BAB 2
TINJAUAN PUSTAKA

2.1 *Game Hidden Object*

Permainan bergenre *hidden object* merupakan permanin yang menggunakan kejelian mata untuk mengenali objek-objek. Pada permainan ini objek-objek tersebar dari sebuah objek gambar terletak pada grid. Untuk memainkan game ini, pemain mencari objek gambar dan menandai semua objek-objek tersebar yang tersembunyi di dalam objek gambar pada grid. Dalam permainan *hidden object,* daftar objek tersebar yang tersembunyi disediakan. Biasnya banyak objek yang terkait mudah bagi pemain untuk dicari. Objek tersebar dapat diatur dalam posisi horizontal,vertikal,diagonal atau terlihat sebagian. Semakin cepat pendeteksian objek, maka skor yang tinggi akan didapatkan. Criminal case, smootflife, Abyss: Wraith of Eden berikut adalah beberapa gameyang termasuk kedalam jenis game *hidden object* [5].

2.2 Kecerdasan *Buatan*

Kecerdasan buatan berasal dari bahasa Inggris "Artificial Intelligence" atau disingkat AI, yaitu intelligence adalah kata sifat yang berarti cerdas, sedangkan artificial artinya buatan. Kecerdasan yang dimaksud disini merajuk pada mesin yang mampu berpikir, menimbang tindakan yang diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia.

Teknik pemecahan masalah yang ada di dalam *Artificial Intelligence* dibagi menjadi 4 yaitu [10]:

a. *Searching*

   Searching merupakan teknik menyelesaikan masalah dengan cara merepresentasikan masalah ke dalam *state* dan ruang masalah serta menggunakan strategi pencarian untuk menemukan solusi.

b. *Reasoning*

   Reasoning yaitu mempresentasikan masalah ke dalam basis pengetahuan dan melakukan proses penalaran untuk menemukan solusi.

c. *Planning*

Planning adalah suatu metode penyelesaian masalah dengan cara memecah masalah ke dalam sub-sub masalah yang lebih kecil, menyelesaikan sub-sub masalah satu demi satu, kemudian menggabungkan solusi-solusi dari sub-sub masalah tersebut menjadi sebuah solusi lengkap dengan tetap mengingat dan menangani interaksi yang terdapat pada sub-sub masalah tersebut.

d. *Learning*

Learning merupakan teknik menyelesaikan masalah dengan mengotomatisasi mesin untuk menemukan aturan yang diharapkan bisa berlaku umum untuk data-data yang belum pernah diketahui.

2.3 Citra Digital

Citra adalah representasi, kemiripan dari suatu objek. Citra sebagai keluaran suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal-sinyal video atau bersifat digital yang dapat lansung disimpan dalam suatu media penyimpanan.

Citra digital adalah citra yang dapat diolah menggunakan komputer. Yang disimpan dalam sebuah memori komputer hanyalah angka-angka yang menunjukan besar intensitas pada masing-masing piksel. Karena citra digital berbentuk data numeris, maka citra digital dapat diolah mengunakan komputer [6].

2.3.1 Komponen Citra Digital

Setiap citra digital mempunyai beberapa karakteristik, diantaranya ukuran citra, resolusi dan format. Komputer dapat mengolah isyarat-isyarat elektronika yang merupakan kumpulan sinyal-sinyal biner yang mempunyai nilai o atau 1. Citra digital harus memiliki format yang sesuai agar dapat merepresentasikan suatu obyek pencitraan dalam bentuk kombinasi data biner. Format citra digital yang banyak di pakai umumnya adalah citra biner, skala keabuan (*grey scale*), warna dan warna berindeks [6].

2.3.2 Citra Warna

Citra warna yang memberikan rentang paling lebar adalah red (R), green (G) dan blue (B) dan biasa disebut dengan RGB. RGB adalah Warna lain dapat diperoleh dengan mencampurkan ketiga warna pokok tersebut dengan perbandingan tertentu. Setiap warna mempunyai intensitas sendiri dengan nilai maksimum 255 (8-bit). Misal warna kuning merupakan kombinasi warna merah dan hijau sehingga nilai RGB: 255 255 0 [6].

2.3.3 Tahap *Scaling*

Scaling merupakan proses merubah ukuran gambar digital atau menormalisasi ukuran gambar, Proses scaling sangatlah penting karena pada system pengenalan objek membutuhkan data objek gambar yang seragam ukurannya. Tahap scaling ini dapat dilakukan dengan menggunakan properti *width* dan *height* atau properti skala seperti scalex dan scaley .

Penskalaan citra (*scaling*) adalah sebuah operasi geometri yang memberikan efek memperbesar atau memperkecil ukuran citra *input* sesuai dengan variabel penskalaan citranya. Penskalaan citra (*scaling*) digunakan untuk memperbesar (*zoom-in*) atau memperkecil (*zoom-out*) citra. Rumus yang digunakan untuk penskalaan citra adalah rumus (2.1) :

$$x' = Sh\ x$$
$$y' = Sv\ y \qquad \text{.....(2.1)}$$

Keterangan :

Sh = faktor skala horizontal

Sv = faktor skala vertikal

2.3.4 Grayscaling

Grayscale adalah jenis gambar yang menangani gradiasi warna hitam dan putih, yang menghasilkan efek warna abu-abu. Pada gambar ini, warna dinyatakan dengan intensitas. Dalam hal ini intensitas berkisar antara 0 sampai 255. Nilai 0

menyatakan hitam dan nilai 255 menyatakan putih. Konversi gambar normal (RGB) ke gambar grayscale dilakukan dengan menggunakan rumus (2.2) [6].

$$Grayscale = (0.299 * \mathrm{R}) + (0.587 * \mathrm{G}) + (0.114 * \mathrm{B}) \ldots\ldots (2.2)$$

2.3.5 Tresholding

Tresholding digunakan untuk mengatur jumlah derajat keabuan yang ada pada citra.. Proses *tresholding* ini pada dasarnya adalah proses pengubahan kuantisasi pada citra, sehingga untuk melakukan *tresholding* dengan derajat keabuan dapat digunakan rumus (2.3) [14]:

$$\mathrm{g(x,y)} = \begin{cases} 1 & if \ \ f(x,y) > T \\ 0 & if \ \ f(x,y) \leq T \end{cases} \qquad (2.3)$$

Keterangan :

- Putih Jika level keabuannya >T
- Hitam jika level keabuannya <= T

Proses *tresholding* akan menghasilkan citra biner, yaitu citra yang memiliki dia nilai tingkat keabuan yaitu hitam dan putih. Secara umum proses *tresholding* citra *grayscale* untuk menghasilkan citra biner adalah sebagai berikut: Berikut ini contoh thresholding untuk mengubah citra grayscale menjadi 2 warna yaitu hitam dan putih

2.4 Computer Vision

Terminologi lain yang berkaitan erat dengan pengolahan citra digital adalah *computer vision* atau *machine computer*. Pada hakikatnya, *computer vision* mencoba meniru cara kerja visual manusia (*Human Vision*). *Human Vision* sesungguhnya sangat kompleks, manusia melihat objek dengan indera penglihatan (mata) kemudian objek citra diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti objek apa yang tampak dalam pandangan matanya. Hasil interpretasi ini mungkin digunakan untuk mengambil suatu keputusan [7].

Sebagaimana layaknya mata dan otak, *computer vision* adalah merupakan proses otomatis yang mengintegrasikan sejumlah besar proses untuk persepsi visual, seperti akuisisi citra, pengolahan citra, pengenalan dan membuat keputusan. *Computer vision* mencoba meniru cara kerja sistem visual manusia *(human vision)* yang sesungguhnya sangat kompleks, setelah data objek yang bersangkutan dimasukkan dalam bentuk citra (*image*) [7].

Proses-proses dalam *computer vision* dibagi dalam 3 (tiga) aktifitas :

1. Memperoleh atau mengakuisisi citra digital.
2. Melakukan teknik komputasi untuk memproses atau memodifikasi data citra.
3. Menganalisis dan menginterpretasi citra menggunakan hasil pemrosesan untuk tujuan tertentu, misalnya memandu robot, mengontrol peralatan, memantau manufaktur dan lain-lain.

Pengolahan citra merupakan proses awal (*preprocessing*) pada *computer vision*, sedangkan pengenalan pola merupakan proses untuk menginterpretasi citra. Teknik-teknik di dalam pengenalan pola memainkan peranan penting dalam *computer vision* untuk mengenali objek.

Pengolahan citra merupakan proses awal pada computer vision, pengenalan pola merupakan proses untuk menginterpretasi citra.

2.5 Pengenalan Pola

Pengenalan pola adalah mengelompokan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin (komputer). Tujuan pengelompokan adalah untuk mengenali suatu objek di dalam citra. Manusia bisa mengenali objek yang dilihatnya karena otak manusia telah belajar mengklasifikasi objek-objek di alam sehingga mampu membedakan suatu objek dengan objek lainnya. Kemampuan sistem visual manusia yang dicoba ditiru oleh mesin. Komputer menerima masukan berupa citra objek yang akan diidentifikasi, memproses citra tersebut dan memberikan keluaran berupa informasi atau deskripsi objek di dalam citra.[6].



Gambar 2.1 Pola pengenalan[8]

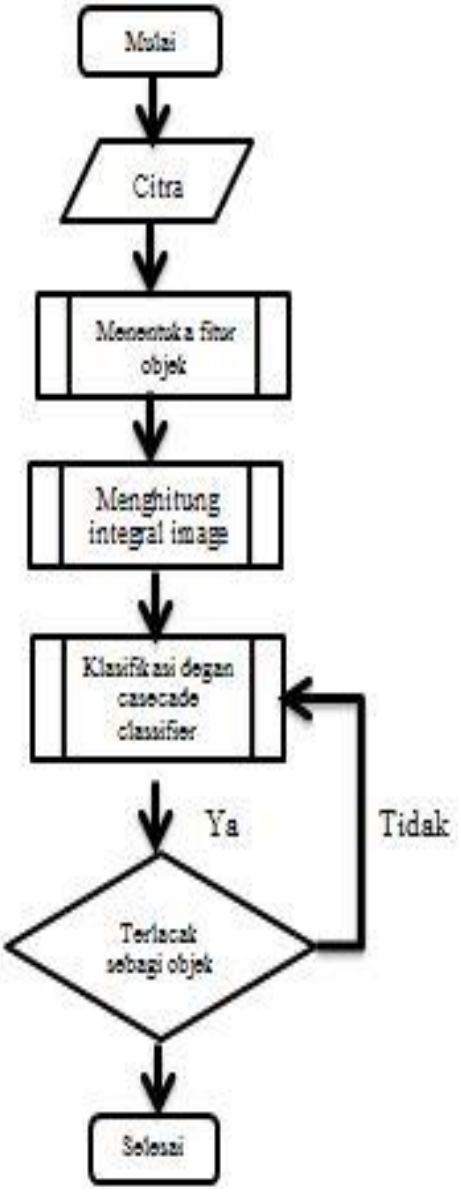
Contoh pengenalan pola misalnya citra pada gambar adalah citra mata yang digunakan sebagai data masukan untuk mengenali mata. Dengan menggunakan suatu algoritma pengenalan pola, diharapkan komputer dapat mengenali bahwa citra tersebut adalah mata.

2.6 *Haar Cascade Classifier*

Proses deteksi adanya citra objek dalam sebuah gambar pada *EmguCV*, menggunakan sebuah metode yang dipublikasikan oleh Paul Viola dan Michael Jones tahun 2001. Pada umumnya disebut dengan metode *Haar Classifier*. Metode ini menggunakan *statistical model* (*classifier*). Pendekatan untuk mendeteksi objek dalam gambar menggabungkan empat konsep utama:

1. Training data
2. Fitur segi empat sederhana yang disebut fitur Haar
3. Integral image untuk pendeteksian fitur secara cepat
4. Pengklasifikasi bertingkat (*Cascade classifier*) untuk menghubungkan banyak fitur secara efisien.

Haar Cascade Classifier merupakan suatu metode pengklasifikasian bertingkat, dimana input dari setiap tingkatan merupakan *output* dari tingkatan sebelumnya. Pada *classifier* tingkat pertama, yang menjadi inputan adalah seluruh citra, semua citra yang berhasil melewati *classifier* pertama akan dilanjutkan ke classifer ke dua, dan seterusnya. Apabila suatu citra berhasil melewati semua tingkat *classifier*, maka citra tersebut dinyatakan sebagai objek. Sedangkan untuk citra yang gagal melewati suatu *classifier* akan langsung dieliminasi dan dinyatakan sebagai bukan objek. Hal ini sangat mempercepat proses pengklasifikasian, karena jumlah inputan yang diterima di setiap *classifier* akan semakin berkurang. Berikut adalah alur proses metode *haar cascade classifier.*



.Gambar 2.2 metode *haar cascade classifier*

2.6.1 *Training* Data pada *Haar*

Metode ini memerlukan 2 tipe gambar objek dalam proses *training* yang dilakukan[3], yaitu :

1. Positive samples

Berisi gambar objek yang ingin di deteksi, apabila ingin mendeteksi objek maka positive samples ini berisi gambar objek, begitu juga objek lain yang ingin dikenali.

2. Negative samples

Berisi gambar objek selain objek yang ingin dikenali, umumnya berupa gambar background (tembok, pemandangan, lantai, dan gambar lainnya).

Training dari Haar menggunakan dua tipe sampel diatas.Informasi dari hasil training ini lalu dikonversi menjadi sebuah parameter model statistik.

2.6.2 Sistem Kerja Algoritma *Haar Cascade Classifier*

Algoritma Haar memakai metode statistikal dalam melakukan proses pengenalan objek. Metode Haar menggunakan *simple haar-like features* dan juga menggunakan *AdaBoost Classifier*[1]. *Cascade classifier* dirancang sedemikian rupa untuk meningkatkan tingkat pendeteksian dan mengurangi jumlah *false positive* (citra negatif yang dianggap sebagai citra positif). Setiap tingkatan *classifier* merupakan representasi hasil dari algoritma *boosting*. Jadi, disetiap tingkat *classifer* memiliki sejumlah *weak classifiers*. Setiap *weak classifier* merupakan nilai-nilai fitur Haar yang digunakan (jenis, ukuran, dan lokasi), nilai *threshold* terbaik untuk setiap fitur, serta nilai batasan setiap fitur tersebut.

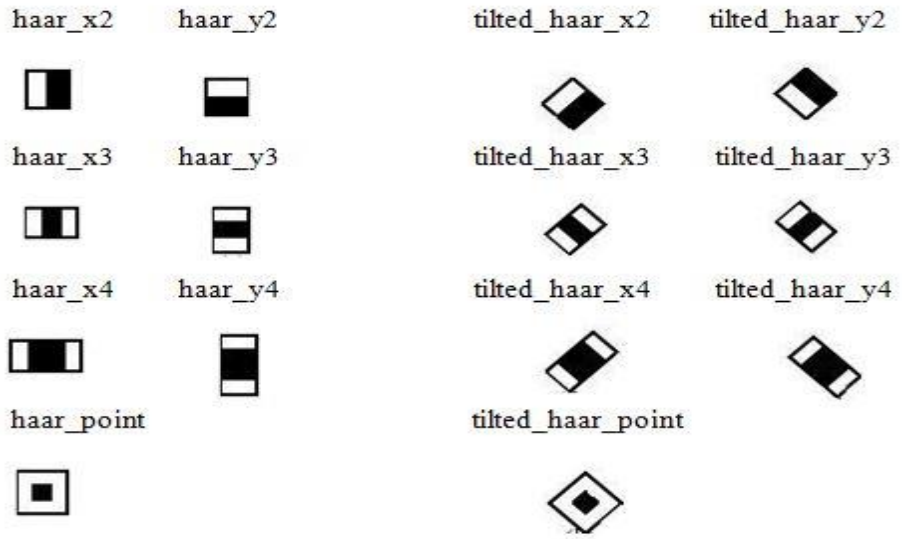
Semakin tinggi tingkat *classifer*, semakin banyak pula jumlah *weak classifier* yang ada. Hal ini mengakibatkan semakin sulitnya suatu citra untuk berhasil melewati tingkatan *classifier* tersebut, sehingga jumlah citra yang dieliminasi akan semakin banyak,dan jumlah citra yang berhasil lolos ke *classifier* tingkat selanjutnya akan semakin sedikit[1].

Classifier ini menggunakan gambar berukuran tetap. Cara kerja dari Haar dalam mendeteksi objek adalah menggunakan teknik *resize* pada gambar dan mencari apakah terdapat bagian dari gambar yang berbentuk seperti objek atau tidak.

Haar juga memiliki kemampuan untuk melakukan *scaling* sehingga dapat mendeteksi adanya objek yang berukuran lebih besar ataupun lebih kecil dari gambar pada *classifier*. Tiap *feature* dari *haar-like feature* didefinisikan pada bentuk dari *feature*, diantaranya koordinat dari feature dan juga ukuran dari *feature* tersebut.

2.6.3 *Haar Like Features*

Algoritma Haar menggunakan metode statistikal dalam melakukan pendeteksian objek. Tiap *feature* dari *Haar-like feature* didefinisikan pada bentuk dari *feature*, diantaranya koordinat dari *feature* dan juga ukuran dari *feature* tersebut. *Feature* yang akan di *training* adalah *feature* yang didapat dari proses *cropping.* Adapun macam-macam variasi *feature haar* :



Gambar 2.3 *Haar like feature* [3]

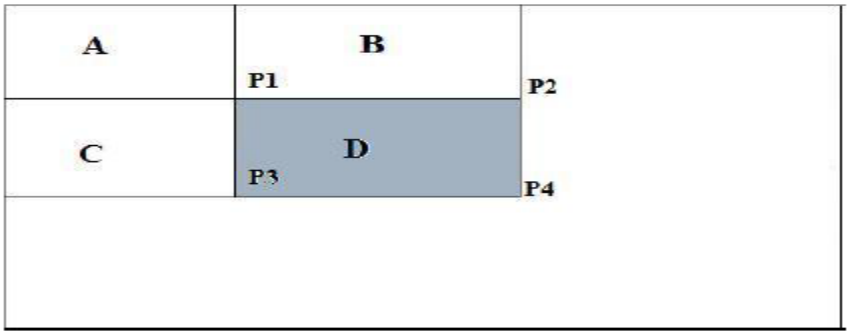
Tiap *feature* terbagi atas dua bagian, yaitu bagian berwarna putih dan bagian berwarna hitam. Penggunaan fitur dilakukan karena pemrosesan fitur berlangsung lebih cepat dibandingkan pemrosesan citra perpiksel sehingga cocok untuk diterapkan pada proses pendeteksian objek secara *real time.* Pada penelitian ini untuk menghitung nilai fitur Haar pada sebuah citra dan pada skala yang berbeda secara cepat menggunakan satu teknik yang disebut *integral image.*

Secara umum integrating berarti menggabungkan unit-unit kecil menjadi satu. Dalam hal ini unit-unit kecil adalah nilai piksel. Nilai integral suatu piksel adalah jumlah dari semua piksel diatas dan di kiri piksel tersebut. Seluruh citra dapat digabungkan dengan operasi nilai yang lebih sedikit tiap pikselnya. *Integral image* sangat membantu dalam perhitungan fitur *Haar-like*. Dengan

menggunakan *integral image*, perhitungan fitur *Haar-like* dapat dilakukan dengan sangat cepat [3].

Seluruh citra dapat digabungkan dengan operasi nilai yang lebih sedikit tiap pikselnya. *Integral image* sangat membantu dalam perhitungan fitur *Haar-like*. Dengan menggunakan *integral image*, perhitungan fitur *Haar-like* dapat dilakukan dengan sangat cepat.

Rumus integral image : $P(\mathrm{x},\mathrm{y}) = \sum_{x \le x, y \le y} i(x,y)$.....(2.1)



Gambar 2.4 *Integral image* [3]

$P_1 = A, P_2 = A + B, P_3 = A + C, P_4 = A + B + C + D$

$P_1 + P_4 - P_2 - P_3 = A + A + B + C + D - A - B - A - C = D$

Contoh perhitungan nilai fitur menggunakan *integral image* adalah sebagai berikut[14]:

s(x,y)=s(x,y)+s(x-1,y)+s(x,y-1)-s(x-1,y-1) [2.5]

1	2	3
2	3	4
4	5	1

1	2	3
2	3	4
4	5	1

1	1+2	1+2+3
1+2	1+2+2+3	1+2+3+2+3+4
1+2+4	1+2+3+4+5	1+2+3+2+3+4+4+5

Maka hasil perhitungannya:

1	3	6
3	8	15
7	17	25

f(x) = Sumblack rectangle (pixel gray level) – Sumwhite rectangle (pixel gray level)

= |(6+0-(0+0))+(25+3-(15+0))-(15+1-(6+0))|

= |6+13-10|

= 9

2.7 Pemrograman Berorientasi Objek

Perkembangan gaya pemrograman mencapai gaya pemrograman orientasi objek setelah era pemrograman terstruktur. Pemrograman berorientasi objek menggantikan pemrograman terstruktur karena mempunyai banyak keunggulan dalam menangani proyek yang luar biasa kompleks. Pemrograman menggunakan bahasa orientasi objek menawarkan fleksibilitas, keunggulan dan kemudahan perawatan [2].

Dengan pemrograman berorientasi objek, para pengembang menciptakan blok-blok kode, yang dinamakan objek [2]. Objek-objek ini kemudian digunakan oleh berbagai aplikasi. Suatu saat jika objek mebutuhkan perubahan, para pengembang hanya melakukan perubahan sekali saja dan kemudian ia dengan mudah akan mewariskannya ke objek-objek lain yang menjadi turunannya. Dengan kepraktisannya, perusahaan-perusahaan perangkat lunak saat ini berusaha mengadopsi teknologi baru ini dan mengintegrasikannya pada aplikasi-aplikasi yang sudah ada. Kenyataannya, kebanyakan aplikasi-aplikasi modern saat ini berorientasi objek.

Paradigma berorientasi objek adalah cara yang berbeda dalam memandang aplikasi-aplikasi [2]. Dengan pendekatan berorientasi objek, para pengembang membagi aplikasi-aplikasi besar menjadi objek-objek, yang mandiri satu terhadap lainnya. Pengembang kemudian mengembangkan aplikasi dengan membuat objek- objek itu saling mengirim pesan (*message*) dan bekerja-sama. Dimbil suatu contoh. Dengan pemrograman terstruktur, katakanlah ingin membuat form dengan *listbox* di dalamnya. Untuk hal ini, maka harus menulis kode-kode yang jumlahnya sangat banyak untuk membuat form dan *listbox* itu sendiri dan kemudian membuat kode- kode untuk tombol OK. Dengan pemrograman berorientasi objek, maka dengan sederhana hanya perlu menggunakan objek-objek yang telah ada sebelumnya: form, *listbox*, dan tombol OK.

Maka, alih-alih membuat segalanya dari awal dengan pemrograman terstruktur, dengan pemrograman berorientasi objek tinggal menggunakan objek-objek yang sudah ada dan kemudian berfokus pada ‘apa yang unik’ untuk aplikasi tertentu yang sedang dikembangkan. “*Do not reinventing the wheel!*”,

kata banyak ahli. Maka, keunggulan utama mengembangkan objek-objek sekali saja dan menggunakannya lagi (*reusable*) berulang-ulang. Dengan pendekatan berorientasi objek, difokuskan pada informasi dan perilaku yang dimiliki suatu objek sehingga kemudian pengembang dapat mengembangkan sistem/perangkat lunak yang fleksibel dalam menghadapi perubahan-perubahan informasi dan/atau perilaku yang dituntut pengguna.

2.8 UML (*Unified Modelling Language*)

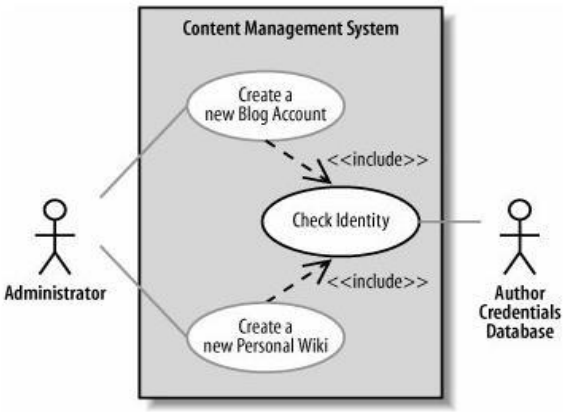
UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah untuk dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka yang lain [8]. UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, *Object Modelling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE) [8]. Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan design ke dalam empat tahapan iteratif, yaitu identifikasi kelas-kelas dan objek- objek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian interface dan implementasi. Keunggulan metode Booch adalah pada detail kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi ini adalah analisis, design sistem, design objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari Jacobson lebih memberi penekanan pada *use case*. OOSE memiliki tiga tahapan yaitu membuat model *requirement* dan analisis, design dan implementasi, dan model pengujian (*test model*). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak. Dengan UML, metode Booch, OMT dan OOSE

digabungkan dengan membuang elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam daripada metode lainnya.

2.8.1 *Use Case Diagram*

Use Case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut skenario. Setiap skenario mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem lain, perangkat keras atau urutan waktu. Dengan demikian secara singkat biasa dikatakan *use case* adalah serangkaian scenario yang digabungkan bersama-sama oleh tujuan umum pengguna [2].

Berikut ini adalah contoh *Use Case Diagram*:



Gambar 2.6 *Use Case Diagram*

2.8.2 *Use Case Scenario*

Sebuah diagram yang menunjukkan kasus penggunaan dan aktor dapat menjadi titik awal yang bagus, tetapi tidak memberikan cukup detail untuk desainer sistem untuk benar-benar memahami persis bagaimana kekhawatiran sistem akan dipenuhi. Use Case Skenario adalah cara terbaik untuk mengungkapkan informasi penting ini dalam bentuk penggunaan description setiap kasus berbasis teks harus disertai dengan satu.[2]

Berikut ini adalah contoh *Use Case Scenario*:

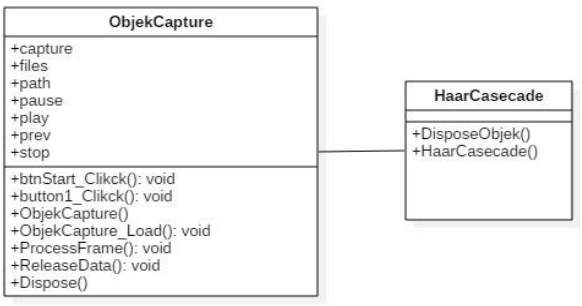
Use case name	**Create a new Blog Account**	
Related Requirements	Requirement A.1.	
Goal In Context	A new or existing author requests a new blog account from the Administrator.	
Preconditions	The system is limited to recognized authors and so the author needs to have appropriate proof of identity.	
Successful End Condition	A new blog account is created for the author.	
Failed End Condition	The application for a new blog account is rejected.	
Primary Actors	Administrator.	
	Secondary Actors	Author Credentials Database.
	Trigger	The Administrator asks the CMS to create a new blog account.
Main Flow	Step	Action
	1	The Administrator asks the system to create a new blog account.
	2	The Administrator selects an account type.
	3	The Administrator enters the author's details.
	4	The author's details are verified using the Author Credentials Database.
	5	The new blog account is created.
	6	A summary of the new blog account's details are emailed to the author.
Extensions	Step	Branching Action
	4.1	The Author Credentials Database does not verify the author's details.
	4.2	The author's new blog account application is rejected.

Gambar 2.7 *Use Case Scenario* [2]

2.8.3 *Class Diagram*

Class Diagram adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem/perangkat lunak yang sedang kita kembangkan. Diagram kelas memberikan gambaran/diagram statis tentang sistem/perangkat lunak dan relasi-relasi yang ada di dalamnya.[2]

Berikut ini adalah contoh *Class Diagram*:

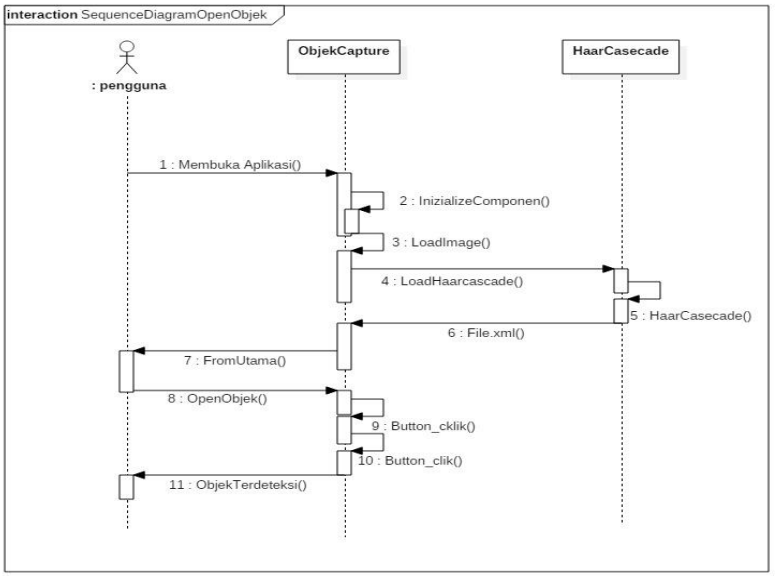


Gambar 2.8 *Class Diagram*

2.8.4 *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh objek dan *message* (pesan) yang diletakkan diantara objek-objek ini dalam *use case*.[2]

Berikut ini adalah contoh *Sequence Diagram*:



Gambar 2.9 *Sequence Diagram*

2.8.5 *Activity Diagram*

Activity Diagram digunakan untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya adalah activity diagram mendukung perilaku paralel sedangkan *flowchart* tidak bias[2].

2.9 Pengujian

Ada dua macam pengujian yang akan dilakukan, yaitu pengujian performansi dan black box testing. Pengujian performansi yaitu dengan melakukan percobaan-percobaan dalam kondisi-kondisi tertentu seperti pengaruh posisi objek tersebar pada saat pendeteksian, Pengaruh objek pada saat terlihat sebagian, dan pengaruh ketika objek berbeda-beda. Pengujian pada black box testing yaitu menemukan kesalahan yang terdapat pada program.

2.9.1 Pengujian Performasi

Pengujian performansi dilakukan dengan serangkaian percobaan-percobaan dalam kondisi-kondisi tertentu yang dapat mempengaruhi keefektifan kinerja sistem pendeteksian objek tersebar.

2.9.2 Pengujian Black Box

Black Box Testing bertujuan untuk menunjukkan fungsi-fungsi perangkat lunak tentang cara beroperasinya, apakah masukan data keluaran telah berjalan sebagaimana yang diharapkan dan apakah informasi yang disimpan secara eksternal selalu dijaga kemutakhirannya. Tehnik pengujian black-box berfokus pada domain informasi dari perangkat lunak, dengan melakukan test case dengan menpartisi domain input dari suatu program dengan cara yang memberikan cakupan pengujian yang mendalam. Metode pengujian graph-based mengeksplorasi hubungan antara dan tingkah laku objek-objek program. Partisi ekivalensi membagi domain input ke dalam kelas data yang mungkin untuk melakukan fungsi perangkat lunak tertentu. Analisis nilai batas memeriksaa kemampuan program untuk menangani data pada batas yang dapat diterima. Metode pengujian yang terspesialisasi meliputi sejumlah luas kemampuan perangkat lunak dan area aplikasi. GUI, arsitektur client/ server, dokumentasi dan fasilitas help dan sistem real time masing-masing membutuhkan pedoman dan tehnik khusus untuk pengujian perangkat lunak [12].

2.9.3 Confusion Matrix

Confusion Matrix (Kohavi dan Provost, 1998) berisi informasi tentang klasifikasi aktual dan yang telah diprediksi yang dilakukan oleh sistem klasifikasi. Kinerja sistem tersebut umumnya dievaluasi dengan menggunakan data dalam matriks. Tabel berikut menunjukkan *Confusion Matrix* untuk klasifikasi dua kelas [13].

***Tabel 2. 1 Tabel* Confusion Matrix**

		Predicted Class	
		Positif	**Negatif**
Actual Class	**Positif**	True Positives	False Positive
	Negatif	False Negative	True Negative

True positives adalah jumlah *record* positif yang diklasifikasikan sebagai positif, *false positives* adalah jumlah *record* positif yang diklasifikasikan sebagai

negatif, *false negatives* adalah jumlah *record* negatif yang diklasifikasikan sebagai positif, *true negatives* adalah jumlah *record* negatif yang diklasifikasikan sebagai negatif. Setiap kolom dari *Confusion Matrix* merupakan contoh di kelas yang telah diprediksi, sedangkan setiap baris mewakili contoh di kelas yang sebenarnya. Setelah didapat *true positives, false positives, true negatives* dan *false negatives*, selanjutnya hitung nilai *precision* dan akurasinya. *Precision* adalah ukuran terhadap suatu kelas yang telah diprediksi. Berikut Rumus dari *precision* dan akurasi.

$$Akurasi = \frac{TP + TN}{TP + FP + TN + FN} \qquad (2.6)$$

$$Precision = \frac{TP}{TP + FP} \qquad (2.7)$$

Dimana:

TP	:	Jumlah *True Positive*
FP	:	Jumlah *False Positive*
TN	:	Jumlah *True Negative*
FN	:	Jumlah *False Negative*

2.10 Software Pendukung

Pada softwaere pendukung yang diperlukan alat–alat berupa perangkat lunak atau *library* yang akan digunakan. Berikut ini merupakan alat yang akan digunakan pada proses pembangunan perangkat lunak:

2.10.1 Bahasa Pemrograman C#

Bahasa C# adalah sebuah bahasa pemrograman modern yang bersifat general-purpose, berorientasi objek, yang dapat digunakan untuk membuat program di atas arsitektur Microsoft .NET Framework. Bahasa C# ini memiliki kemiripan dengan bahasa Java, C dan C++ .

Bahasa pemrograman ini dikembangkan oleh sebuah tim pengembang di Microsoft yang dipimpin oleh Anders Hejlsberg, seorang yang telah lama malang

melintang di dunia pengembangan bahasa pemrograman karena memang ialah yang membuat Borland Turbo Pascal, Borland Delphi, dan juga Microsoft J++.

Kini, C# telah distandarisasi oleh European Computer Manufacturer Association (ECMA) dan juga International Organization for Standardization (ISO) yang mendukung beberapa fitur baru semacam Language Integrated Query (LINQ) dan lain-lainnya.

2.10.2 Microsoft Visual Studio 2013

Merupakan sebuah perangkat lunak lengkap (suite) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi *console*, aplikasi Windows, ataupun aplikasi Web. *Visual Studio* mencakup kompiler, SDK, Integrated Development Environment (IDE), dan dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket *Visual Studio* antara lain Visual C++, Visual C#, *Visual Basic*, *Visual Basic*.NET, *Visual InterDev*, *Visual* J++, *Visual* J#, *Visual FoxPro*, dan *Visual SourceSafe*. *Microsoft Visual* Studio dapat digunakan untuk mengembangkan aplikasi dalam native *code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed code* (dalam bentuk Microsoft Intermediate Language diatas. NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework). Microsoft kini merilis Microsoft Visual Studio 2010 dan Microsoft .NET Framework 4.0. Dua tool yang paling banyak digunakan untuk mengembangkan program diatas Windows, Windows Mobile, Web (ASP.NET), Silverlight, dan beberapa platform lainnya.