2.3. Teori Pendukung

2.3.1. Pendeteksi dan Pengidentifikasi Wajah Kucing

Pendeteksi wajah kucing pada alat ini adalah sebuah proses pengolahan citra yang telah ditangkap oleh kamera. Proses ini dilakukan pada Raspberry Pi. Proses ini mampu menemukan dan membedakan wajah kucing dengan object lain di sekitarnya.

Pengidentifikasi wajah kucing pada alat ini adalah sebuah proses pengolahan citra yang dilakukan setelah pendeteksian wajah kucing dilakukan. Setelah wajah kucing terdeteksi, kemudian wajah kucing tersebut dibandingkan dengan data pada database yang telah dibuat sebelumnya.

2.3.2. Metode Deteksi Haar Classifier

Haar Cascade adalah Metode *Machine Learning* algoritma pendeteksian yang diusulkan oleh Paul Viola dan Michael Jones[13]. Metode ini mampu dirancang untuk mendeteksi wajah dan bagian tubuh dalam suatu gambar, tetapi dalam pengembangan nya dapat dilatih untuk mengidentifikasi hampir semua objek[14]. Deteksi kucing pada alat ini merupakan pengembangan dari algoritma yang dikembangkan Viola dan Jones.

Metode ini adalah pendekatan berbasis *machine learning* di mana fungsi *cascade* dilatih dari banyak gambar positif (gambar yang terdapat objek yang ingin dideteksi) dan gambar negatif (gambar yang tidak terdapat objek yang ingin dideteksi)[10][13].

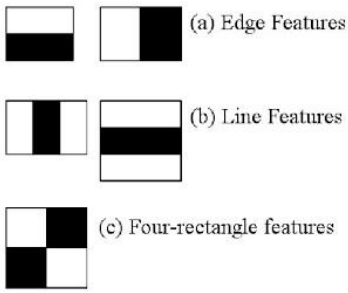
Metode ini memiliki empat tahap sebagai berikut[13].

1. Seleksi Fitur Haar
2. *Integral image*
3. Metoda *Adaptive Boost*
4. Klasifikasi Cascading

2.3.2.1. Fitur Haar

Langkah pertama dalam pendeteksian objek adalah Fitur Haar. Fitur Haar atau *haar like features* merupakan bingkai persegi dengan ukuran 24x24 *pixel* untuk mengekstraksi informasi intensitas warna hitam atau putih, kontras, dan gelap cerah dalam bentuk numerik dari daerah gambar yang dibingkai[hub.pactpub.com].

Metoda ini menggunakan prinsip *Haar-Wavelet* yaitu gelombang tunggal persegi panjang yang memiliki interval gelap(rendah) dan interval terang(tinggi). Gambar 2.1 adalah lima variasi fitur haar yang digunakan dalam pemrograman.



Gambar 2.1. Contoh fitur haar[12][13]

Nilai dari *Haar like feature* adalah selisih dari jumlah numerik piksel dari daerah persegi panjang. Di bawah ini adalah rumus sederhana dari perhitungan *haar like feature*

$$F_{(Haar)} = \sum F_{putih} - \sum F_{hitam}$$

Keterangan:

$F_{(Haar)}$ = Nilai fitur total

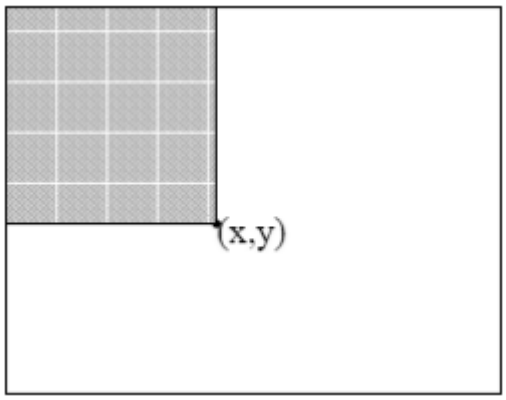
$\sum F_{putih}$ = Nilai fitur pada daerah terang

$\sum F_{hitam}$ = Nilai fitur pada daerah gelap

Untuk gambar bergerak seperti video, proses ini dilakukan secara diskrit dengan mencuplik video pada frame rate tertentu. yang kemudian dibandingkan nilai rata-rata *pixel* keduanya. Penggunaan fitur haar dilakukan untuk mempercepat proses komputasi dibandingkan dengan pemrosesan *image per pixel*[13].

2.3.2.2. Integral *Image*

Integral image merupakan suatu teknik untuk mempercepat menghitung fitur haar secara cepat dengan mengubah nilai setiap piksel menjadi suatu representasi citra baru[13]. Gambar 2.2 merupakan representasi dari integral



Gambar 2.2. representasi integral image(x,y)[13]

Integral image dihitung menggunakan persamaan berikut.

$$S_{(x,y)} = i_{(x,y)} + S_{(x,y-1)} + S_{(x-1,y)} - S_{(x-1,y-1)}$$

Keterangan:

$S_{(x,y)}$ = Nilai integral image pada (x , y);

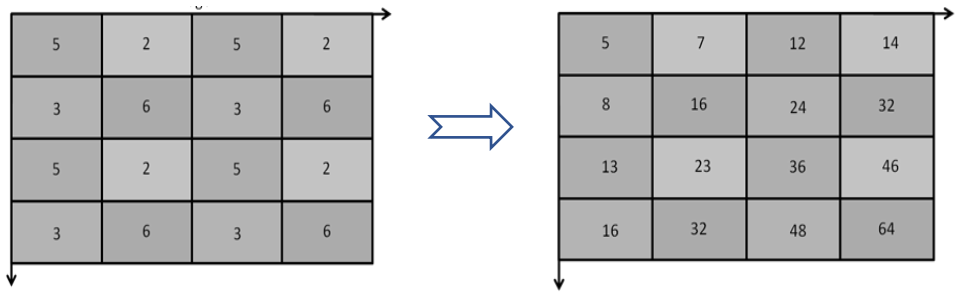
$i_{(x,y)}$ = Nilai intensitas fitur haar pada (x,y);

$S_{(x,y-1)}$ = Nilai integral image pada (x, y-1);

$S_{(x-1,y)}$ = Nilai integral image pada (x-1, y);

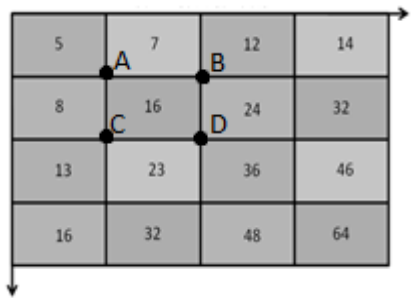
$S_{(x-1,y-1)}$ = Nilai integral image pada (x-1, y-1);

Berikut adalah contoh Hasil integral image pada Gambar 2.3.



Gambar 2.3. Hasil Integral Image

Perhitungan nilai suatu fitur dapat dilakukan secara cepat dengan menghitung nilai *integral image* pada empat buah titik sebagaimana ilustrasi pada Gambar 2.4.



Gambar 2.4. Perhitungan nilai Fitur

Jika nilai intergal image titik A adalah 5, nilai intergal image titik B adalah 12 dan nilai intergal image C adalah 13, dengan ilustrasi tersebut maka jumlah piksel di daerah D dapat diketahui dengan hanya menentukan 4 titik. Hal ini tentu mempercepat proses komputasi[12][14].

2.3.2.3. Metoda *Adaptive Boost*

Metoda Adaptive Boost atau Adaboost adalah algoritma machine learning untuk menentukan fitur-fitur haar yang spesifik yang akan digunakan untuk mengatur nilai ambang atau *threshold*. Algoritma Adaboost berfungsi untuk melakukan pemilihan atau penyeleksian fitur fitur dalam jumlah banyak dengan hanya memilih fitur-fitur tertentu[16]. Penyeleksian dilakukan dengan cara mengevaluasi setiap fitur terhadap data latih untuk dicari fitur mana yang memiliki nilai pembeda diatas threshold.

2.3.2.4. Klasifikasi Cascading

Kombinasi *Cascade of Classifier* merupakan tahap terakhir dalam metode Viola-jones. Dengan mengkombinasikan pengklasifikasian dalam sebuah Struktur *cascade* atau *Cascade of Classifier*, kecepatan dari proses pendeteksian dapat meningkat, yaitu dengan cara memusatkan perhatian pada daerah-daerah dalam image yang berpeluang saja. Hal ini dilakukan untuk menentukan dimana letak objek yang dicari pada suatu gambar.

Karakteristik dari algoritma Viola-jones adalah adanya klasifikasi bertingkat. Klasifikasi pada algoritmna ini terdiri dari tiga tingkatan dimana tiap

tingkatan mengeluarkan subimage yang diyakini bukan objek. Tahapan dirancang untuk menolak sampel negatif secepat mungkin. Asumsinya adalah bahwa sebagian besar jendela tidak mengandung objek yang menarik. Sebaliknya, hasil positif yang sebenarnya jarang terjadi dan perlu waktu untuk diverifikasi[17].

1) Benar-benar positif terjadi ketika sampel positif diklasifikasikan dengan benar.
2) Sebuah false positive terjadi ketika sampel negatif secara keliru diklasifikasikan sebagai positif.
3) Salah negatif terjadi ketika sampel positif secara keliru diklasifikasikan sebagai negatif.

Agar berfungsi dengan baik, setiap tahap dalam kaskade harus memiliki tingkat filtrasi terhadap gambar negatif yang baik.

2.3.3. Metode Local Binary Pattern Histogram

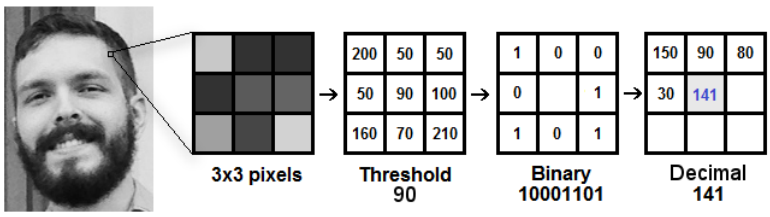
Local Binary Pattern Histogram(LBPH) adalah metode pemrosesan gambar yang diusulkan pada 2006 oleh Ahonen, T. and M. Pietikinen[18]. LBPH adalah algoritma yang digunakan untuk pengenalan wajah, yang didasarkan pada *Local Binary Pattern*[19].

Local Binary Pattern (LBP) adalah operator tekstur yang sederhana namun sangat efisien yang memberi label piksel suatu gambar dengan cara menetapkan lingkungan dari setiap piksel dan menganggap hasilnya sebagai angka biner. Ini pertama kali dijelaskan pada tahun 1994 (LBP) dan sejak itu telah ditemukan sebagai fitur yang kuat untuk klasifikasi tekstur. Lebih lanjut telah ditentukan bahwa ketika LBP dikombinasikan dengan histogram deskripsi gradien berorientasi (HOG), itu meningkatkan kinerja deteksi pada beberapa set data. Menggunakan LBP yang dikombinasikan dengan histogram, kami dapat mewakili gambar wajah dengan vektor data sederhana[20].

Ini adalah algoritma yang sangat populer, banyak digunakan, karena kekuatan pembedaan dan kesederhanaan komputas. Karena LBP adalah deskriptor visual, LBP juga dapat digunakan untuk tugas pengenalan wajah, Lebih lanjut telah ditentukan bahwa ketika LBP dikombinasikan dengan histogram deskripsi gradien berorientasi (HOG), itu meningkatkan kinerja deteksi pada beberapa set data. Hal

ini menjadi pertimbangan penting dalam pemrosesan gambar pada raspberry pi yang memiliki *resource* terbatas[20].

Gambar 2.6 menggambarkan proses sebuah ekstraksi sebuah gambar menggunakan LBP.



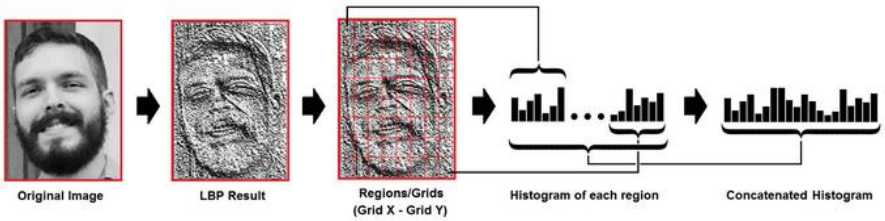
Gambar 2.6. Proses ekstraksi gambar pada metode LBP[20]

Berdasarkan Gambar 2.6, berikut adalah beberapa langkah metode LBP[20].

1) Misalkan terdapat gambar wajah dalam skala abu-abu. Kemudian diambil sebuah luas kecil berukuran 3x3 piksel nilai yang berisi intensitas setiap piksel (0 ~ 255).
2) Nilai intensitas pada pusat dari matriks digunakan sebagai *threshold*. Nilai ini akan digunakan untuk menentukan nilai-nilai baru dari piksel di sekitarnya.
3) Jika nilai pada sekitar piksel pusat lebih kecil dari *threshold* maka akan diberi nilai 0. Dan jika sama atau lebih besar maka diberi nilai 1.
4) Matriks hanya akan berisi nilai-nilai biner (mengabaikan nilai pusat). Kita perlu menggabungkan setiap nilai biner dari setiap posisi dari garis matriks demi baris menjadi nilai biner baru.
5) Kemudian, kami mengonversi nilai biner ini ke nilai desimal dan menetapkannya ke nilai pusat matriks, yang sebenarnya merupakan piksel dari gambar asli.

Pada akhir prosedur ini (prosedur LBP), kami memiliki gambar baru yang mewakili karakteristik gambar asli dengan lebih baik.

Sekarang, dengan menggunakan gambar yang dihasilkan pada langkah terakhir, kita dapat menggunakan parameter Grid X dan Grid Y untuk membagi gambar menjadi beberapa grid, seperti yang dapat dilihat pada Gambar 2.7.



Gambar 2.7. konversi data LBP ke LBPH