yaitu $100 < Cb < 150$ dan $150 < Cr < 200$. Nilai yang dipakai hanya Cb dan Cr karena nilai warna asli dari citra tersebut tidak dipengaruhi Y yang merupakan nilai dari intensitas cahayanya. Berikut ini adalah contoh citra 10 x 10 piksel pada tabel 3.5.

Tabel 3.5 Nilai CbCr pada citra dengan ukuran 10x10 piksel

Cb	122.945	115.904	141.195	112.234	132.962	101.838	115.209	127.383	128.053	105.842
Cr	112.215	154.03	119.64	157.529	130.518	124.345	145.011	135.986	151.235	146.941
	137.254	111.732	65.399	146.142	138.842	130.742	116.609	121.558	123.294	124.468
	117.787	136.857	167.179	125.896	122.926	137.103	160.557	129.872	134.572	133.552
	99.5	89.609	133.722	113.688	89.586	114.76	125.05	107.347	120.665	122.554
	123.893	147.076	123.339	86.733	116.365	147.574	134.288	136.321	133.752	136.767
	125.226	113.187	68.143	97.506	108.87	113.478	122.775	129.134	93.236	118.741
	163.314	138.697	139.341	147.044	157.658	139.065	154.579	136.948	117.85	139.459
	123.921	105.862	109.63	126.387	113.433	126.495	114.361	117.301	123.234	100.597
	124.094	141.957	150.479	129.091	150.279	138.62	137.677	133.881	149.524	178.504
	136.577	104.347	124.187	116.077	105.757	103.103	126.673	127.679	104.806	102.368
	140.725	155.069	130.692	147.361	168.123	173.533	130.705	135.108	150.014	174.482
	127.28	115.678	105.002	122.401	114.637	101.053	100.504	110.809	88.686	106.908
	125.211	137.464	174.056	138.452	141.783	137.754	165.237	148.213	158.432	136.392
	102.473	113.468	133.652	137.566	115.937	128.833	111.777	123.57	107.929	112.116
	148.316	141.557	140.783	185.807	182.249	139.072	125.643	138.678	137.057	150.492
	111.82	109.364	75.19	111.198	107.149	115.532	81.84	114.936	116.802	95.891
	151.37	143.881	168.916	160.602	149.22	100.962	151.653	176.6	148.904	185.076
	122.875	122.125	113.867	114.592	119.624	105.328	96.34	98.69	133.903	117.991
	129.659	134.346	151.454	152.997	138.071	165.702	182.513	143.532	151.119	144.146

Dengan *threshold* yang sudah ditetapkan pada persamaan 2.3 yaitu $100 < Cb < 150$ dan $150 < Cr < 200$, maka akan didapatkan hasil segmentasi seperti tabel 3.6.

Tabel 3.6 Nilai Cb dan Cr berdasarkan *threshold* pada citra dengan ukuran 10x10 piksel

Cb	1	1	1	1	1	1	1	1	1	1
Cr	0	1	0	1	0	0	0	0	1	0
	1	1	0	1	1	1	1	1	1	1
	0	0	1	0	0	0	1	0	0	0
	0	0	1	1	0	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0
	1	1	0	0	1	1	1	1	0	1
	1	0	0	0	1	0	1	0	0	0
	1	1	1	1	1	1	1	1	1	1
	0	0	1	0	1	0	0	0	0	1
	1	1	1	1	1	1	1	1	1	1
	0	1	0	0	1	1	0	0	1	1
	1	1	1	1	1	1	1	1	0	1
	0	0	1	0	0	0	1	0	1	0
	1	1	1	1	1	1	1	1	1	1
	0	0	0	1	1	0	0	0	0	1
	1	1	0	1	1	1	0	1	1	0
	1	0	1	1	0	0	1	1	0	1
	1	1	1	1	1	1	0	0	1	1
	0	0	1	1	0	1	1	0	1	0

Berdasarkan tabel 3.6, nilai Cb dan Cr yang memenuhi angka 1 dianggap sebagai daerah kulit. Berikut ini tabel 3.7 menunjukkan daerah kulit dan bukan kulit.

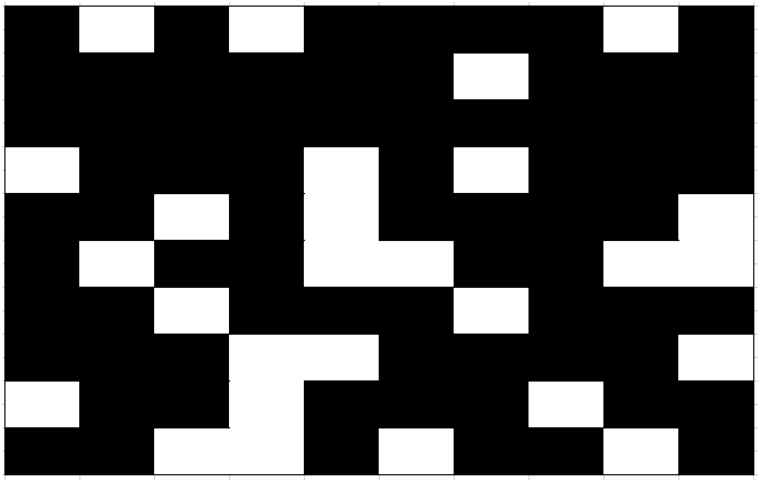
Tabel 3.7 Matriks konvolusi dengan ukuran 10x10 piksel

Bukan	Kulit	Bukan	Kulit	Bukan	Bukan	Bukan	Bukan	Kulit	Bukan
Bukan	Bukan	Bukan	Bukan	Bukan	Bukan	Kulit	Bukan	Bukan	Bukan
Bukan	Bukan	Bukan	Bukan	Bukan	Bukan	Bukan	Bukan	Bukan	Bukan
Kulit	Bukan	Bukan	Bukan	Kulit	Bukan	Kulit	Bukan	Bukan	Bukan
Bukan	Bukan	Kulit	Bukan	Kulit	Bukan	Bukan	Bukan	Bukan	Kulit
Bukan	Kulit	Bukan	Bukan	Kulit	Kulit	Bukan	Bukan	Kulit	Kulit
Bukan	Bukan	Kulit	Bukan	Bukan	Bukan	Kulit	Bukan	Bukan	Bukan
Bukan	Bukan	Bukan	Kulit	Kulit	Bukan	Bukan	Bukan	Bukan	Kulit
Kulit	Bukan	Bukan	Kulit	Bukan	Bukan	Bukan	Kulit	Bukan	Bukan
Bukan	Bukan	Kulit	Kulit	Bukan	Kulit	Bukan	Bukan	Kulit	Bukan

3.4.3 Konversi ke dalam Citra Biner

Setelah tahu piksel mana saja yang merupakan area kulit dan bukan kulit, maka selanjutnya akan dikonversi ke citra biner. Pada proses ini, daerah kulit akan ditandai dengan warna putih, sedangkan daerah bukan kulit akan ditandai dengan warna hitam. Metode yang digunakan untuk citra biner ini menggunakan *segmentasi warna* dimana proses ini akan dicek satu per satu nilai piksel apakah sesuai dengan aturan *threshold*. Jika sesuai dengan aturan *threshold*, maka akan diberi nilai 1, jika tidak bernilai 0.

Tabel 3.8 Pembagian area kulit dan bukan kulit menggunakan citra biner



3.4.4 Proses Morfologi

Hasil segmentasi warna seperti tahap sebelumnya akan menghasilkan beberapa *noise* pada citra biner berupa piksel-piksel kecil yang bukan kulit, namun nilainya lolos dari *threshold* yang digunakan, maka akan ikut tersegmentasi. Berikut gambar 3.12 merupakan contoh citra hasil segmentasi :

Gambar 3.12 Contoh citra hasil proses binarisasi

Pada Gambar 3.12 terdapat beberapa *noise* yang terlihat. *Noise* tersebut berhasil dikurangi dengan menggunakan proses morfologi yaitu *opening*. *Opening* ini melibatkan 2 proses yaitu erosi dan dilasi. Proses erosi akan mengurangi *kernel* yang ditentukan berdasarkan koordinat. Sedangkan proses dilasi akan

menambahkan koordinat berdasarkan *kernel*. Proses *opening* digunakan untuk menghilangkan objek-objek kecil dan kurus serta dapat membuat tepian citra yang memiliki ukuran besar menjadi lebih *smooth*.

Gambar 3.13 Perbandingan citra sebelum proses *opening* (kiri) dan setelah dilakukan proses *opening* (kanan)

Berikut ini ilustrasi penggunaan proses morfologi yaitu *opening* pada kasus wajah dengan *noise* yang ikut terbawa hasil segmentasi.

1	1	1
1	1	1
1	1	1

Gambar 3.14 Kernel yang digunakan dengan ukuran 3x3 piksel

Pada citra biner, nilai 0 sebagai latar belakang dan nilai 1 sebagai *foreground*. Berikut ini contoh citra biner berukuran 10x10 piksel yang berisi wajah dan *noise* ditunjukkan pada tabel 3.9.

Tabel 3.9 Citra biner berukuran 10x10 piksel berisikan wajah dan *noise*

0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	1	1	0
1	1	1	0	0	0	1	1	1	0
1	1	1	0	0	0	1	1	1	0
0	1	1	0	0	0	1	1	1	0
0	1	0	0	0	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	1
0	1	0	0	1	1	1	0	0	0
1	0	0	0	0	0	0	0	0	0

Berdasarkan persamaan 2.6, proses *opening* diawali dengan proses erosi kemudian diikuti proses dilasi. Tepian citra yang bernilai 0 dianggap sebagai latar belakang (*background*). Berikut ini hasil proses morfologi *opening*.

Tabel 3.10 Hasil proses erosi menggunakan kernel berukuran 3x3 piksel

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Tabel 3.11 Hasil proses dilasi menggunakan kernel berukuran 3x3 piksel

0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	1	1	0
1	1	1	0	0	0	1	1	1	0
1	1	1	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

3.4.5 *Connected Component Labeling*

Proses ini akan memberikan label pada objek yang terdeteksi di citra setelah proses morfologi dilakukan untuk mengurangi *noise*. Proses *Connected Component Labeling* menggunakan 8-*connectivity* dimana setiap piksel akan mengecek tetangganya melalui 8 arah dan bergerak maju menelusuri baris sampai piksel terakhir. Berikut ini contoh citra hasil proses morfologi dengan ukuran 10x10 piksel.

0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	1	1	0
1	1	1	0	0	0	1	1	1	0
1	1	1	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 3.15 Contoh citra berukuran 10x10 piksel yang berisi objek dengan arahnya

Pada gambar 3.15 , angka 0 merupakan latar belakang dan angka 1 menunjukkan objek hasil segmentasi. Pengecekan dimulai dari titik awal citra dan terus maju sampai menemukan piksel 1. Pengecekan dilakukan 8 arah jika belum ada objek tetangga yang diberi label maka akan dibuat label seperti tabel 3.12.

Tabel 3.12 Pemberian label pertama

0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	1	1	0
1	1	1	0	0	0	1	1	1	0
1	1	1	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

Dalam iterasi berikutnya, akan dicek apakah ada objek tetangga yang sudah diberi label, jika sudah ada maka akan diberi label yang sama. Jika tidak ada maka akan diberi label yang baru seperti tabel 3.13.

Tabel 3.13 Pemberian label kedua

0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	2	1	1	0
1	1	1	0	0	0	1	1	1	0
1	1	1	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

Tabel 3.14 Pemberian label kedua (lanjutan)

0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	2	2	2	0
1	1	1	0	0	0	2	2	2	0
1	1	1	0	0	0	2	2	2	0
0	0	0	0	0	0	2	2	2	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

Tabel 3.15 Pemberian label ketiga

0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	2	2	2	0
1	1	1	0	0	0	2	2	2	0
1	1	1	0	0	0	2	2	2	0
0	0	0	0	0	0	2	2	2	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	3	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

Setelah dilakukan pengecekan sampai baris terakhir, terdapat 3 label berbeda. Hasil akhir pemberian label ditunjukkan pada tabel 3.16 berikut ini.

Tabel 3.16 Hasil akhir pemberian label

0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	2	2	2	0
1	1	1	0	0	0	2	2	2	0
1	1	1	0	0	0	2	2	2	0
0	0	0	0	0	0	2	2	2	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	3	3	3	0	0	0
0	0	0	0	3	3	3	0	0	0
0	0	0	0	3	3	3	0	0	0
0	0	0	0	0	0	0	0	0	0

3.4.6 Citra *Grayscale*

Setelah objek diberikan label menggunakan *Connected Component Labeling*, hasil proses *Connected Component Labeling* akan diketahui kandidat wajah dengan warna berbeda dan akan diketahui koordinat dari kandidat wajah tersebut. Citra asli akan dipotong berdasarkan koordinat yang sudah didapatkan tersebut. Kemudian citra hasil potongan tersebut akan diubah ke citra *grayscale* agar mempermudah proses deteksi wajah.

Nilai R *(Red)*, G *(Green)*, B *(Blue)* yang diubah menjadi citra *grayscale* dijabarkan pada tabel 3.17.

Tabel 3.17 Nilai citra RGB dengan ukuran 10x10 piksel

R	80	135	60	99	74	123	175	87	179	163
G	120	77	75	34	66	142	139	68	123	126
B	95	69	100	20	80	76	122	73	142	88
	53	145	220	70	69	97	142	94	89	92
	74	130	150	68	77	74	68	92	75	80
	88	98	31	110	99	88	67	78	69	76
	62	143	99	82	132	75	89	88	97	95
	83	112	108	187	175	33	75	76	86	76
	11	35	118	119	73	17	73	33	73	70
	128	170	229	234	153	89	145	167	132	134
	44	150	225	200	89	68	83	145	170	110
	66	123	90	142	67	42	92	155	78	97
	89	187	140	73	140	90	134	83	124	178
	100	162	93	71	92	65	116	73	74	67
	87	120	67	68	75	70	91	52	80	42
	201	220	162	187	154	265	95	129	177	280
	167	164	157	145	66	165	89	112	133	178
	198	129	150	132	45	142	88	117	95	154
	114	172	190	78	126	163	190	231	180	99
	121	154	88	55	98	150	111	189	122	87
	117	132	70	50	77	93	75	164	52	43
	173	240	176	256	199	93	143	97	88	120
	134	213	143	113	73	66	155	73	74	72
	89	189	167	183	88	77	114	71	33	52
	132	173	266	190	216	114	193	176	189	278
	82	142	188	118	173	184	154	64	143	153
	62	110	94	104	140	132	62	72	133	122
	90	83	190	78	168	140	188	146	190	78
	88	70	139	23	147	58	69	120	132	43
	77	61	124	11	135	34	37	62	165	32

Berdasarkan tabel 3.17, dengan rumus *grayscale* pada persamaan 2.1 dijabarkan sebagai berikut.

Gray(0,0) = (80 + 120 + 95) / 3 = 98

Setelah dilakukan pengulangan perhitungan terhadap piksel-piksel citra sebanyak panjang dan lebar citra, kemudian dihasilkan citra *grayscale* seperti tabel 3.18 berikut.

Tabel 3.18 Hasil konversi citra RGB menjadi *Grayscale*

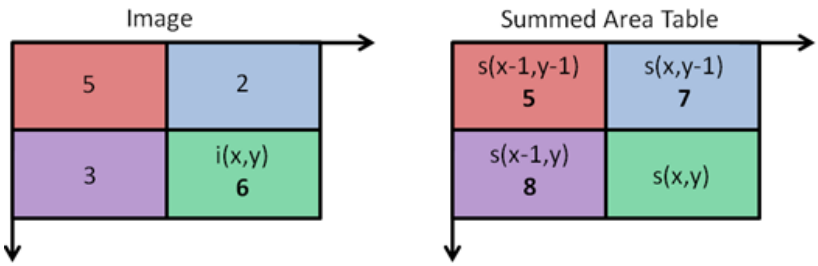
98	94	78	51	73	114	145	76	148	126
72	124	134	83	82	86	92	88	78	83
52	97	108	129	127	42	79	66	85	80
79	148	181	192	103	66	107	156	127	114
92	156	100	71	102	75	114	69	93	96
189	171	156	155	88	191	91	119	135	204
117	153	116	61	100	135	125	195	118	76
132	214	162	184	120	79	137	80	65	81
92	142	183	137	176	143	136	104	155	184
85	71	151	37	150	77	98	109	162	51

3.4.7 *Deteksi Wajah menggunakan Haar Cascade Classifier*

Citra *grayscale* kemudian digunakan *Haar Cascade Classifier* untuk deteksi wajah. *Haar Cascade* memiliki 4 tahap sebagai berikut.

1. Setiap fitur Haar terdiri dari gabungan kotak berwarna hitam dan putih. Masing-masing fitur Haar dapat ditentukan berapa banyak loncatan saat bergerak (x,y). Misalnya fitur pertama (x,y) kemudian fitur kedua (2x,y). Fitur Haar memiliki *width* dan *height* yang dapat dicoba-coba sehingga ukuran fitur dapat dihitung misalnya (*width* * *height* * 2). Nilai fitur Haar dapat dihitung dengan cepat menggunakan *integral image*.

2. *Integral image* menggabungkan unit-unit kecil menjadi satu. Unit-unit kecil yang dimaksud adalah nilai piksel. Persamaan *integral image* dapat dilihat di persamaan 2.8 dan persamaan 2.9.



Gambar 3.16 Contoh *integral image*

Dengan contoh gambar 3.16, perhitungan dapat diturunkan sebagai berikut.

$$P1(s(x-1,y-1)) = A$$
$$P2(s(x,y-1)) = A+B$$
$$P3(s(x-1,y)) = A+C$$
$$P4(i(x,y)) = A+B+C+D$$

Maka, untuk mencari nilai D dapat dijabarkan sebagai berikut :

P1 + P4 - P2 - P3 = A + (A+B+C+D) - (A+B) - (A+C) = D

27	14	44	32	36	22	11	16	28	33
21	20	20	19	18	18	18	21	22	23
18	18	16	17	17	19	20	22	23	25
19	20	18	21	21	24	25	26	27	25
23	23	23	24	25	27	28	29	29	28
24	23	22	26	27	28	30	31	32	33
24	23	23	26	27	29	31	32	33	33
23	23	24	24	25	28	29	30	32	34
22	23	24	24	25	27	28	30	32	35
23	23	23	25	26	28	29	30	31	33

Gambar 3.17 Contoh penerapan *integral image* pada citra ukuran 10x10 piksel

Pada gambar 3.17 , untuk menghitung nilai fitur berdasarkan persamaan 2.9 dapat dijabarkan sebagai berikut.

$$Nilai fitur = |total\ piksel\ gelap - total\ piksel\ terang|$$
$$h_t(x) = |(26+21-23-18)-(31+18-26-22)|$$
$$= 5$$

Hasil dari perhitungan *integral image* akan menghasilkan nilai perbedaan antar kotak (fitur warna terang dan gelap) yang biasa disebut *threshold*. Dari perhitungan diatas, didapatkan *threshold* bernilai 5.

3. Algoritme *adaboost* digunakan untuk membangun *strong classifier* dengan mengkombinasikan sejumlah *weak classifier*. *Weak classifier* yang dimaksud adalah nilai dari fitur Haar. Tahap awal proses *adaboost* yaitu menginisialisasi bobot awal untuk setiap citra.

   Perhitungan bobot awal menggunakan persamaan 2.10. Pada persamaan ini, variabel W merupakan bobot dan variabel L merupakan jumlah citra positif (memiliki kandidat wajah). Jika jumlah citra positif (L) dari *dataset* yang digunakan sebanyak 130 buah citra, maka nilai bobot awal:

   $$W(1,1) = 1\ /\ (2*L)$$
   $$= 1/\ (2*130)$$
   $$= 0.00384$$

   Nilai *error rate* berdasarkan persamaan 2.11 untuk bobot awal dan nilai *threshold* yang sudah didapatkan sebelumnya yaitu:

   $$\varepsilon_t = \sum_T^t W_t,i\ |\ h_t(x) - y_i\ |$$
   $$\varepsilon_t = (0.00384) * |5-1| = 0.01536$$

   Untuk setiap citra positif, lakukan pengulangan untuk perhitungan nilai fitur

dan *error rate* seperti di atas. Nilai bobot awal tetap sama untuk perhitungan selanjutnya sampai semua fitur selesai dihitung.

Kemudian pilih nilai *error rate* yang paling kecil. Anggap nilai *error rate* paling kecil dari semua fitur adalah 0.01536. Lakukan *update* nilai bobot awal menjadi nilai bobot baru menggunakan persamaan 2.12. Variabel E_t merupakan nilai bobot setelah *error rate* pada citra positif.

$\beta_t = \frac{E_t}{1-E_t}$
$= 0.01536 \ / \ 1 - 0.01536$
$= 0.98464$

Setelah dilakukan *update* bobot, dapat ditentukan *strong classifier* untuk menyatakan fitur mengandung wajah atau bukan wajah menggunakan persamaan 2.13.

$\alpha = log \ 1 \ / \ 0.98464$
$= 0.00671$

$H_t(x) = 0.00671 \geq 1/2 * 0.00671$
$= 0.00671 \geq 0.00335$

Dari perhitungan tersebut, dapat dikatakan bahwa citra positif dengan fitur yang dipilih tersebut benar mengandung wajah.

4. Pada *cascade classifier*, setiap tahap merupakan hasil klasifikasi *training adaboost*. Hasil tahap *adaboost* menghasilkan *error rate* yang kecil. *Threshold adaboost* menghasilkan akurasi deteksi yang tinggi dan *false positif* yang tinggi juga. Oleh karena itu, digunakan *cascade classifier* untuk mengurangi tingkat *false positive* pada citra positif hasil klasifikasi *adaboost*. Berikut ini contoh 3 tahap *cascade classifier*.

   (a) Filter pertama dipilih 1 fitur dengan akurasi sebesar 100% dan 50% tingkat kesalahan (*false positive*).

   (b) Filter kedua dipilih 5 fitur dengan akurasi sebesar 100% dan 40% tingkat kesalahan (*false positive*).

   (c) Filter ketiga dipilih 1 fitur dengan akurasi sebesar 100% dan 10% tingkat kesalahan (*false positive*).

   Pada klasifikasi *filter* pertama, tiap subcitra akan diklasifikasi menggunakan

satu fitur. Jika hasil nilai fitur dari *filter* tidak memenuhi kriteria *threshold*, hasil akan ditolak atau bukan wajah (*false positive*).

Algoritme kemudian bergerak ke *sub-window* selanjutnya dan menghitung nilai fitur lagi. Jika didapat hasil sesuai dengan *threshold* yang diinginkan, maka dilanjutkan ke tahap *filter* selanjutnya.

Jika terdapat *sub-window* yang gagal melewati salah satu *filter*, maka daerah *sub-window* tersebut digolongkan bukan wajah. Namun, jika semua *filter* dapat terlewati, maka daerah *sub-window* tersebut dianggap sebagai wajah. Wajah yang terdeteksi akan ditandai dengan sebuah *rectangle*.