KLASIFIKASI IKAN MENGGUNAKAN VIOLA-JONES OBJECT DETECTION FRAMEWORK

Skripsi

Disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer

Oleh:
Nehemiah Austen Pison
1313619021

PROGRAM STUDI ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI JAKARTA

2023

LEMBAR PERSETUJUAN

Dengan ini saya mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta

Nama	: Nehemiah Austen Pison
No. Registrasi	: 1313619021
Program Studi	: Ilmu Komputer
Judul	: Klasifikasi Ikan Dengan Menggunakan Viola-Jones Object Detection Framework

Menyatakan bahwa proposal ini telah siap diajukan untuk seminar pra skripsi.

Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Muhammad Eka Suryana, M.Kom	**Med Irzal, M.Kom**
NIP. 19770615 200312 1 001	NIP. 19851223 201212 1 002

Mengetahui,
Koordinator Program Studi Ilmu Komputer

Dr. Ria Arafiyah, M.Si
NIP. 19751121 200501 2 004

KATA PENGANTAR

Puji dan Syukur penulis panjakatkan kepada kehadirat Tuhan Yang Maha Esa, karena atas rahmat dan karunia-nya yang melimpah penulis dapat menyelesaikan proposal skripsi ini dengan baik. Adapaun jenis penelitian dengan judul *Klasifikasi Ikan Dengan Menggunakan Viola-Jones Object Detection Framework*.

Dalam menyelesaikan proposal ini, penulis selalu mendapat bantuan dari orang di sekitar penulis baik dalam bentuk bimbingan dalam mengerjakan proposal ini maupun dorongan semangat dalam pengerjaan. Oleh maka dari itu, penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Yth. Para petinggi di lingkingkungan FMIPA Universitas Negeri Jakarta
2. Yth. Ibu Dr. Ria Arafiyah, M.Si selaku Koordinator Program Studi Ilmu Komputer.
3. Yth. Bapak Muhammad Eka Suryana,M.Kom selaku Dosen Pembimbing I yang telah membimbing dalam pengerjaan proposal skripsi ini
4. Yth. Bapak Med Irzal, M.Kom selaku Dosen Pembimbing II yang telah membimbing dalam pengerjaan proposal skripsi ini.
5. Orangtua penulis yang telah memberi dukungan selama pengerjaan proposal skripsi ini.
6. Teman-teman yang telah memberikan dukungan dan bantuan dalam pengerjaan proposal skripsi ini.

Dalam penulisan proposal skripsi ini penulis menyadari keterbatasan ilmu pengetahuan dan kemampuan penulis yang menyebabkan proposal ini jauh dari sempurna, baik dari segi penulisan, penyajian materi, dan juga bahasa. Oleh karena itu penulis meminta kritik dan saran yang dapat djadikan sebagai pembelajaran serta dapat membangun penulis agar lebih baik lagi dalam mengerjakan tugas-tugas dan permasalahan yang ada kedepannya.

Akhir kata, penulis berharap proposal skripsi ini dapat bermanfaat bagi semua pihak baik itu bagi FMIPA Universitas Negeri Jakarta, teman-teman dari progran studi Ilmu Komputer Universitas Negeri Jakarta dan para pembaca sekalian.

Semoga Tuhan Yang Maha Esa senantiasa membalas kebaikan semua pihak yang telah membantu penulis dalam menyelesaikan proposal ini.

Jakarta, 2 Juni 2023

Nehemiah Austen Pison

DAFTAR ISI

DAFTAR ISI vi

DAFTAR GAMBAR vii

DAFTAR TABEL viii

I PENDAHULUAN 1

1.1 Latar Belakang Masalah 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan Penelitian 3
1.5 Manfaat Penelitian 4

II KAJIAN PUSTAKA 5

2.1 Pengertian Klasifikasi Objek 5
2.2 *Viola Jones Object Detection Framework* 5
2.2.1 *Features* 6
2.2.2 *Integral Image* 7
2.2.3 *Adaboost* 9
2.2.4 *Weaklearn* 10
2.2.5 *Attentional Cascade* 11

III METODOLOGI PENELITIAN 13

3.1 Tahapan Penelitian 13
3.2 Desain Sistem 13
3.3 Training *Strong Classifier* 14
3.3.1 *Input Dataset* Pelatihan 15
3.3.2 Pembuatan *Haar like Features* 15
3.3.3 Pembuatan *Decision Tree* 17
3.3.4 *Boosting* 17
3.3.5 Pembuatan *Attentional Cascade* 19
3.4 Skenario Eksperimen dan Validasi 20
3.4.1 *Pre-processing* dan Penghitungan *Integral Image* 20
3.4.2 Klasifikasi 20

3.4.3 Anotasi 22

DAFTAR PUSTAKA **23**

DAFTAR GAMBAR

Gambar 2.1 Beberapa *Haar-like features* yang digunakan framework Viola-Jones. 6
Gambar 2.2 Sebuah nilai *Integral Image* (x, y) dan area yang diwakilinyak 7
Gambar 2.3 Jumlah dari intensitas cahaya pada persegi D dapat dihitung dengan 4 referensi *array*. Nilai dari 1 adalah jumlah intensitas cahaya pada persegi A, Nilai dari 2 adalah persegi A + B, nilai dari 3 adalah A + C dan nilai 4 adalah A + B + C + D. 8

Gambar 3.1 Diagram alir untuk algoritma pelatihan klasifikasi objek . . . 13
Gambar 3.2 Diagram alir untuk algoritma pendeteksian objek 13
Gambar 3.3 Contoh gambar Abudefduf, Amphiprion, Chaetodon dan contoh gambar-gambar negatif 15
Gambar 3.4 Sebuah fitur dua persegi menghadap ke kiri, lokasi x = 12 piksel, lokasi y = 10 piksel, dengan ukuran 8 x 8 piksel. . . . 16
Gambar 3.5 Contoh sebuah *decision tree* dengan kelas 0, 1, 2 dan 3 17
Gambar 3.6 *Workflow* dari *Attentional Cascade* 19
Gambar 3.7 Gambaran ukuran *sub-window* 72x41 piksel pada gambar 300x220 21
Gambar 3.8 Gambaran *sub-window* yang berhasil mengklasifikasi ikan didalam gambar 22

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Sektor ikan di Indonesia adalah sebuah pasar yang sangat besar, nilainya mencapai Rp 2.400.000.000.000.000 rupiah pada tahun 2022. Meskipun pasar yang besar, potensi perikanan di Indonesia belum maksmimal, tutur Menteri Kelautan dan Perikanan RI Wahyu Sakti Trenggono. Namun produktivitas perikanan di Indonesia masih kalah dari Vietnam. Ia menegaskan bahwa potensial perikanan tersebut harus ditangkap dengan baik terutama di sektor budidaya yang saat ini berjalan dalam sekala kecil, berbeda dengan negara tetangga seperti Vietnam yang mengusai eksport ke berbagai negara (Sunartono 2023).

Budidaya ikan di Indonesia memiliki problem yang lumayan besar yaitu diperlukannya usaha yang besar untuk menghitung dan mengawasi jumlah ikan yang dibudidayakan. Dalam penghitungan bibit pada proses jual beli contohnya, dalam menghitungan bibit lele para pedagang masih menghitung ikan dengan cara manual (Al-Amri 2020). Bibit ikan dipindahkan satu persatu atau ditimbang sesuai berat untuk mendapatkan jumlah ikan. kedua cara tersebut antara sangat tidak efisien atau kurang akurat. Dalam metode penghitungan, ikan dihitung satu per satu dengan tangan atau dengan bantuan sendok atau centong, yang memungkinkan penghitung untuk mengambil ikan dengan jumlah tertentu. Metode ini bisa memakan waktu yang cukup lama bila ikan yang dihitung berjumlah besar, karena itu metode ini biasanya hanya digunakan dalam menghitung ikan dalam jumlah yang sedikit. Sementara itu metode penimbangan hanya menghasilkan jumlah perkiraan yang tidak selalu akurat, namun cepat. Dalam metode ini bibit ikan dimasukan ke dalam suatu wadah yang lalu ditimbang. Berat hasil penimbangan lalu bisa dijadikan acuan kira-kira jumlah ikan yang terdapat di dalam wadah. Metode ini cepat dan cukup efisien dalam menghitung ikan dalam jumlah yang sangat besar. Namun jumlah bibit ikan tidaklah akurat dan hanya berupa perkiraan belaka.

Problem perhitungan ikan ini akan sangat terasa pada industri budidaya ikan yang sangat mementingkan kepadatan populasi dalam tempat budidaya ikan. Populasi ikan yang berlebihan dapat memperlambat pertumbuhan ikan (Diansari dkk., 2013), tetapi di sisi lain populasi ikan yang terlalu kecil akan mengurangi

efisiensi lahan yang dimiliki peternak ikan. Dalam mengatasi problem Al-Amri 2020 menciptakan sebuah sistem penghitungan menggunakan sensor *proximity*. Hasil uji coba mendapat hasil yang baik dengan persentase error sebesar 4,07% dengan penghitungan memakan waktu 228 detik per 1000 ikan. Jauh lebih cepat dibanding kecepatan hitung manual yang memakan waktu 20 menit per 1000 ikan. Cara lain dipakai oleh Rusydi 2019 untuk mendeteksi ikan. Rusydi menciptakan sebuah alat penghitungan dengan katup otomatis yang akan terbuka bilamana jumlah ikan yang diinginkan telah tercapai. Alat tersebut mendeteksi ikan menggunakan konsep *through beam* di mana ikan akan terdeteksi ketika melewati pipa oleh inframerah dan photodioda. Alat tersebut dapat mendeteksi ikan dengan kecepatan 58 ms per ikan dengan tingkat akurasi 100%.

Penggunaan alat deteksi fisik seperti yang digunakan Al-Amri 2020 maupun Rusydi 2019 memiliki beberapa kekurangan, seperti ukuran ikan bergantung kepada ukuran alat yang dipergunakan. Alat Rusydi sangat bergantung dengan kelandaian dan kecepatan lewat ikan yang melewati pipa sensor, mengganti ukuran ikan yang akan dideteksi mengharuskan tes ulang untuk mendapatkan pengaturan alat yang paling optimal (dalam tes, Rusyidi menemukan bahwa kelandaian pipa 30° memberikan hasil paling akurat dalam mendeteksi ikan). Sementara pada alat Al-Amri, lubang keluar ikan yang perlu dimodifikasi untuk mengamodasi ikan yang lebih besar. Metode-metode tersebut sangatlah tidak fleksibel dalam industri peternakan ikan yang tidak hanya menternakan satu jenis ikan saja.

Deteksi Objek Cepat (*Rapid Object Detection*) adalah sebuah algoritma yang diciptakan untuk pendeteksian muka Viola dkk., 2004. Viola menjelaskan kalau algoritma deteksi yang diciptakannya dapat mendeteksi muka dari gambar berukuran 384 x 288 pixel dari kamera berkecepatan 15 *frame* per detik. Deteksi objek dapat digunakan untuk berbagai hal seperti anotasi gambar, penghitungan mobil, deteksi muka, rekognisi muka, pelacakan gambar dan lain-lain. Setiap algoritma deteksi objek bekerja dengan cara yang berbeda-beda namun dengan konsep yang kurang lebih sama. Setiap kelas objek pasti memiliki fitur yang dapat menunjukan jati diri objek tersebut, misalnya objek bola sepak pastilah bulat dan umumnya memiliki dua warna yaitu hitam dan putih dengan pola yang spesifik. Muka manusia memiliki mata, hidung dan mulut yang dapat dibedakan dengan mahluk lainnya misalnya dengan kucing. Metode-metode deteksi objek umumnya menggunakan pendekatan neural network dan *non-neural network* untuk mendefinisikan fitur dari kelas objek yang berusaha dideteksi.

Adaboost (Freund dkk., 1996) adalah sebuah pendekatan *non-neural network* yang sering digunakan untuk mendefinisikan fitur dari objek yang ingin dideteksi (Weber 2006). Adaboost telah digunakan untuk deteksi berbagai objek seperti deteksi plat nomor kendaraan bermotor (Ho dkk., 2009), deteksi muka (Viola dkk., 2004), deteksi pesawat terbang (Freund dkk., 1996) dan lain-lain. Adaboost mencari fitur sebuah kelas objek dengan menggunakan sekumpulan *weak learner* untuk membuat sebuah *strong learner*. Kumpulan weak learner tersebut nantinya akan dinilai sesuai dengan akurasi mereka, di mana weak learner yang secara konsisten benar menebak fitur sebuah objek akan memiliki nilai lebih dalam keputusan klasifikasi akhir.

Berdasarkan latar yang telah dijelaskan, penulis mengusulkan untuk mendeteksi jenis ikan dengan menggunakan metode *Viola-Jones Object Detection Framework*. *Viola-Jones Object Detection Framework* berguna untuk mengkonstruksi sebuah *classifier* objek ikan yang nantinya dapat mendeteksi ikan dalam input gambar maupun video. Hasil yang diharapkan adalah sistem mampu mengklasifikasi ikan dari gambar secara akurat.

1.2 Rumusan Masalah

Dari uraian permasalahan di atas, perumusan masalah dalam penelitian ini adalah '**Bagaimana caranya mengklasifikasi ikan menggunakan metode *Viola-Jones Object Detection Framework*?**'.

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. Klasifikasi ikan menggunakan *Viola-Jones Object Detection Framework*.
2. Klasifikasi harus bisa melakukan klasifikasi tiga kelas genus ikan, Abudefduf, Amphiprion, Chaetodon, dan satu kelas negatif.
3. Klasifikasi dilakukan dengan gambar tampak samping ikan saja.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah menmbuat program yang mampu mengklasifikasi ikan dengan menggunakan *Viola-Jones Object Detection Framework*.

1.5 Manfaat Penelitian

1. Bagi penulis

   Memperoleh gelar sarjana dalam bidang Ilmu Komputer, serta menambahkan pengalaman dalam pembuatan sebuah program komputer dengan aplikasi dunia nyata. Dan menambahkan pengetahuan penulis tentang deteksi objek, metode *Adaboost* dan *Harr-Like Features*.

2. Bagi Program Studi Ilmu Komputer

   - Mahasiswa

     Diharapkan penelitian ini dapat digunakan sebagai penunjang referensi, khususnya pustaka tentang klasifikasi object dengan *Viola-Jones Object Detection Framework*.

   - Bagi Peneliti Selanjutnya

     Diharapkan penelitian ini dapat digunakan sebagai dasar atau kajian awal bagi peneliti lain yang ingin meneliti permasalahan yang sama.

BAB II

KAJIAN PUSTAKA

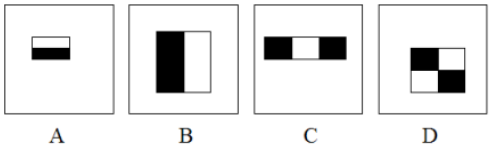
2.1 Pengertian Klasifikasi Objek

Fungsi dari klasifikasi objek adalah untuk memberikan deskripsi label ke sebuah segmentasi objek. Bila dilihat dari representasi fitur objek, ini dapat dicapai dengan melihat tanda-tanda keberadaan fitur yang mengindikasikan kelas dari objek. Hal ini umumnya dicapai dengan menentukan *threshold* diantara kelas-kelas yang direpresentasikan oleh *training set* yang sudah dilabeli. Pelatihan dilakukan dengan secara data yang sudah dilabeli secara manual atau data yang belum dilabeli. Pelatihan otomatis umumnya mementingkan distribusi dari setiap kelas, dan melabeli setiap contoh dengan sesuai, Namun, isu yang umum didalam kedua kasus adalah pemilihan fitur-fitur yang akan digunakan untuk klasifikasi objek (Renno dkk., 2007).

2.2 *Viola Jones Object Detection Framework*

Paul Viola dan Michael, J, Jones (ibid.) mempublikasikan sebuah makalah ilmiah dengan judul "*Robust Real-Time Face Detection*". Makalah tersebut mendeskripsikan sebuah framework pendeteksian wajah yang dapat memproses gambar secara cepat dengan tingkat akurasi yang tinggi. Ada tiga kontribusi penting dari makalah tersebut: Pertama adalah pengenalan sebuah representasi gambar baru yang dipanggil *Integral Image* yang memungkinkan penghitungan fitur yang dipakai oleh detektor dilakukan dengan cepat. Yang kedua adalah sebuah classifier yang efisien dan sederhana, yang dibuat menggunakan algoritma pembelajaran *Adaboost* (Freund dkk., 1996) untuk memilih sejumlah fitur-fitur kritis dari fitur-fitur potensial. Yang ketiga adalah sebuah metode untuk menggabungkan fitur-fitur tersebut dalam sebuah bentuk *cascade* yang memungkinakan algoritma untuk memfokuskan deteksi di area-area yang memiliki potensial saja.

2.2.1 *Features*

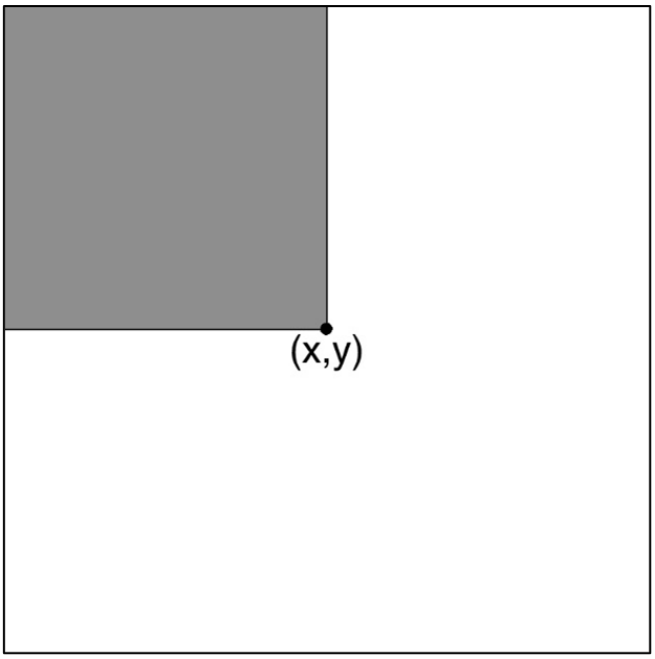


Gambar 2.1: Beberapa *Haar-like features* yang digunakan framework Viola-Jones.

Ada banyak alasan dimana penggunaan fitur lebih baik daripada nilai piksel secara langsung. Alasan paling umum adalah fitur dapat berfungsi untuk meng-*encode ad-hoc domain knowledge* yang sulit dipelajari dengan jumlah data pelatihan yang terbatas. Untuk sistem *framework Viola-Jones* ada alasan besar lainnya untuk penggunaan fitur yaitu sistem berbasis fitur beroperasi lebih cepat daripada sistem yang berbasis nilai piksel.

Fitur sederhana yang digunakan mirip dengan *Haar Basis Function* yang digunakan Papageorgiou dkk., 1998. Lebih tepatnya, tiga jenis fitur. Nilai dari sebuah *fitur dua persegi* adalah perbedaaan diantara jumlah nilai piksel didalam dua area persegi. Area-area tersebut memiliki ukuran dan bentuk yang sama, dan juga bersebelahan secara horizontal dan vertikal. Subuah *fitur tiga persegi* menghitung jumlah piksel dua area persegi di bagian luar dikurangi dengan jumlah nilai piksel persegi yang ada ditengah keduanya. Yang terakhir *fitur empat persegi* menghitung perbedaan nilai dari dua pasang persegi diagonal.

2.2.2 *Integral Image*



Gambar 2.2: Sebuah nilai *Integral Image* (x, y) dan area yang diwakilinyak

Fitur-fitur persegi dapat dihitung secara cepat menggunakan representasi tidak langsung dari gambar, hal ini diberi nama *integral image*. *integral image* pada lokasi x, y berisikan penjumlahan di atas dan di kiri dari x, y dan nilai x, y itu sendiri:

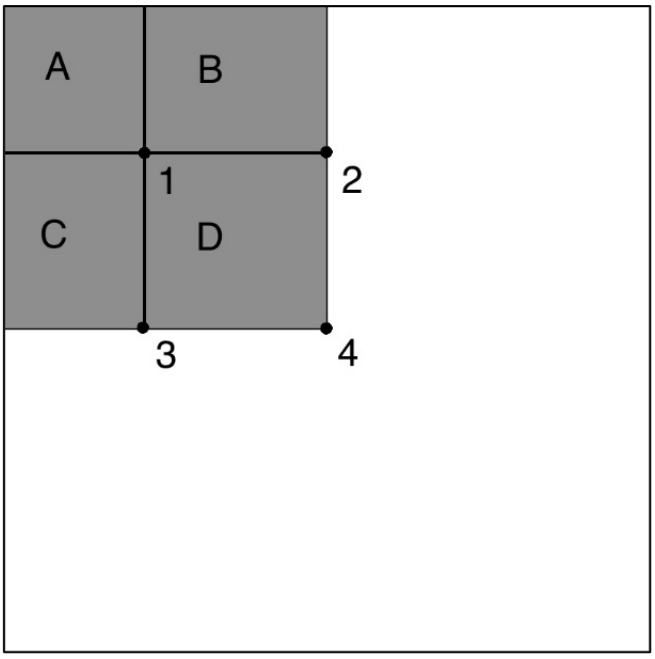
$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i\left(x', y'\right), \tag{2.1}$$

Dimana $ii(x, y)$ adalah *integral image* dan $i(x, y)$ adalah nilai piksel dari gambar aslinya. Menggunakan kedua pengulangan berikut ini:

$$s(x, y) = s(x, y - 1) + i(x, y) \tag{2.2}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \tag{2.3}$$

(Dimana *s(x, y)* adalah nilai kumulatif dari baris, *s(x, -1) = 0*, dan *ii(-1, y) = 0*) *Integral Image* dari sebuah gambar dapat dihitung dalam sekali jalan.



Gambar 2.3: Jumlah dari intensitas cahaya pada persegi D dapat dihitung dengan 4 referensi *array*. Nilai dari 1 adalah jumlah intensitas cahaya pada persegi A, Nilai dari 2 adalah persegi A + B, nilai dari 3 adalah A + C dan nilai 4 adalah A + B + C + D.

Menggunakan *integral image*, semua jumlah nilai pada persegi dapat dihitung didalam 4 referensi *array*. Jelas perbedaan diantara kedua jumlah nilai-nilai persegi dapat dihitung dengan delapan referensi. Karena peresegi dua fitur yang didefinisikan diatas melibatkan juga nilai persegi disebelahnya, mereka dapat dikomputasi dengan enam referensi *array*, delapan referensi bilamana ia adalah persegi tiga fitur, dan sembilan referensi untuk persegi empat fitur.

Kecepatan yang didapat dari penghitungan *Integral Image* ini dapat dijustifikasi bila kita membandingkannya dengan perhitungan manual. Sebagai contoh, misalkan kita sedang mencari jumlah total intensitas cahaya pada ukuran area 10x10 piksel. Cara manual mengharuskan kita menghitung sampai 100 kali untuk mendapat jumlah intensitas cahaya pada area tersebut, belum lagi proses ini harus diulang terus-menerus untuk ukuran dan lokasi yang berbeda. Dilain sisi, perhitungan menggunakan *Integral Image* hanya perlu mereferensi tabel yang sudah dibuat sebelum semua usaha klasifikasi, dalam hal ini kita hanya perlu melakukan perhitungan empat kali untuk menghitung intensitas cahaya dalam area tersebut.

2.2.3 *Adaboost*

Algorithm 1 Algoritma Adaboost

1: Diberikan contoh gambar $(x_1, y_1), \ldots, (x_n, y_n)$ dimana $y_i = 0$ untuk contoh negatif dan $y_i = 1, \ldots, N$ untuk contoh kelas-kelas positif
2: Inisialisasi bobot $w_{i,1} = \frac{1}{\text{total contoh}}$
3: Untuk $t = 1, \ldots, T$:

1. Normalisasi bobot, $w_{t,1} \leftarrow \frac{w_{t,1}}{\sum_{j=1}^{n} w_{t,j}}$
2. Pilih *weak classifier* terbaik dengan melihat error yang telah diberi bobot

$$\in_t = \min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|. \tag{2.4}$$

3. Definisikan $h_t(x) = h(x, f_t, p_t, \theta_t)$ dimana f_t, p_t, dan θ_t adalah *minimizer* dari $\in_t$.
4. Perbarui bobot:

$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i} \tag{2.5}$$

dimana $e_i = 0$ bila contoh x_i diklasifikasi secara benar, selainnya $e_i = 1$, dan $\beta_t = \frac{\in_t}{1-\in_t}$

4: *strong classifier* akhirnya adalah:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{selain itu} \end{cases} \tag{2.6}$$

dimana $\alpha_t = \log \frac{1}{\beta_t}$

Dalam *framework Viola-Jones* sebuah varian dari *Adaboost* digunakan untuk memilih fitur dan juga untuk melatih *classifier*. Didalam bentuk aslinya, algoritma pembelajaran *Adaboost* digunakan untuk mem-*boost* performa klasifikasi dari algoritma pembelajaran sederhana. *Adaboost* melakukan ini dengan menggabungkan sekumpulan *classifier* lemah untuk membuat sebuah *classifier* kuat. Didalam istilah *Boosting*, *classifier* lemah disebut juga dengan *weak learner*. Sebagai contoh, algoritma pembelajaran *perceptron* mencari dari sekelompok *perceptron* dan mengambil *perceptron* dengan tingkat kesalahan klasifikasi terendah. Algoritma pembelajaran disebut lemah karena kita tidak berharap *classifier* terbaik untuk mengklasifikasi data dengan benar. Nyatanya, *perceptron* terbaik mungkin hanya memiliki tingkat akurasi 51%. Agar *weak learner* dapat di-*boost*, ia akan dipanggil untuk menyelesaikan sederet problem pembelajaran. Setelah satu ronde

pembelajaran selesai, contoh pembelajarannya akan dibobot ulang untuk menekankan problem yang salah diklasifikasi oleh *weak learner* sebelumnya. Bentuk *final strong classifier* adalah *perceptron*, sebuah kombinasi *weak learner* berbobot yang diikuti oleh *threshold*.

2.2.4 *Weaklearn*

Algorithm 2 Metode Pembuatan *Decision Tree*

1: Anotasi semua dataset sesuai kelasnya. *Decision Tree* lalu akan dimulai dari akar
2: Pilih atribut terbaik untuk melakukan *split* dengan melakukan *Information Gain* (IG):

$$IG(S, A) = Entropi(S) - sum_v \frac{S_v}{S} x Entropi(S_v) \tag{2.7}$$

dimana:

- S: kumpulan sampel pada *node* sekarang
- A: atribut yang digunakan untuk *split*
- v: Sebuah nilai atribut A yang membagi S menjadi turunan S_v
- Entropi(S) = $-sum_c p_c log_2(p_c)$: Ukuran kemurnian pada kumpulan contoh dengan target label. Dimana p_c adalah proporsi contoh S yang ada pada kelas c.

3: *split* pohon ke *node* turunan sesuai atribut yang dipilih
4: tentukan apabila seluruh contoh sudah jatuh ke kelas yang benar, bila tidak maka ulangi langkah 2 dan 3. Hal ini dilakukan dengan memvalidasi pohon yang sudah dibuat

Framework Viola Jones menggunakan sebuah *weaklearn* yang bernama *Decision Stump*, atau sebuah *Decision Tree* yang hanya memiliki dua daun kelas saja. *Decision Tree* sendiri mampu digunakan untuk permasalahan *multi-class*.

Decision Tree menghasilkan sebuah classifier didalam bentuk sebuah pohon pilihan, sebuah struktur yang berbentuk:

- Sebuah daun, mengindikasi sebuah kelas, atau
- Sebuah *decision node* yang menspesifikasi sebagian tes untuk dikerjakan atas sebuah nilai atribut, dengan satu *branch* dan *subtree* untuk setiap hasil dari tes.

Sebuah *Decision Tree* dapat digunakan untuk mengkasifikasi sebuah kasus dengan memulai dari akar pohon dan bergerak sampai sebuah daun ditemukan. Pada

setiap *node* yang bukan merupakan daun, hasil dari tes kasus dideterminasi dan perhatian berubah ke akar dari *subtree* sesuai dengan hasil tersebut. Ketika proses pada akhirnya (dan dengan pasti) menuju ke sebuah daun, kelas dari kasus diprediksi sesuai yang ada di daun.

2.2.5 *Attentional Cascade*

Attentional Cascade adalah sebuah *cascade* dari banyak *classifier* yang dibuat untuk meningkatkan performa deteksi dengan secara radikal mengurangi waktu komputasi. Intinya *weak classifier* yang telah di-*boost* dapat dibuat lebih kecil dan efisien, yang dapat menolak mayoritas *sub-window* negatif dan mendeteksi sebagian besar dari *sub-window* positif. *Classifier* yang lebih sederhana digunakan untuk menolak mayoritas *sub-window* sebelum *classifier* yang lebih kompleks dipanggil untuk menurunkan tingkat *false positives*.

Struktur dari *cascade* merefleksikan fakta bahwa pada gambar apapun mayoritas *sub-window* pasti negatif. Oleh karena itu, *cascade* berusaha untuk sebanyaknya menolak sub-window negatif pada tahapan seawal mungkin. Sementara hasil positif akan memicu evaluasi dari setiap *classifier* dalam *cascade*, hal ini sangatlah langka.

Layaknya sebuah *Decision Tree*, sebuah *classifier* dilatih menggunakan contoh-contoh yang telah berhasil melewati tahap sebelumnya. Oleh karenanya, *classifier* pada tahap kedua menghadapi tantangan yang jauh lebih sulit daripada yang pertama. Contoh yang semakin sulit yang dihadapi *classifier* di tahap-tahap yang semakin jauh menekan seluruh *receiver operating characteristic* (ROC) kebawah.

Algorithm 3 Algoritma Pelatihan Untuk Pembuatan *Cascaded Detector*

1: Pengguna memilih nilai dari *f*, nilai maksimum *false positive* pada setiap tahap yang dapat diterima, dan *d*, nilai minimum *detection rate* per tahap yang dapat diterima
2: Pengguna memilih target keseluruhan *false positive rate*, F_{target}
3: *P* = kumpulan contoh positif
4: *N* = kumpulan contoh negatif
5: $F_0 = 1.0; D_0 - 1.0$
6: $i = 0$
7: $whileF_i > F_{target}$
$-i \leftarrow i + 1$
$-n_i = 0; F_i = F_i{-_1}$
-while $F_i > fxF_i{-_1}$

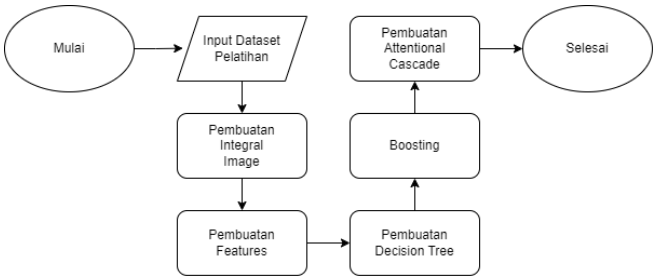
- $n_i \leftarrow n_i + 1$
- Gunakan *P* dan *N* untuk melatih sebuah *classifier* dengan fitur n_i menggunakan *Adaboost*
- Evaluasi *cascade classifier* dengan *set* validasi untuk menentukan F_i dan D_i.
- kurangi *threshold* untuk *classifier* ke-i sampai *cascade classifier* memiliki tingkat deteksi paling tidak $dxD_i{-_1}$ (hal ini juga akan mempengaruhi F_i)

$-N \leftarrow$ Ø
-if $F_i > F_target$ evaluasi *cascade detector* menggunakan *set* negatif dan masukan semua deteksi gagal ke *set* N

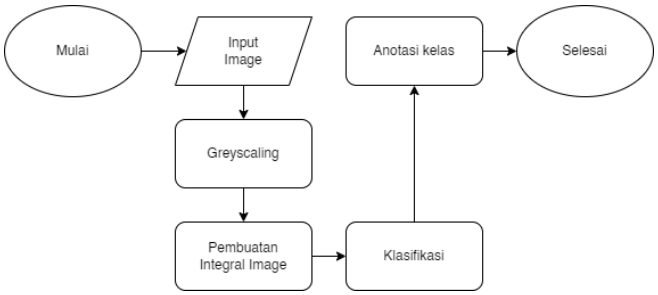
BAB III
METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Gambar *flowchart* berikut mengilustrasikan proses pelatihan dari dataset dan juga proses penggunaan yang sesungguhnya.



Gambar 3.1: Diagram alir untuk algoritma pelatihan klasifikasi objek



Gambar 3.2: Diagram alir untuk algoritma pendeteksian objek

3.2 Desain Sistem

Dalam proses pembuatan *classifier* perlu dilewati tahapan *training*. Tujuan *training* adalah untuk menciptakan suatu *strong classifier* yang nantinya dapat digunakan untuk melakukan klasifikasi yang sebenarnya. Pertama, gambar yang akan menjadi contoh pelatihan dianotasi sesuai kelasnya masing-masing dengan memberikan label yang sesuai dengan kelas mereka masing-masing, vontoh latihan ini berisikan gambar-gambar yang tidak memiliki kelas yang benar, atau *false example* dan juga gambar-gambar yang memiliki kelas yang benar, atau *positive*

example. Setelah itu gambar melalui proses *pre-processing* dan disesuaikan untuk mengoptimalkan proses pelatihan.

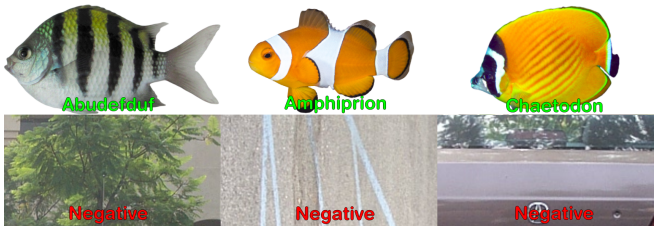
Pertama sebuah set *features* akan dibuat dengan cara mencoba semua kemugnkinan yang ada dengan bentuk fitur yang dimiliki. Set ini akan berisikan informasi fitur-fitur yang nantinya akan dipakai untuk mendapatkan nilai fitur yang sesungguhnya. Set gambar latihan lalu akan dibaca menggunakan semua fitur ini dan hasilnya akan dicatat dalam tabel csv. Algoritma lalu akan mengkonstruksi sebuah *decision tree* untuk setiap *feature* untuk menentukan nilai *feature threshold* setiap kelas. *Decision tree* lalu akan di-*boosting* untuk menentukan nilai bobot votingnya pada *strong clossifier*. Akhrinya dari sekumpulan *decision tree* ini dibuatlah sebuah *final strong classifier* yang berbentuk *cascade*.

Dengan *final strong classifier*, barulah klasifikasi objek yang sesungguhnya dapat dilakukan. Pertama gambar yang akan dideteksi akan melalui *pre-processing*. setiap sub-window akan dicek menggunakan *strong classifier* untuk menentukan kelasnya. Hal ini dilakukan pada tiga area: area kiri untuk mengklasifikasi mulut dari ikan, area tengah untuk mengklasifikasi sirip dari ikan, dan terakhir area kanan untuk mengklasifikasi bentuk ekor dari ikan. Hasil ketiga *sub-window* ini nantinya juga akan ber-*voting* dimana jika ada dua atau lebih *sub window* berhasil mengklasifikasikan kelas yang sama, maka kelas itu dipilih sebagai kelas dari objek pada gambar. Kelas dari objek lalu akan dituliskan di pojok kiri atas gambar.

3.3 Training *Strong Classifier*

Pada *Training* ada tiga tahapan yang perlu dijalankan untuk menghasilkan sebuah *Strong Classifier*, yaitu penginputan *dataset* pelatihan yang sudah dianotasi, pembuatan *features*, pembuatan *decision tree* untuk setiap *features*, *Boosting* dan pemilihan fitur untuk pemebuatan *attentional cascade*.

3.3.1 *Input Dataset* Pelatihan

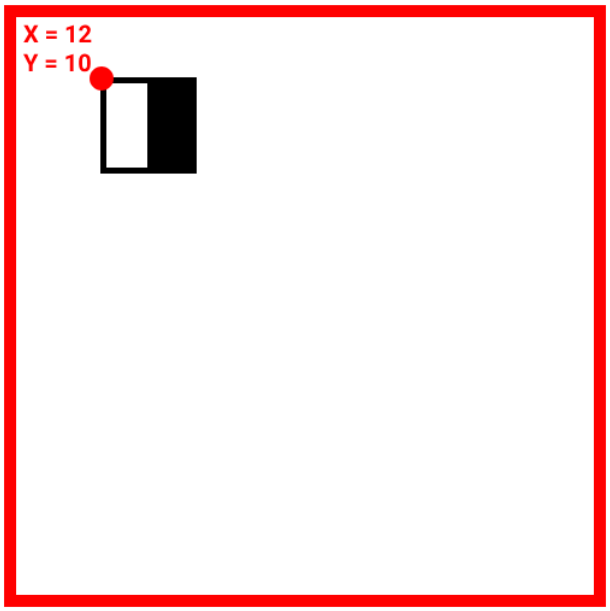


Gambar 3.3: Contoh gambar Abudefduf, Amphiprion, Chaetodon dan contoh gambar-gambar negatif

Dataset yang akan dipakai diambil dari FishBase (Dapat dilihat di https://fishbase.mnhn.fr/) yang berisikan berbagai gambar ikan termasuk dari genus Abudefduf, Amphiprion dan Chaetodon. Ketiga genus ikan ini dipilih karena bentuknya yang berbeda satu-sama lain. Selain itu *dataset* juga akan ditambahkan dari hasil penelitian lapangan secukupnya, dengan mempertimbangkan waktu komputasi pada tahap pelatihan. Untuk contoh pelatihan gambar dibuat berukuran 350x200 piksel, dengan warna *greyscale*. Selain itu juga akan dipilih contoh pelatihan negatif atau kumpulan gambar yang tidak terdapat kelas ikan dari http://www.vision.caltech.edu/datasets/ dengan jumlah yang sama, resolusi sama dan perlakuan yang sama. Anotasi dilakukan dengan menyimpan label dalam sebuah *array*, label diambil dari sumber folder gambar. Kelas 0 direservasi untuk kelas negatif, sementara kelas 1, 2 dan 3 direservasi untuk Abudefduf, Amphiprion dan Chaetodon. *Dataset* ini lalu dibagi menjadi tiga yaitu *training dataset*, *testing dataset*, dan *validation dataset* dengan jumlah yang sama.

3.3.2 Pembuatan *Haar like Features*

Fitur-fitur yang akan digunakan dalam klasifikasi dibuat berdasarkan *Haar-like Features* dan berisikan informasi penting yang dapat digunakan untuk mencari nilai sebuah fitur. Sebuah fitur berisikan tipe fiturnya, lokasi fitur tersebut didalam *sub-window*, dan ukuran dari fitur tersebut. Misalnya ada sebuah fitur, ia bertipe dua persegi menghadap ke kiri, lokasi x-nya adalah 12 piksel, dan lokasi y-nya adalah 10 piksel, dia memiliki ukuran 8 x 8 piksel.



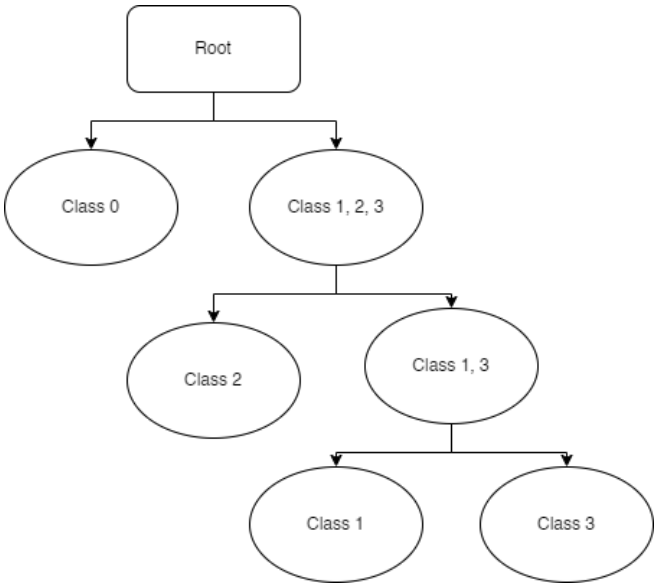
Gambar 3.4: Sebuah fitur dua persegi menghadap ke kiri, lokasi x = 12 piksel, lokasi y = 10 piksel, dengan ukuran 8 x 8 piksel.

Dengan informasi yang ada didalam sebuah fitur tersebut, nilai fitur dapat dicari dengan mudah menggunakan rumus:

$$\sum \text{nilai piksel area putih} - \sum \text{nilai piksel area hitam} \tag{3.1}$$

Pada tahap ini fitur yang dipilih akan dibuat untuk semua posibilitas lokasi yang ada dan ukuran yang ada. Hal ini dilakukan dengan membuat fitur dimulai dari kiri atas gambar dengan ukuran 2 x 2 piksel untuk fitur dua persegi, empat persegi dan diagonal. Dan fitur 1 x 3 piksel atau 3 x 1 piksel untuk fitur tiga persegi, sampai pojok kanan bawah. Hal ini juga dilakukan sampai ukuran fitur lebih besar dari *sub-window* tidak bisa muat lagi.

3.3.3 Pembuatan *Decision Tree*



Gambar 3.5: Contoh sebuah *decision tree* dengan kelas 0, 1, 2 dan 3

Setiap *weak learner* adalah sebuah *decision tree* yang akan mengambil nilai dengan fitur untuk digunakan sebagai variabel klasifikasi. fitur dapat digunakan untuk mencari sebuah nilai perbandingan intensitas cahaya dari dua area pada gambar dan mendeteksi keberadaan suatu fitur seperti perbedaan warna, garis, maupun perbedaan kontras pada gambar.

Decision tree pertama akan dibuat dari *root* atau akar, yang lalu akan bercabang hingga batas maksimum telah dicapai. *threshold* yang digunakan dalam pembuatan *decision tree* adalah salah satu nilai fitur yang dibaca dari gambar. Dengan cara ini, *decision tree* tidak perlu mencoba semua nilai yang mungkin, dan dengan demikian mempercepat proses pembuatan *decision tree*. Batas maksimum tinggi *decision tree* yang dipilih adalah tiga tingkat, hal ini dikarenakan waktu komputasi yang memakan waktu bila *decision tree* memiliki lebih dari 3 tingkat. Di lain sisi, menggunakan tingkat kurang dari tiga tidak mungkin memenuhi persyaratan klasifikasi empat kelas.

3.3.4 *Boosting*

Boosting ditunjukan untuk memberikan nilai *voting* untuk setiap *weak classifier* yang nantinya akan digunakan dalam klasifikasi akhir. Sebelum *boosting*

dimulai, *weak classifier* yang kurang diskriminatif akan dibuang. Hal ini dilakukan dengan membandingkan hasil prediksi *weak classifier* dengan label yang sebenarnya, dimana *weak classifier* yang gagak memprediksi 50% dari set tes akan dibuang. Ini dilakukan untuk mengurangi jumlah *weak classifier* yang akan dipakai berikutnya.

Pada tahap ini *boosting* akan dijalankan dari *decision tree* yang paling akurat ke yang paling tidak akurat. Penentuan akurasi ini dilakukan dengan membandingkan label contoh validasi dengan hasil prediksi setiap *weak classifier*. *Weak classifier* yang tidak akurat akan mendapat suara *voting* yang lemah. Rumus untuk mencari bobot voting α dari sebuah *weak classifier* adalah sebagai berikut:

$$\begin{aligned} \epsilon &= \frac{\sum_i \text{image weights}_i \times \text{indikator}_i}{\sum_i \text{image weights}_i} \\ \alpha &= 0.5 \times \log\left(\frac{1-\epsilon}{\epsilon + 1e - 10}\right) \end{aligned} \tag{3.2}$$

weak classifier terbaik akan mulai dan mengklasifikasi seluruh contoh *dataset* validasi dan mencatat contoh yang gagal diklasifikasi oleh *weak classifier* tersebut. Lalu bobot dari contoh tersebut akan dinaikan, dengan maksud agar bobot *voting* fitur yang paling akurat akan lebih tinggi daripada fitur-fitur yang hanya mendekati menebak. Rumus menaikan bobot:

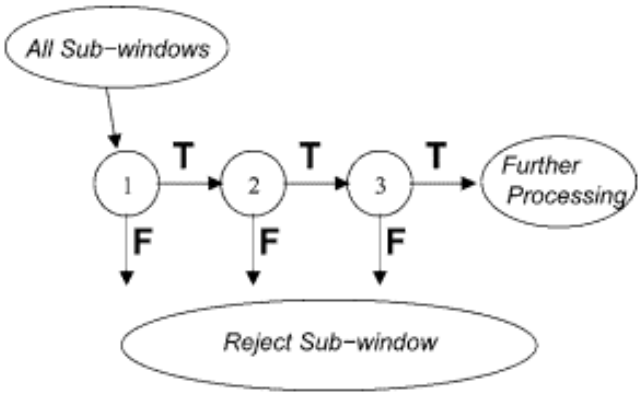
$$\text{image weights} \times = \exp(\alpha \times \text{indikator}) \tag{3.3}$$

$$\text{image weights} \div = \sum \text{image weights} \tag{3.4}$$

weak classifier berikutnya lalu akan melakukan prediksi dengan set yang sama, namun dengan bobot gambar yang sudah berubah karena *weak classifier*. Sementara bobot *voting weak classifier* sebelumnya akan disimpan ke array untuk digunakan nanti.

Ketika semua *weak learner* sudah dicari bobot *voting*-nya, akan dibandingkan akurasi *strong cloassifier* yang dibuat dengan iterasi sebelumnya. Bila didapat bahwa ada penurunan akurasi, maupun tidak ada perubahan, maka iterasi Boosting akan disudahi dan *strong cloassifier* pada iterasi ini menjadi *final strong classifier*.

3.3.5 Pembuatan *Attentional Cascade*



Gambar 3.6: *Workflow* dari *Attentional Cascade*

Karena bobot *voting* dan urutan *weak classifier* sudah ditentukan pada tahap *Boosting*, pada tahap ini hanya perlu membagi *weak classifier* menjadi beberapa *stage* yang nantinya bisa dipanggil secara terpisah. Sebuah *stage* berlaku layaknya sebuah *strong cloassifier* kecil yang ditargetkan hanya memiliki tingkat akurasi paling tidak 50% saja. Hal ini dilakukan agar pendeteksian seluruh *sub-window* dapat dilakukan tanpa harus memanggil keseluruhan dari *strong cloassifier*. Metode konstruksi sebuah *stage cascade* adalah sebagai berikut:

Algorithm 4 Cascade Train Stage

```
1:  function TRAIN_STAGE
2:      detection_rate ← 0
3:      while detection_rate < 0.5 do
4:          if len(features) == 0 then
5:              break
6:          end if
7:          self.features.append(features)
8:          detection_rate ← accuracy_score()
9:      end while
10: end function
```

Sebelum pembuatan *attentional cascade*, *weaklearner* yang kurang diskriminatif akan dibuang dari *Strong Classifier*. *Attentional cascade* akan dibuat dari *weaklearner* yang tersisa secara bertahap. Pertama, target *false positive* pada setiap *cascade* harus ditentukan oleh pengguna. *Framework VIola-Jones*

menggunakan target *false positive* 50% untuk *cascade* pertama, dan *false positive* 80% untuk semua *cascade* setelahnya. Target *false posiive* ini akan disesuaikan saat penelitian lapangan, bila target akurasi tidak berhasil dicapai dengan target *false positive* sebelumnya. Pada setiap fase *cascade*, *weaklearner* akan ditambahkan satu-persatu. Setiap sebuah *weaklearner* ditambahkan, tes akan dilakukan dengan *dataset* tes. *Weaklearner* akan terus ditambahkan hingga *false positive rate* yang ditentukan untuk fase itu dicapai. Fase akan terus bertambah hingga akurasi sempurna dicapai, atau hingga *weaklearner* sudah habis.

3.4 Skenario Eksperimen dan Validasi

Tahapan ini adalah penggunakaan *classifier* yang sebenarnya dengan tujuan memvalidasi akurasi dari *classifier* tersebut. Gambar ikan yang akan dipakai dalam proses validasi akan melalui beberapa langkah dalam tahap ini, yaitu: *Pre-processing* dalam bentuk *grayscaling*, Penghitungan *Integral Image*, dan deteksi yang sesungguhnya menggunakan *strong classsifier* yang telah dibuat dengan metode Sliding Window.

3.4.1 *Pre-processing* dan Penghitungan *Integral Image*

Untuk klasifikasi sebenarnya, gambar *input* akan diproses terlebih dahulu. Gambar awalnya akan melakui proses *pre-processing* dan dirubah kedalam warna *grayscale* untuk mempermudah penghitungan dengan bantuan *library OpenCV*. Setelah tu sebuah matriks sebesar resolusi gambar akan dibuat untuk penghitungan *Integral Image* yang nantinya akan mempercepat proses penghitungan fitur. Pembuatan *integral image* pada tahap ini sama persis dengan pembuatan pada tahap pelatihan.

3.4.2 Klasifikasi

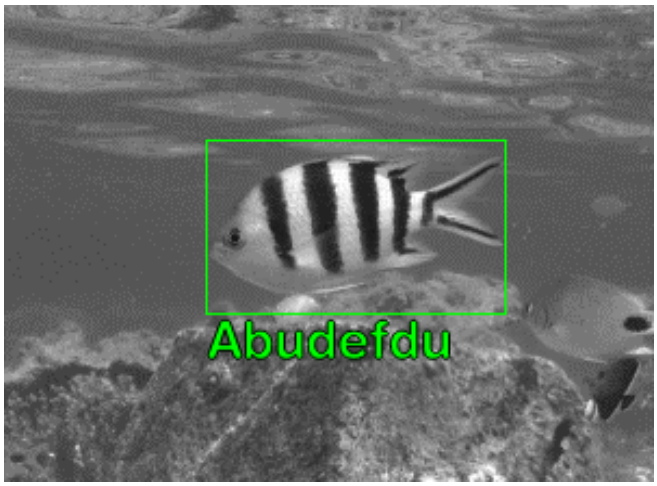
Pada tahap ini, gambar yang sudah berubah dalam bentuk *integral image* akan diklasifikasi menggunakan *strong classifier*. *Strong classifier* akan berjalan dari pojok kiri atas gambar, atau piksel pertama, mencoba untuk mengklasifikasi sebuah *sub-window* berukuran 72x41 piksel. Bila posisi tersebut sudah selesai diklasifikasi, terlepas hasilnya, *sub-window* akan bergerak ke kanan sebanyak satu piksel dan mengklasifikasi area baru tersebut. Hal ini akan terus berlanjut hingga *sub-window*

bisa dibuat lagi ke kanan. Bilama demikian *sub-window* akan kembali ke ka kanan, namun kali ini diturunkan sebanyak satu piksel. Pergerakan *sliding window* ini bertujuan agar seluruh bagian dari gambar dapat terklasifikasi.

Gambar 3.7: Gambaran ukuran *sub-window* 72x41 piksel pada gambar 300x220

Jika *sub-window* dengan ukuran 72x42 piksel sudah mengklasifikasi seluruh bagian gambar hingga pojok kanan bawah, maka *sub-window* akan diperbesar dengan faktor 1.25 dan proses dimulai lagi dari pojok kiri atas. Selain itu ukuran dan lokasi dari *feature* akan disesuaikan untuk mangakomodasi *sub-window* yang sudah diperbesar. Hal ini dilakukan untuk mengklasifikasi objek yang memiliki ukuran berbeda didalam gambar.

3.4.3 Anotasi



Gambar 3.8: Gambaran *sub-window* yang berhasil mengklasifikasi ikan didalam gambar

Setelah semua *sub-window* sudah diklasifikasi dengan menggunakan *strong classifier*. Algoritma akan menggambarkan di lokasi *sub-window* yang positif terklasifikasi (Memiliki salah satu kelas ikan yang telah dilatih ke *strong classifier*), *bounding box* untuk menganotasi kelasnya. Pengguna lalu dapat menentukan dari hasil anotasi bilamana target akurasi sudah tercapai.

DAFTAR PUSTAKA

Al-Amri, C. F. (2020). "Rancangan Bangun Fish Counter Untuk Menghitung Bibit Ikan Lele".

Diansari, R., E. Arini, and T. Elfitasari (2013). "Pengaruh Kepadatan Yang Berbeda Terhadap Kelulushidupan dan Pertumbuhan Ikan Nila (Oreochromis niloticus) Pada Sistem Resirkulasi Dengan Filter Zeolit".

Freund, Y. and R. E. Schapire (1996). "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting".

Ho, W. T., H. W. Lim, and Y. H. Tay (2009). "Two-stage License Plate Detection using Gentle Adaboost and SIFT-SVM".

Papageorgiou, C. P., M. Oren, and T. Poggio (1998). "A General Framework for Object Detection".

Renno, J.-P., D. Makris, and G. A. Jones (2007). "Object Classification in Visual Surveillance Using Adaboost".

Rusydi, M. I. (2019). "Perancangan Mesin Penghitung Benih Ikan Otomatis untuk Membantu Kinerja Peternak Ikan".

Sunartono (2023). *Fantastis! Menteri KKP Sebut Nilai Pasar Ikan Capai Rp2.400 Triliun*. URL: https://news.harianjogja.com/read/2023/02/28/500/1127655/fantastis-menteri-kkp-sebut-nilai-pasar-ikan-capai-rp2400-triliun (visited on 08/08/2023).

Viola, P. and M. J. Jones (2004). "Robust Real-Time Face Detection".

Weber, B. (2006). "Generic Object Detection using AdaBoost".