**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**USULAN TUGAS AKHIR**

# **IDENTITAS PENGUSUL**

**Nama** : **Suryawan Bimantoro**

**NRP** : **5107 100 007**

Dosen Wali : Ir. FX Arunanto

1. **JUDUL TUGAS AKHIR**

***Pembuatan Alat Kemudi dengan menggunakan Open Computer Vision untuk Spaceship pada Game Orbiter Space Flight Simulator.***

1. **URAIAN SINGKAT**

Open Computer Vision (OpenCV) adalah sebuah *library* yang mampu melakukan analisa citra, baik dari kamera ataupun citra biasa secara *real-time*.

Orbiter Space Flight Simulator adalah sebuah game simulasi pesawat luar angkasa. Game ini *free*, namun tidak berupa game *open source*. Walau demikian game ini dapat dilakukan modifikasi dengan menambahkan *plug-in* menggunakan Orbiter SDK yang tersedia langsung pada file instalasinya.

Orbiter memiliki masalah pada proses interaksinya dengan user yang menggunakan *keyboard*. Cara pengemudian dengan *keyboard* ini sangat menyulitkan dari sisi konfigurasi yang tidak dapat diubah hingga fungsi *keyboard* yang sangat banyak sehingga sulit untuk diingat. Karena masalah ini maka perlu dilakukan perubahan, yaitu dengan menggunakan alat kemudi yang dapat dicetak langsung, murah, lebih fleksibel, serta lebih responsif.

1. **LATAR BELAKANG**

Orbiter Space Flight Simulator merupakan sebuah game simulasi nyata pesawat luar angkasa. Sebuah game *free* namun bukan *open source* yang dapat diberi tambahan (*plug-in* atau *add on*). Penambahan tersebut dapat dilakukan dengan mengkompilasi proyek yang menggunakan *library* Orbiter SDK yang tersedia pada instalasi Orbiter Space Flight Simulator dalam bentuk \*.dll (*Dynamic Library*). Orbiter SDK sendiri adalah *library* yang disediakan Orbiter untuk melakukan implementasi fungsi-fungsi yang dapat dijalankan pada simulasi game Orbiter Space Flight Simulator.

Open Computer Vision (OpenCV) adalah sebuah *library* yang digunakan untuk menganalisa citra, guna mengenali isi dari citra tersebut. OpenCV mampu menganalisa citra baik dalam bentuk *single frame* (file citra) atau *multi frame* (citra dalam video).

Orbiter Space Flight Simulator memiliki beberapa kekurangan, diantaranya adalah konfigurasi *keyboard* yang digunakan sebagai alat kemudi. Alat kemudi yang disediakan sangat merepotkan, khususnya untuk para pengguna laptop yang kebanyakan tidak memiliki set tombol *numpad* pada *keyboard*-nya. Di sisi lain, konfigurasi tombol tidak bisa diubah secara visual, dan cenderung lebih mempersulit pemakai karena tidak disediakan petunjuk penggunaan *keyboard* sebagai alat kemudi.

Pada tugas akhir ini akan dilakukan penambahan untuk mengatasi kesulitan dalam mengemudikan pesawat luar angkasa dengan menggunakan keyboard. Penggabungan *library* Orbiter SDK dengan OpenCV bisa menjadi solusi untuk membentuk sebuah alat kemudi yang responsif. Orbiter SDK akan bertindak sebagai penghubung dengan sistem Orbiter Space Flight Simulator dan OpenCV akan bertindak sebagai penganalisa citra cetakan panel kemudi yang ditangkap melalui kamera yang telah terintegrasi dengan Windows.

1. **RUMUSAN MASALAH**

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana menggunakan *library* Orbiter SDK untuk membangun objek pesawat luar angkasa.
2. Bagaimana menggabungkan *library* Orbiter SDK dan OpenCV untuk membuat sebuah alat kemudi yang dapat berinteraksi dengan pengguna.
3. Bagaimana membuat proses interaksi alat kemudi dengan pengguna dapat dilakukan secara *real-time*.

1. **BATASAN MASALAH**

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, diantaranya sebagai berikut:

* + - 1. Tombol yang digunakan nantinya adalah operasi mesin utama, operasi mesin hover, operasi mesin mundur, anggukan, gulingan, gelengan, operasi pergerakan linier (maju, mundur, naik, turun, kiri, kanan), dan mode navigasi.
      2. Panel dicetak pada kertas berwarna putih.
      3. Proses interaksi dilakukan dengan menutup salah satu cetakan tombol dengan kertas yang berwarna putih dengan ukuran yang sama besar dengan ukuran cetakan tombol.
      4. Proses deteksi objek pada OpenCV menggunakan data hasil training.
      5. Tugas akhir ini menggunakan skenario yang sudah disediakan pada game Orbiter Space Flight Simulator

1. **TUJUAN TUGAS AKHIR**

Pada tugas akhir ini memiliki tujuan sebagai berikut:

* + - 1. Menerapkan *library* Orbiter SDK untuk membuat objek pesawat luar angkasa.
      2. Menerapkan *library* Orbiter SDK dan OpenCV untuk membuat sebuah alat kemudi.
      3. Membuat proses interaksi alat kemudi dengan pengguna dapat dilakukan secara *real-time*.

1. **MANFAAT TUGAS AKHIR**

Manfaat yang diharapkan dari tugas akhir ini adalah untuk menciptakan sebuah alat kemudi yang responsif. Selain itu, diharapkan juga penulis dapat menciptakan.

1. **RINGKASAN TUGAS AKHIR**
   1. **Orbiter SDK**

Orbiter SDK adalah sebuah *library* yang digunakan untuk membuat tambahan *plug-in* yang dapat digunakan oleh sistem Orbit Space Flight Simulator dalam bentuk \*.dll (*Dynamic Library*). Ruang lingkup pemrograman yang disediakan oleh Orbiter SDK akan memungkinkan pengembang modul pihak ketiga untuk melakukan peningkatan fungsionalitas sistem Orbiter Space Flight Simulator[1]. Contoh-contoh modul yang dapat dikembangkan adalah[1]:

* 1. Pesawat luar angkasa yang sudah termasuk *custom flight model*, panel instrumen, dll.
  2. Objek antariksa (planet, bintang, dll) yang sudah termasuk kode atmosfer.
  3. Intrumen tambahan, seperti mode MFD (*Multifunctional Display*).
  4. Modul global, seperti *networking*, suara, alat kemudi, dll.
  5. **OpenCV**

OpenCV atau Open Computer Vision adalah sebuah *library* yang digunakan untuk melakukan sekitar 500 fungsi implementasi visi komputer, pemrosesan citra, dan beberapa penggunaan umum dari algoritma numerik[2].

Mengapa digunakan OpenCV? Saat ini tidak ada API standar (seperti OpenGL dan DirectX pada pemrosesan grafik komputer, atau OpenSSL pada pemrosesan kriptografi), kebanyakan dari perangkat lunak visi komputer terdiri atas tiga jenis, yaitu[2]:

1. Kode riset (lambat, tidak stabil, dan tipe yang independen/tidak kompatibel untuk setiap *library*/*toolbox*).
2. Alat-alat komersil yang dijual di pasaran (seperti Matlab, Halcon, dll) cenderung memiliki harga yang mahal.
3. Spesialisasi solusi dengan perangkat keras.

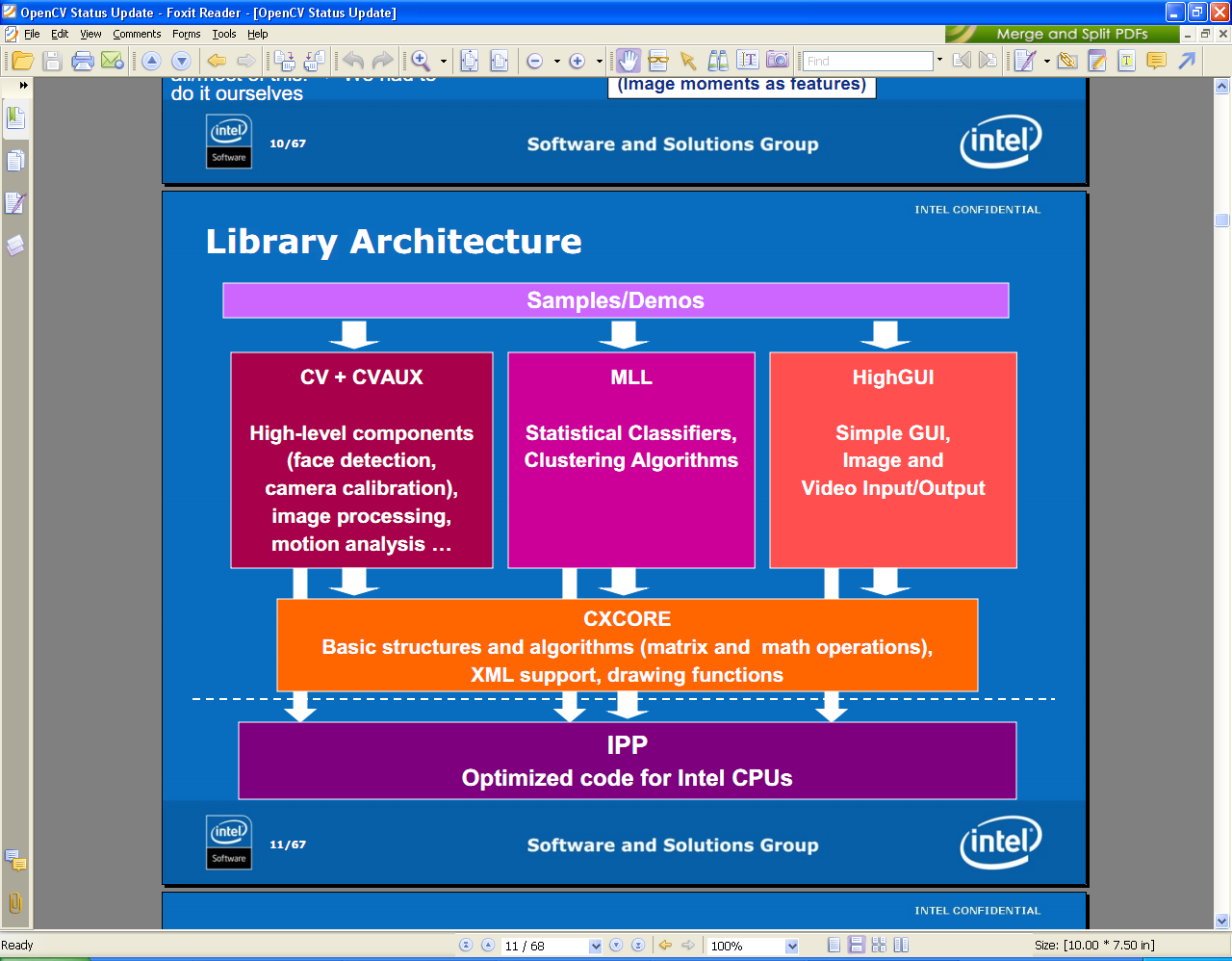
Selain itu dengan menggunakan *library* standar ini (OpenCV) dapat menyederhanakan pengembangan sebuah sistem baru dengan lebih mudah. Apalagi dengan tersedianya fitur optimasi spesial untuk arsitektur prosesor yang berbasiskan Intel[2].

1. Membuat model penggunaan baru dengan performa real-time untuk algoritma yang sedikit sulit (seperti deteksi wajah).
2. Membuat platform Intel menjadi lebih atraktif untuk pengembang visi komputer.

Arsitektur yang digunakan oleh OpenCV adalah dengan membagi proses ke dalam tiga bagian, yaitu:

1. CV+CVAUX yang bertujuan untuk memproses komponen tingkat tinggi (deteksi objek, kalibrasi kamera), pemrosesan citra, analisa pergerakan, dll.
2. MLL yang bertujuan untuk memproses klasifikasi statistik dan algoritma pengelompokan.
3. HighGUI yang bertujuan untuk memproses GUI sederhana serta I/O citra atau video.

Dari ketiga bagian tersebut, selanjutnya akan digabungkan menjadi satu ke dalam satu proses, yaitu CXCORE. CXCORE akan memproses stuktur dan algoritma dasar (seperti matriks dan operasi matematika), dukungan XML, fungsi menggambar, dll. Jika sistem bekerja pada prosesor berbasiskan Intel, IPP akan ikut berperan serta dalam proses untuk mengoptimasikan implementasi kode yang dibuat. Untuk memperjelas mengenai arsitektur OpenCV dapat dilihat pada diagram di bawah ini.



Gambar – Arsitektur *library* OpenCV

* 1. **Implementasi yang Sudah Ada**

Contoh implementasi Orbiter SDK yang sudah pernah ada dan berhubungan dengan tugas akhir ini adalah Remote Vessel Control. Remote Vessel Control bekerja melalui metode *plug-in*. Pada implementasi ini memperbolehkan pengguna untuk mengemudikan pesawat luar angkasa melalui jendela *dialog box*. Remote Vessel Control adalah contoh implementasi yang sangat sederhana. Hanya bekerja dengan menangkap pesawat luar angkasa mana yang sedang aktif atau sedang digunakan oleh pengguna. Selanjutnya interaksi dilakukan dengan menekan tombol-tombol yang ada di *dialog box*. Setiap tombol memiliki aksi tertentu yang akan mengaktifkan fungsi-fungsi kemudi tertentu pada pesawat luar angkasa.

Sedangkan salah satu contoh implementasi OpenCV yang sudah pernah ada dan berhubungan dengan tugas akhir ini adalah pendeteksian objek dengan menggunakan algoritma Ferns. Implementasi ini mampu mendeteksi objek dalam waktu 80 ms per citra yang ditangkap. Sementara *frame rate* game ini berbeda pada masing-masing hardware yang digunakan. Namun bisa dianggap sama dengan *frame* video, yaitu 24 *frame* per detik atau sekitar 40 ms per *frame*. Namun saat me-*load* data xml yang berisi data citra yang dianggap penting, justru memakan banyak waktu. Ada kemungkinan saat proses *loading* *plug-in* pada game Orbiter Flight Simulator akan terjadi kesalahan.

* 1. **Desain Plug-in Alat Kemudi**

Pada tugas akhir ini akan dibuat sebuah *plug-in* alat kemudi yang menggunakan Orbiter SDK sebagai penghubung ke sistem Orbiter Space Flight Simulator dan menggunakan OpenCV sebagai pemrosesan citra cetakan panel kemudi yang ditangkap oleh kamera. Proses tersebut akan terbagi menjadi beberapa bagian, yaitu:

1. *Loading* data xml tombol pada panel kemudi yang dilakukan di awal sesaat setelah *plug-in* diaktifkan.
2. Mendapatkan objek pesawat luar angkasa yang sedang aktif (*Get Shuttle*).
3. Selama simulasi (*In Simulation*) akan dilakukan proses sebagai berikut secara berulang-ulang, yaitu:
   * + Menganalisa hasil tangkapan kamera.
     + Mengimplementasikannya pada pesawat luar angkasa yang sedang aktif.
4. Membersihkan seluruh objek yang digunakan *plug-in* (*Destroy*)*.*
   * 1. *Loading Data*

Seperti dijelaskan sebelumnya, *plug-in* alat kemudi ini berawal dengan proses *loading* data XML yang berisi data citra tombol pada panel kemudi yang dianggap penting. Proses ini dilakukan guna melakukan *training* pada sistem untuk mengenali objek-objek yang ditangkap kamera. Jika proses *training* tidak dilakukan, proses analisa akan memakan banyak waktu sehingga kurang responsif. Namun, pada contoh implementasi yang sudah ada, kelemahan justru terjadi saat proses ini. Sehingga, nantinya akan dibentuk sebuah citra tombol yang sesederhana mungkin, namun dapat membedakan tombol satu dengan tombol lainnya. Sehingga, proses *loading* data XML tidak akan memakan banyak waktu dan tentu saja meningkatkan responsivitas *plug-in*.

* + 1. *Get Shuttle*

Setelah proses *loading* data xml dapat diselesaikan, *plug-in* ini diharuskan untuk membuat objek pesawat luar angkasa yang sedang aktif digunakan oleh pengguna. Proses ini perlu dilakukan untuk memperbolehkan *plug-in* melakukan interaksi dengan sistem. Dengan kata lain, proses ini adalah pintu depan untuk dapat memasuki sistem Orbiter Space Flight Simulator. Setelah proses ini dilakukan, nantinya setiap tindakan pada panel cetakan yang dilakukan oleh pengguna akan dapat langsung diimplementasikan ke pesawat luar angkasa yang sedang aktif. Karena pada Orbiter SDK sudah berbasiskan *Object Oriented Programming*, maka setiap aksi nantinya dapat dilakukan dengan mengakses objek yang terbentuk dari proses ini.

* + 1. *In Simulation*

Kemudian selama simulasi, melalui *library* OpenCV akan menganalisa citra dari kamera dan mengimplementasikan hasil analisa kamera ke pesawat luar angkasa yang sedang aktif tersebut. Interaksi antara pengguna yang terjadi pada proses simulasi ini dilakukan dengan menutup salah satu tombol yang memiliki satu jenis grup operasi dengan menggunakan kertas berwarna putih. Setiap tombol yang tidak ada (tidak terlihat di kamera), menunjukkan bahwa tombol tersebut adalah tombol yang sedang aktif. Aksi pada pesawat luar angkasa akan menggunakan objek yang telah dibuat pada proses *Get Shuttle*.

* + 1. *Destroy Object*

Jika proses simulasi telah berakhir, *plug-in* diharuskan membersihkan seluruh sistem memori yang digunakan, seperti objek pesawat luar angkasa yang terakhir diakses, data training sistem, dan memori penggunaan tangkapan citra kamera serta menon-aktifkan penggunaan kamera. Apabila proses ini tidak dilakukan, sistem Orbiter tidak akan bisa kembali ke *launch pad*, karena saat proses kembali ke *launch pad*, sistem Orbiter akan mengecek apakah seluruh resource memori yang digunakan selama simulasi telah dibersihkan seluruhnya. Pada versi terakhir Orbiter ini (Orbiter 2010), Orbiter tidak mampu melakukan pembersihan secara otomatis. Jika terdapat sisa sedikit saja, Obiter akan langsung *crash*, dan sistem Orbiter akan langsung ditutup.

Pada *flowchart* di bawah ini akan dijelaskan bagaimana proses *plug-in* ini bekerja. Mulai dari awal *plug-in* diaktifkan, hingga simulasi berakhir. Namun, pada *flowchart* ini akan dijelaskan bagian kulit terluar dari proses utama yang harus dikerjakan oleh *plug-in*. Pada setiap prosesnya akan diperlihatkan secara ringkas apa saja yang dilakukan *plug-in* selama game Orbiter berjalan. Perlu diperhatikan juga, Orbiter tidak akan melakukan *loading* secara otomatis setiap modul atau *plug-in* yang sebelumnya telah diaktifkan oleh pengguna. Namun, pengguna perlu membuka fungsi *loading* modul terlebih dahulu. Setelah proses *loading* modul dilakukan, barulah modul ini dapat bekerja.

Gambar – Flowchart Plug-in

In Simulation?

END

T

F

Mengambil data XML panel

Mengambil objek *Spaceship*

Analisa input dari kamera

Implemantasi hasil analisa input ke *Spaceship*

Membersihkan seluruh objek *plug-in*

*Plug-in* membutuhkan sebuah cetakan panel kemudi yang memiliki gambar tombol yang sederhana, namun mudah dibedakan antara satu tombol dengan tombol lainnya. Tombol tersebut tidak boleh memiliki kesamaan, karena proses analisa akan terganggu jika terdapat kesamaan pada tombol-tombol tersebut. Penulis telah merancang sebuah ide yang dapat menampung kebutuhan tersebut. Panel cetakan tersebut akan terlihat seperti pada gambar di bawah ini.

Gambar – Contoh cetakan panel kemudi

**H+**

**H0**

**H-**

**M+**

**M0**

**M-**

**R+**

**R0**

**R-**

**PU**

**BL**

**YL**

**YR**

**BR**

**PD**

**LU**

**LL**

**LR**

**LD**

**LF**

**LB**

**N+**

**N-**

**HL**

**HA**

**PG**

**RG**

**X2**

**X1**

**WU**

**WD**

Masing-masing tombol pada contoh gambar panel cetakan tersebut memiliki fungsinya masing-masing, seperti dijelaskan dibawah ini:

* + - 1. Operasi mesin utama (M+, M0, dan M-) adalah satu set tombol yang tidak diperbolehkan untuk ditutup secara bersamaan. Fungsinya adalah untuk menambah, menghentikan, dan mengurangi dorongan dari mesin utama.
      2. Operasi mesin *hover*/naik (H+, H0, dan H-) adalah satu set tombol yang tidak diperbolehkan untuk ditutup secara bersamaan. Fungsinya adalah untuk menambah, menghentikan, dan mengurangi dorongan dari mesin hover. Perlu diperhatikan, bahwa tidak semua pesawat luar angkasa memiliki fitur mesin *hover*.
      3. Operasi mesin *retro*/mundur (R+, R0, dan R-) adalah satu set tombol yang tidak diperbolehkan untuk ditutup secara bersamaan. Fungsinya adalah untuk menambah, menghentikan, dan mengurangi dorongan dari mesin *retro*. Perlu diperhatikan, bahwa tidak semua pesawat luar angkasa memiliki fitur mesin *retro*.
      4. Operasi pergerakan rotasi (PU, PD, BL, BR, YL, YR, dan X1) adalah satu set tombol yang tidak diperbolehkan untuk ditutup bersamaan pada fungsi yang berlawanan, seperti PU dengan PD, BL dengan BR, dan YL dengan YR. Fungsi set tombol ini adalah untuk mengangguk ke atas dan ke bawah, mengguling ke kiri dan ke kanan, dan menggeleng ke kiri dan ke kanan. Khusus X1 bisa langsung ditutup bersamaan dengan tombol lain pada set ini, fungsinya adalah untuk menghentikan rotasi yang terjadi pada pesawat luar angkasa.
      5. Operasi pergerakan linier (LF, LB, LU, LD, LL, dan LR) adalah satu set tombol yang tidak diperbolehkan untuk ditutup bersamaan pada fungsi yang berlawanan, seperti LF dengan LB, LU dengan LD, dan LL dengan LR. Fungsi set tombol ini adalah untuk bergerak linier ke depan dan ke belakang, ke atas dan ke bawah, serta ke kiri dan ke kanan.
      6. Operasi mode navigasi (N+, N-, HL, HA, PG, RG, X2, WU, dan WD) adalah satu set tombol yang boleh ditutup bersamaan, kecuali untuk WU dan WD tidak diperbolehkan untuk ditutup bersamaan. Fungsi tombol pada operasi ini adalah untuk mengarahkan pesawat ke orbit normal, mengarahkan pesawat ke orbit abnormal, merotasikan pesawat ke level horizon, menjaga ketinggian pesawat, merotasikan pesawat ke arah kecepatan positif menuju objek yang dituju, merotasikan pesawat ke arah kecepatan negatif menuju objek yang dituju, menghentikan rotasi pesawat, dan menaikkan atau menurunkan roda pesawat.

Selanjutnya bagaimana cara melakukan proses validasi yang menunjukkan bahwa tangkapan dari kamera valid untuk dianggap sebagai sebuah aksi. Jika tangkapan tersebut dikatakan valid, maka aksi yang sesuai dengan hasil tangkapan akan dilaksanakan, jika tidak valid maka tidak melakukan aksi apapun. Berikut ini adalah persyaratan hasil tangkapan kamera dikatakan valid atau tidak valid:

1. Jika tidak ada satupun citra tangkapan kamera yang sesuai dengan data training kamera, maka dikatakan tidak valid sebagai sebuah aksi.
2. Jika lebih dari satu set tombol yang seharusnya tidak diperbolehkan untuk ditutup bersamaan, oleh pengguna baik sengaja ataupun tidak sengaja menutupnya, maka dianggap tidak ada aksi pada set tombol tersebut.
3. Jika sesuai dengan operasi yang disebutkan sebelumnya (nomor 1 sampai 6 pada penjelasan fungsi panel cetakan), maka dianggap sebagai tangkapan yang valid dan aksi bisa dilakukan.
4. Jika tidak ada satupun tombol yang ditutup pada hasil tangkapan kamera, maka dianggap tidak ada aksi yang dilakukan.
5. **METODOLOGI**
6. **Studi Literatur**

Pada tahapan ini akan dilakukan studi literatur mengenai metode yang digunakan, diantaranya :

* + - * 1. Cara pembuatan proyek dengan menggunakan Orbiter SDK.
        2. Cara menganalisa citra dari kamera dengan menggunakan OpenCV.

1. **Analisis**

Pada tahapan ini akan dilakukan analisa *requirement* yang akan dibutuhkan *plug-in* dan analisa proses yang nantinya akan terjadi pada *plug-in*.

1. **Perancangan**

Pada tahapan ini akan dilakukan perancangan desain *plug-in* dan alur algoritma yang dapat memenuhi hasil analisa *requirement* dan analisa proses yang dibuat pada tahap analisis.

1. **Implementasi**

Implementasi merupakan tahap membangun *plug-in* alat kemudi untuk game Orbiter Space Flight Simulator.

1. **Pengujian dan Evaluasi**

Pada tahap ini dilakukan uji coba terhadap sistem yang telah dibuat, dengan cara menilai apakah *plug-in* alat kemudi dapat bekerja dengan baik dan sudah memadai dalam membantu pengguna saat mengemudikan pesawat luar angkasa.

1. **Penyusunan Buku Tugas Akhir**

Tahap terakhir merupakan penyusunan laporan yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perancangan dan prototyping yang telah dibuat. Secara garis besar, buku laporan tugas akhir ini terdiri atas beberapa bagian yaitu:

1. Pendahuluan
   1. Latar Belakang
   2. Permasalahan
   3. Batasan Tugas Akhir
   4. Tujuan
   5. Metodologi
   6. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Uji Coba dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka
7. **JADWAL KEGIATAN TUGAS AKHIR**

Tugas akhir ini diharapkan bisa dikerjakan menurut jadwal sebagai berikut:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Kegiatan | Bulan | | | | | | | | | | |
| Februari | | Maret | | April | | | Mei | Juni | Juli | |
| 1. | Penyusunan Proposal Tugas Akhir |  | |  |  |  | |  |  |  |  | |
| 2. | Studi Literatur |  |  |  | |  | | |  |  |  |  |
| 3. | Implementasi |  | |  |  |  | | |  |  |  |  |
| 4. | Pengujian dan Evaluasi |  | |  |  |  |  | |  |  |  |  |
| 5. | Penyusunan Buku Tugas Akhir |  | |  |  |  |  | |  |  |  |  |

1. **DAFTAR PUSTAKA**

[1] Schweiger, Martin, *Orbiter API Reference Manual 2010 Edition*, 2010.

[2] Pisarevsky, Vadim, *Introduction to OpenCV*, Intel Corporation, Software and Solution Group, 2007.

[3] Pisarevsky, Vadim, *Object Detection with OpenCV*, Intel Corporation, Software and Solution Group, 2007.

[4] <http://www.orbiterwiki.org/wiki/Category:OrbiterSDK>, *Category:Orbiter SDK*, akses terakhir 27 Februari 2011.

[5] <http://www.orbiterwiki.org/wiki/Free_Compiler_Setup>, *Free Compiler Setup*, akses terakhir 27 Februari 2011.

**LEMBAR PENGESAHAN**

###### **Surabaya, 3 Maret 2011**

Menyetujui,

Pembimbing II

Sarwosri, S.Kom, M.T

NIP : 197608092001122001

Pembimbing I

Imam Kuswardayan, S.Kom, M.T

NIP : 197612152003121001