APLIKASI KAMERA VIDEO UNTUK PEMANTAU KEADAAN SUATU RUANGAN

Novrido Charibaldi [1),] Dessyanto Boedi Prasetyo [2),] Jatu Wiedyasari [3)]
[1,2,3)]Jurusan Teknik Informatika UPN "Veteran" Yogyakarta
Jl.Babarsari no 2 Tambakbayan 55281 Yogyakarta
e-mail: novrido@gmail.com

Abstrak
Peningkatan mobilitas manusia membuat perhatian terhadap masalah keamanan meningkat. Sehigga ada kebutuhan untuk memantau suatu ruangan dari jarak jauh yang dilengkapi dengan pengiriman pesan sebagai peringatan. Terdorong dengan perkembangan teknologi multimedia, aplikasi kamera video ini diharapkan mampu untuk memantau keadaan suatu ruangan, mendeteksi dan merekam pergerakan yang terjadi dalam ruangan tersebut dan mengirimkan foto hasilnya dalam bentuk Multimedia Messaging Service (MMS).
Metode yang digunakan dalam perancangan dan pembuatan perangkat lunak ini adalah metode Unified Process (UP). Aplikasi ini berjalan pada sebuah komputer dengan menggunakan teknologi Java Media Framework (JMF) dan Swing dalam Java 2 Standard Edition (J2SE), dengan bahasa pemrograman Java versi 1.6.0, dan menngunakan MySQL untuk basisdata, serta ActiveXperts sebagai aplikasi pengirim MMS. Bahasa pemodelan sistem yang digunakan adalah Unified Modelling Language (UML).
***Keywords** : Deteksi dan Rekam Pergerakan, Java Media Framework, Multimedia Messaging Service*

1. PENDAHULUAN

Masalah keamanan rumah belakangan ini mendapat perhatian serius, mengingat mobilitas manusia semakin meningkat yang menuntut pemilik rumah tidak berada di dalam rumah untuk mengawasinya sepanjang waktu. Telah diciptakan berbagai sistem keamanan rumah misalnya pemasangan alarm, bahkan telah dilengkapi dengan pengiriman pesan singkat berbasis teks atau *Short Messaging Service* (SMS) sebagai peringatan dengan memanfaatkan perangkat *mobile* yaitu *handphone*. Hal tersebut merupakan alternatif bagi pemilik rumah sebagai perangkat pemantau keadaan rumah dari jarak jauh yang relatif murah mengingat biaya pengiriman SMS tidak mahal.

Pesan singkat berbasis teks atau SMS hanya dapat memberikan informasi berupa peringatan tertulis ketika kamera menangkap suatu pergerakan, sedangkan pergerakan yang terjadi bisa disebabkan oleh hewan peliharaan, atau kerabat yang dianggap tidak membahayakan dan tidak perlu dicurigai yang kebetulan memasuki area *frame* kamera. SMS yang berisi peringatan berupa teks membuat pemilik rumah yang berada pada jarak jauh merasa khawatir dan menduga-duga apa atau siapa yang pergerakannya terdeteksi oleh kamera. Seiring perkembangan teknologi multimedia yang dapat menyajikan data baik teks, *image*, video, *audio*, animasi, maupun kombinasi dari data-data tersebut, sistem yang ada dapat dikembangkan dengan memberikan informasi berupa data multimedia melalui *Multimedia Messaging Service* (MMS). *Image* dan video sebagai data multimedia yang ditangkap oleh kamera video dapat memberikan informasi yang lebih dari sekedar pesan berbasis teks. Pesan berbasis multimedia atau MMS yang dikirimkan mampu membuat pemilik rumah melihat apa atau siapa yang ditangkap oleh kamera, sehingga pemilik rumah dapat mengontrol tindakan selanjutnya.

Tujuan penelitian ini adalah menghasilkan aplikasi kamera video untuk pemantauan keadaan suatu ruangan dari jarak jauh, dilengkapi dengan pengiriman pesan berbasis multimedia atau MMS.

Manfaat yang diharapkan dari penulisan skripsi ini adalah kemudahan di dalam melakukan pemantauan keadaan suatu ruangan dari jarak jauh dengan deteksi gerak dilengkapi fasilitas pengiriman pesan berbasis multimedia atau MMS.

2. TINJAUAN PUSTAKA

Studi pustaka pada penelitian ini mempelajari aplikasi dengan tema yang serupa yang telah dibangun oleh Andreas Handojo dan Sugianto, mahasiswa Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra, serta Resmana Lim mahasiswa jurusan Teknik Elektro, Fakultas Teknologi Industri, Universitas Kristen Petra, dengan judul Aplikasi *Security Surveillance System* Menggunakan *Webcam* dan HP dengan Fasilitas *General Packet Radio Services* dan MMS. Aplikasi tersebut dibangun dengan bahasa pemrograman Visual basic 6.0 dan Jbuilder Personal 9, memanfaatkan *tool* VideoOCX untuk pendeteksian gerak, serta Nokia MMS *Library* dan JWAP *Protocol Stack* untuk pengiriman MMS. Aplikasi tersebut dijalankan pada seperangkat komputer yang telah terhubung dengan sebuah *webcam* dan sebuah *handphone* sebagai pengirim MMS.

Cara kerja aplikasi tersebut adalah kamera video *webcam* menangkap segala sesuatu yang terjadi pada suatu ruangan, dan dapat mendeteksi adanya pergerakan yang terjadi. Pendeteksian pergerakan menggunakan *tool* VideoOCX dilakukan dengan menghitung perbedaan yang terjadi antara dua *image*. Metode yang digunakan adalah dengan cara melakukan penghitungan nilai rata-rata dari semua nilai *grayvalue* dalam suatu gambar yang

dapat disebut juga sebagai *mean*. Nilai *mean* yang didapat akan dibandingkan dengan nilai *threshold* yang ditentukan oleh user. Suatu nilai *threshold* digunakan sebagai acuan agar *webcam* mulai meng-*capture* gambar. Ketika aplikasi mendeteksi adanya pergerakan, secara otomatis aplikasi akan menyimpan data multimedia dalam bentuk *image* dengan format JPEG. Selain menyimpan gambar, aplikasi akan mengirimkan foto melalui MMS kepada telepon genggam pemilik rumah atau bangunan.

Perbedaan dengan penelitian ini adalah aplikasi kamera video untuk pemantau keadaan suatu ruangan dilengkapi dengan fasilitas *player* untuk melihat hasil penangkapan baik foto maupun video. Fitur tambahan lainnya adalah menyimpan informasi data hasil dan *login* admin pada sebuah basisdata, serta melakukan penghapusan data baik pada basis data maupun pada *harddisk*. Selain perbedaan fitur yang disediakan, perbedaan yang mendasar adalah bahasa dan teknologi yang digunakan untuk membangun aplikasi. Pada penelitian ini digunakan bahasa pemrograman Java dengan memanfaatkan teknologi *Java Media Framework* untuk pembangunan aplikasi.

Pendeteksian gerak dilakukan dengan cara membandingkan dua buah *image*, yaitu image sekarang dan *image* sebelumnya. Perbandingan yang dilakukan dengan melakukan operasi pengurangan terhadap nilai *pixel-pixel*, sehingga diperoleh sebuah nilai selisih yang disebut dengan perubahan. Nilai perubahan yang telah didapat akan dibandingkan dengan nilai ambang yang telah ditentukan, yaitu 50. Semua perubahan yang terdeteksi akan diakumulasi dan dibandingkan dengan nilai kesensitifan *motion detection* yang ditentukan oleh *user.* Tingkat kesensitifan *motion detection* inilah yang dapat dikendalikan sesuai kebutuhan oleh *user*. Pengiriman MMS dilakukan menggunakan sebuah *software* yang bernama *ActiveXperts* SMS *and* MMS *ToolKit* 5.1 yang dibuat oleh *ActiveXperts Software*.

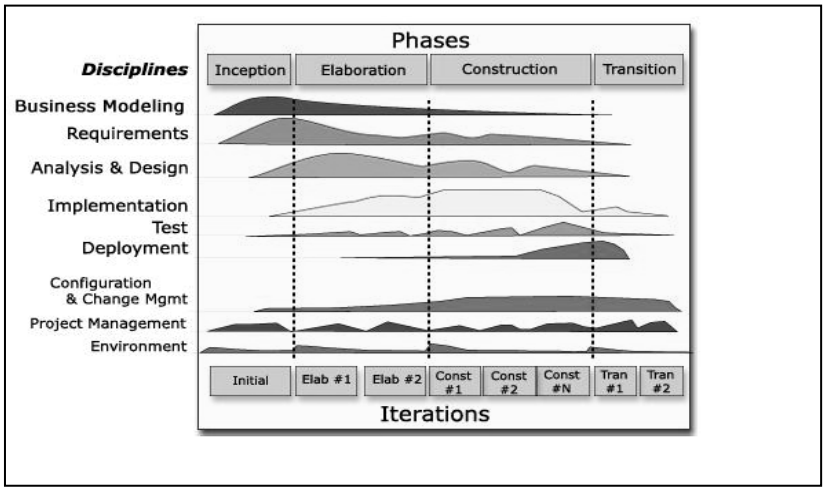
3. METODE PENELITIAN

Penelitian ini dikembangkan dengan menggunakan metodologi *Rational Unified Process* atau *Unified Process* (RUP/UP). RUP/UP merupakan metodologi dengan proses iteratif yang terdiri dari empat fase, yaitu:

a. Insepsi (*Inception*)
b. Elaborasi (*Elaboration*)
c. Konstruksi (*Construction*)
d. Transisi (*Transition*)

Seluruh proyek yang menggunakan RUP/UP harus mengikuti keempat fase tersebut. Arsitektur RUP/UP terdiri dari dua dimensi (Kroll dan Kruchten, 2003) yang dapat dilihat pada gambar 1, yaitu:

a. Dimensi pertama digambarkan pada sumbu horisontal yang mewakili waktu, dan menunjukkan struktur dinamik dari pengembangan perangkat lunak yang dijabarkan dalam tahapan pengembangan atau fase. Setiap fase akan memiliki suatu *major milestone* yang menandakan akhir dari awal dari fase selanjutnya. Setiap fase dapat terdiri dari satu atau beberapa iterasi. Dimensi ini terdiri atas *Inception*, *Elaboration*, *Construction*, dan *Transition*.
b. Dimensi kedua digambarkan pada sumbu vertikal mewakili struktur statis yang menunjukkan *workflows* atau *disciplines* yang harus dikerjakan selama siklus hidup proyek, meliputi *who is doing*, *what*, *how* dan *when*. Dimensi ini terdiri atas *Business Modeling, Requirements, Analysis and Design, Implementation, Test, Deployment, Configuration and Change Manegement, Project Management,* dan *Environtment.*

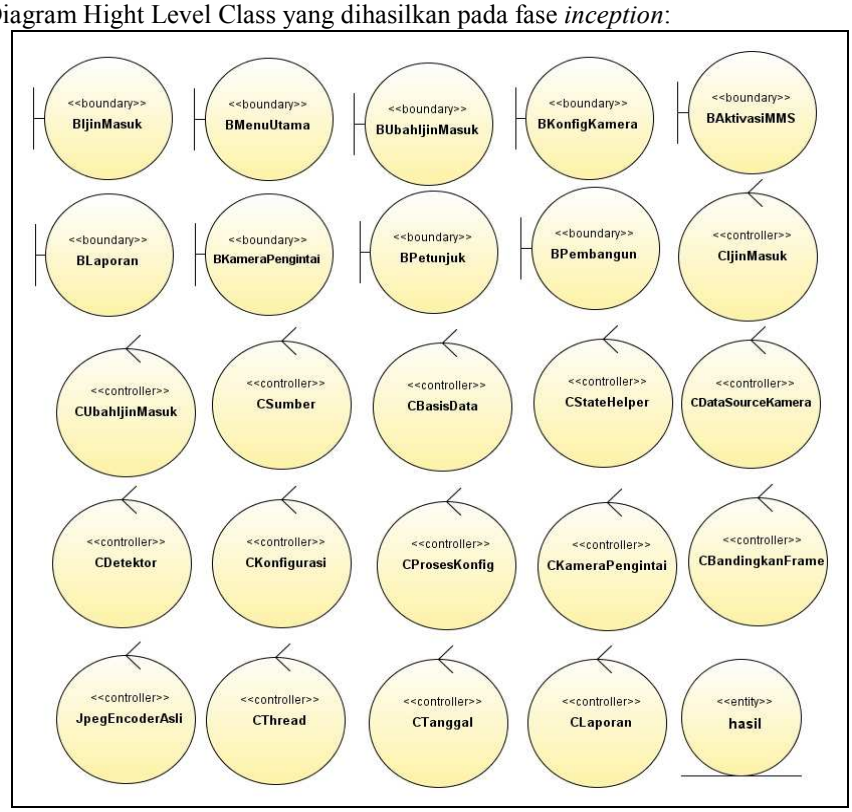


Gambar 1 Arsitektur RUP/UP

3.1. *Requirements Gathering*

Requirements bertujuan untuk mendeskripsikan apa yang harus dilakukan oleh sistem. Terdapat dua jenis *requirements* yang harus dilakukan untuk menghasilkan sebuah sistem yang berkualitas, yaitu kebutuhan fungsional dan kebutuhan non-fungsional. *Requirements* dilakukan pertama kali pada fase *inception*, kemudian kembali dilakukan pada fase *elaboration*.

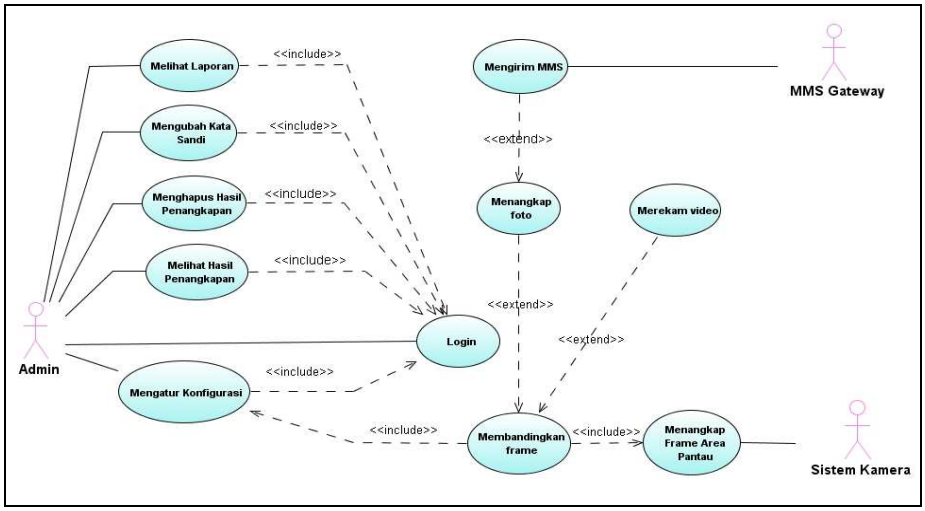
Berikut Diagram Hight Level Class yang dihasilkan pada fase *inception*:



Gambar 2 Diagram *High Level Class* Aplikasi Kamera Video untuk Pemantau Keadaan Suatu Ruangan

3.2. Analysis dan Perancangan

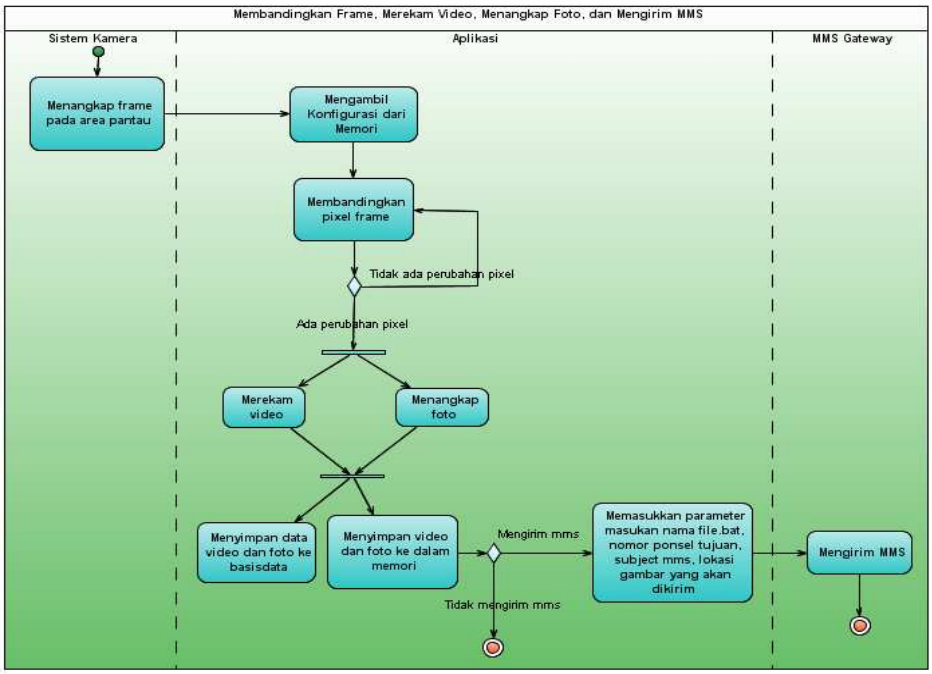
Proses pengembangan perangkat lunak pada RUP/UP berlangsung melalui serangkaian aktivitas yang diturunkan dari *use case*, sehingga diagram UML yang harus dibuat pertama kali adalah diagram *use case*.



Gambar 3 Diagram *Use Case*

Berikut adalah diagram yang menggambarkan urutan proses membandingkan *pixel frame* untuk mendeteksi apakah ada perbedaan ukuran *pixel*. Apabila ditemukan perbedaan ukuran *pixel*, maka akan terjadi proses perekaman video dan penangkapan foto. Proses membandingkan *pixel*, perekaman video, dan

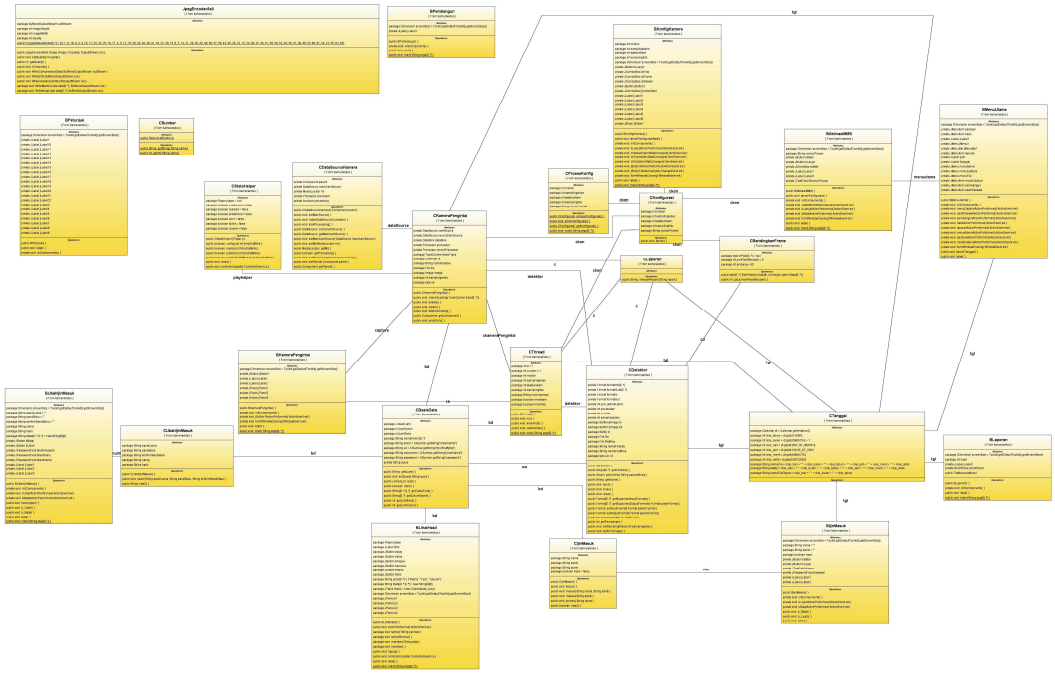
penangkapan foto diatur dengan konfigurasi yang telah dilakukan oleh admin, dan disimpan di dalam sebuah *file* di dalam memori.



Gambar 4 Diagram *Activity* Proses Membandingkan *Frame*, Merekam Video, Menangkap Foto, dan Mengirim MMS

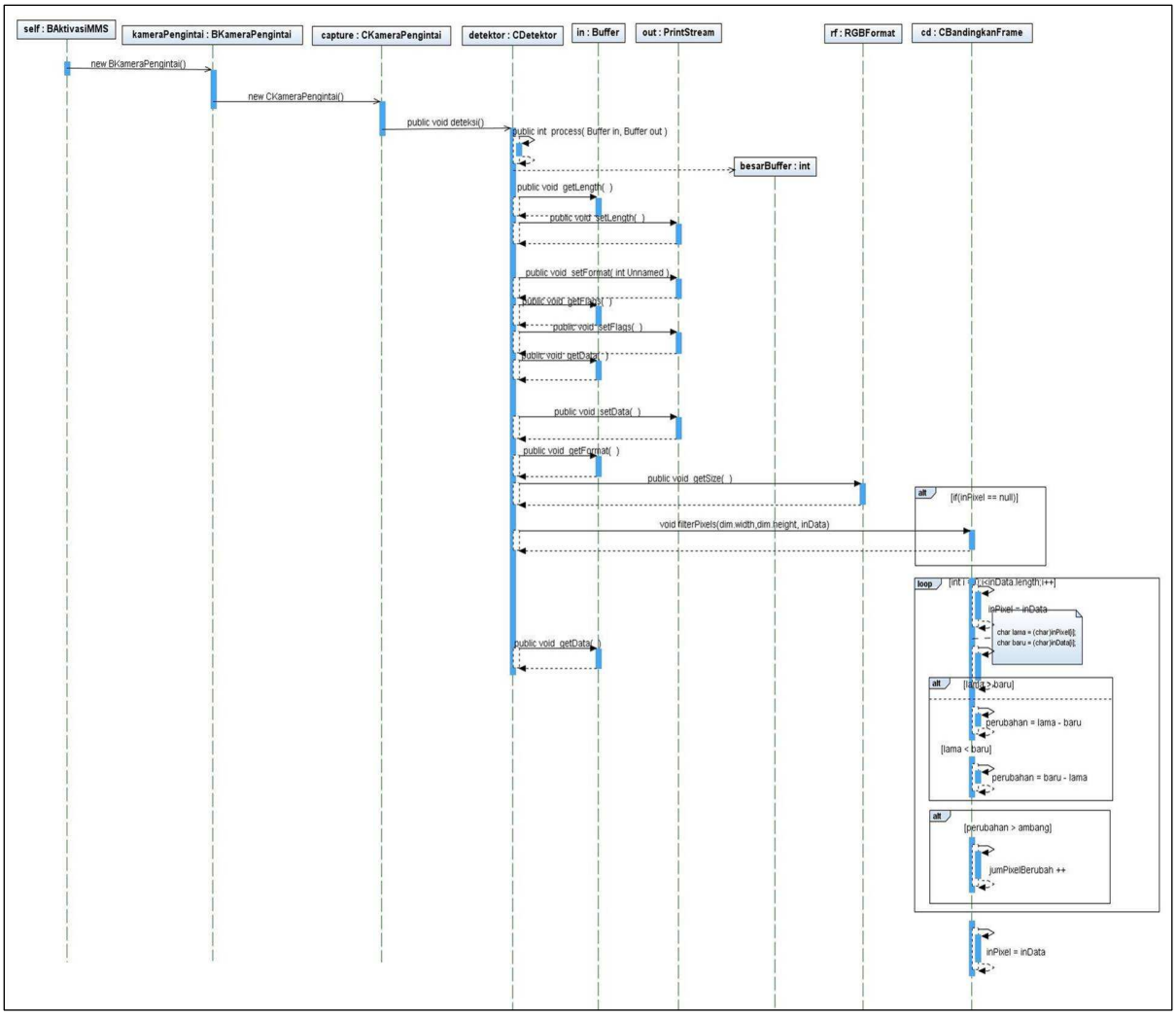
Aplikasi ini dirancang terdiri dari 25 kelas yang terdiri dari kelas BIjinMasuk, BMenuUtama, BUbahIjinMasuk, BKonfigKamera, BAktivasiMMS, BLaporan, BKameraPengintai, BPetunjuk, BPembangun, CIjinMasuk, CUbahIjinMasuk, CSumber, CBasisData, CStateHelper, CDataSourceKamera, CDetektor, CKonfigurasi, CProsesKonfig, CKameraPengintai, CBandingkanFrame, JpegEncoderAsli, CThread, CTanggal, CLaporan, dan kelas hasil.

Berikut Diagram Class dari aplikasi ini:



Gambar 5 Diagram *Class* Aplikasi Kamera Video untuk Pemantauan Keadaan Suatu Ruangan

Use case untuk membandingkan *frame* adalah kasus penting yang harus ditunjukkan dengan Diagram Sequence untuk melihat urutan *method* dan objek-objek yang terlibat di dalamnya. Aplikasi secara otomatis akan membandingkan *frame* setelah *user* melakukan konfigurasi. Berikut diagram tersebut:



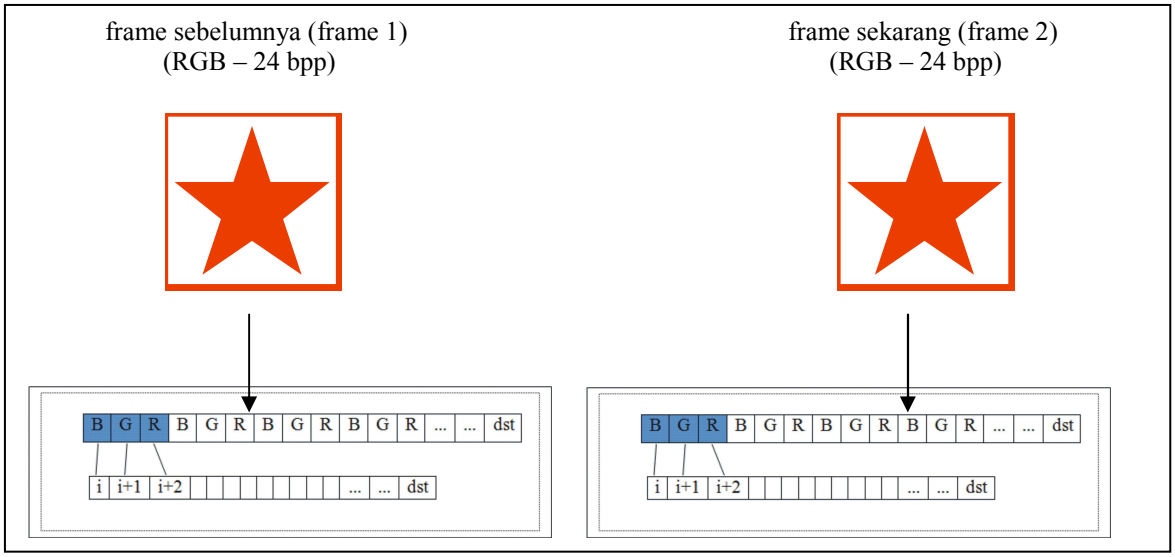
Gambar 6 Diagram Sequence Untuk Use Case Membandingkan Frame

Berikut langkah pendeteksian gerakan dengan cara membandingkan *frame* video saat ini dengan *frame* selanjutnya yang dilakukan pada penelitian ini adalah: (perhatikan gambar 7-9)

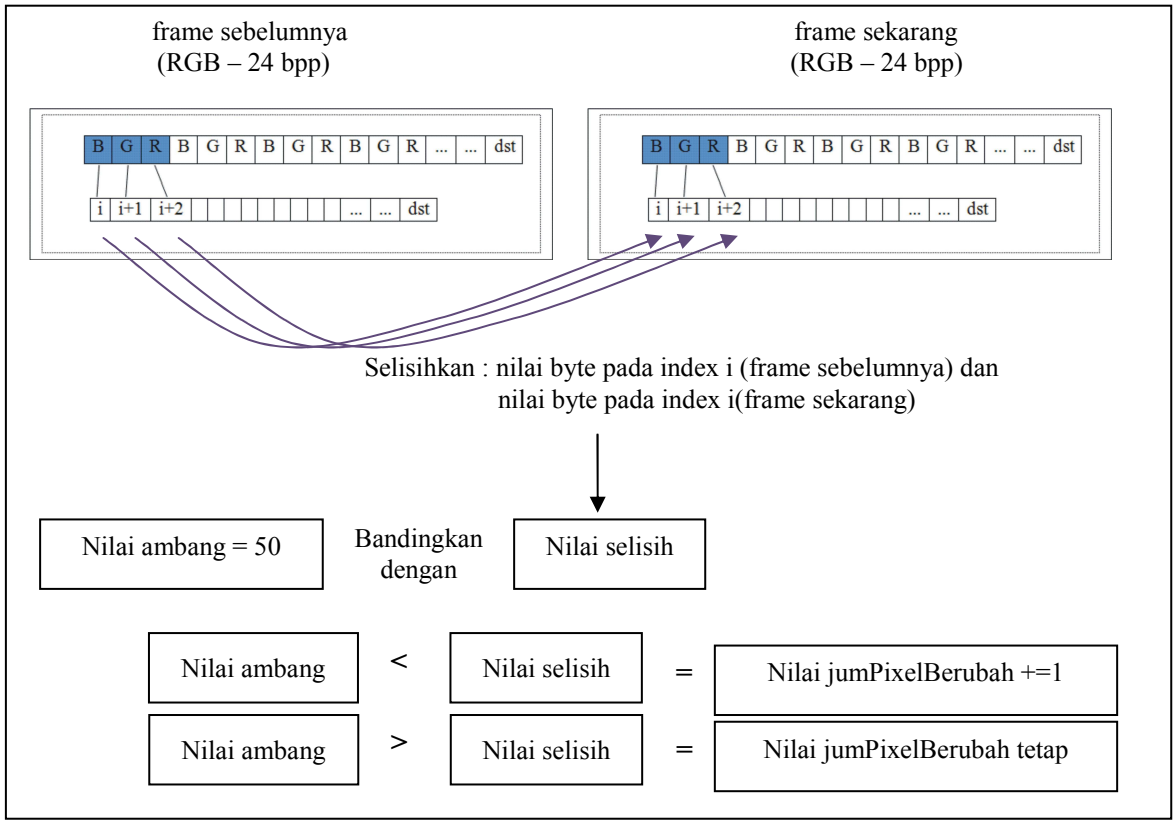
a. Menangkap warna *image* dengan menangkap *byte* yang bertipe *array* satu dimensi pada *buffer* untuk *frame* saat ini dan *frame* sebelumnya. Pengubahan *image* berwarna menjadi *image grayscale* tidak dilakukan karena memakan waktu yang relatif lama dan dapat mengurangi kesensitifan pendeteksian gerakan.
b. Melakukan penghitungan selisih nilai *byte* warna dan membandingkan dengan nilai ambang.
   Selisihkan nilai *byte* masing-masing index pada *image frame* sekarang dengan *frame* sebelumnya. Bandingkan masing-masing selisih dari tiap pixel dengan nilai ambang yang telah ditentukan, yaitu nilai ambang = 50. Apabila nilai selisih dari tiap *pixel* melebihi nilai ambang = 50, maka dianggap telah terjadi perubahan pada satu titik *pixel* yang disimpan ke dalam nilai jumPixelBerubah. Nilai jumPixelBerubah akan terus bertambah untuk setiap selisih yang ditemukan pada tiap *pixel*.
c. Membandingkan nilai akhir jumPixelBerubah dengan nilai kesensitifan *motion detector* yang ditentukan oleh *user*. Apabila nilai jumPixelBerubah melebihi nilai kesensitifan *motion detector* yang ditentukan oleh *user*, maka dideteksi bahwa telah terjadi pergerakan.
d. Melakukan penghapusan *frame* sebelumnya dan digantikan dengan *frame* selanjutnya

Ilusrasi dari keempat langkah tersebut dapat diperhatikan gambar 7-9.

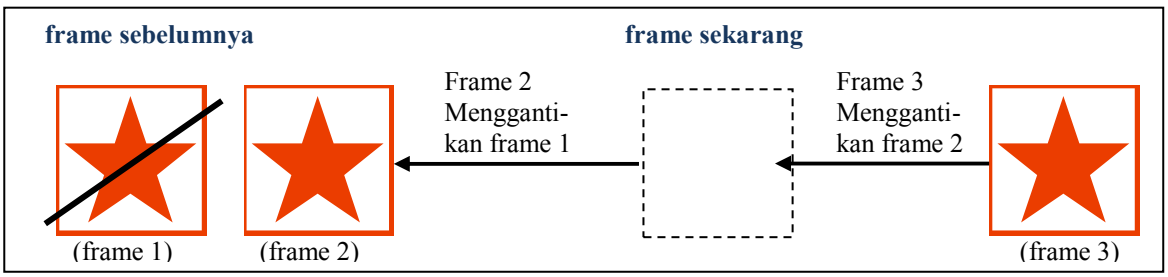
ISSN: 1979-2328



Gambar 7 Menangkap Warna pada *Image Frame* Sekarang dan *Frame* Sebelumnya



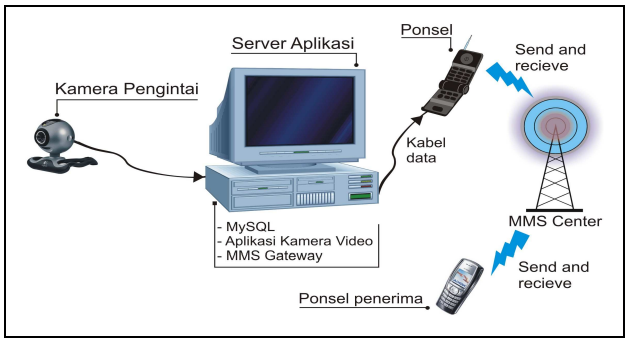
Gambar 8 Penghitungan Selisih Nilai *Byte* Warna dan Perbandingan dengan Nilai Ambang



Gambar 9 Gambar Penghapusan *Frame* Pertama yang Digantikan *Frame* Kedua

Frame sekarang kini berisi *frame* ketiga, sedangkan *frame* sebelumnya berisi *frame* kedua. *Frame* sekarang dan *frame* sebelumnya dibandingkan seperti langkah a hingga langkah c. Apabila hasil perbandingan tidak diperoleh perubahan, maka tidak terjadi pergerakan, kemudian terus melakukan iterasi selanjutnya.

Berikut dirancang arsitektur sistem yang diperlukan agar sistem yang akan dibangun memiliki konstruksi yang baik, proses pengolahan data yang tepat dan akurat, memiliki nilai, memiliki aspek *user friendly* dan memberikan dasar-dasar untuk pengembangan selanjutnya. Arsitektur sistem dalam penelitian ini dapat dilihat pada gambar 10.



Gambar 10 Arsitektur Sistem

4. HASIL DAN PEMBAHASAN

Pada bagian ini hanya akan dibahas hasil implementasi dari beberapa kelas terpenting yang berperan dalam pembangunan aplikasi pada penelitian ini.

Kelas CBandingkanFrame menjalankan proses perbandingan dua buah *frame*, yaitu *frame* sekarang dengan *frame* sebelumnya, dilanjutkan membandingkan nilai perubahan tersebut dengan nilai ambang. Proses perbandingan dua buah *frame* dilakukan dengan metode pengurangan nilai *pixel*. Pengambilan *pixel* video yang ditangkap kamera video telah dilakukan oleh kelas Cdetektor yaitu oleh *method* getData(), dilanjutkan dengan pemanggilan *method* filterPixels(dim.width,dim.height, inData) milik kelas CBandingkanFrame.

Pemrosesan data *image* tidak diawali dengan pengubahan *image* berwarna menjadi *image grayscale* dikarenakan waktu yang diperlukan untuk melakukan operasi tersebut relatif lama. Waktu yang relatif lama membuat deteksi gerakan yang terjadi kurang mendekati kondisi *realtime*, karena terdapat jeda waktu yang menyebabkan aplikasi lambat mengenali suatu pergerakan.

Nilai perubahan yang didapat dari perbandingan dua buah *frame* dibandingkan dengan nilai ambang yang bernilai 50. Nilai jumPixelBerubah akan bertambah jika nilai perubahan lebih besar dari nilai ambang. *Pixel-pixel* yang berubah diakumulasi, dan total perubahan tersebut akan dibandingkan dengan tingkat kesensitifan *motion detection* yang bebas ditentukan oleh *user* dan telah disimpan di dalam file Konfigurasi. Proses perbandingan tingkat kesensitifan *motion detection* dilakukan pada kelas CThread.

Berikut cuplikan class CBandingkanFrame:

```
package kameravideo;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
public class CBandingkanFrame {
    CKonfigurasi ckon;
    byte[] inPixel = null;
    int jumPixelBerubah = 0;
    int ambang = 50;
    public byte[] filterPixels(int width, int height, byte[] inData) {
        if(inPixel == null) inPixel = inData;
        jumPixelBerubah=0;
        for(int i = 0;i<inData.length;i++){
            char lama = (char)inPixel[i];
            char baru = (char)inData[i]
            int perubahan = (lama > baru) ? lama-baru : baru-lama;
            if(perubahan > ambang) jumPixelBerubah++;
        }
```

Modul Program 1. class CBandingFrame

```
        inPixel = inData;
        return inData;
     }
     public int getJumlahPixelBerubah(){
        return jumPixelBerubah;
     }
}
```

Modul Program 2. Lanjutan class CBandingFrame

Kelas CThread bertugas untuk menjalankan proses penangkapan gambar dari *buffer*, membandingkan *frame*, merekam video, menangkap foto, dan mengirimkan mms. Berikut kode program dari beberapa method yang penting pada kelas CThread.

```
public void run() {
        System.out.println(motion);
        System.out.println(samplingvideo);
        System.out.println(samplingfoto);
        System.out.println(wakturekam);
        System.out.println(nomorponsel);
        while (true) {
            try {
                sleep(ckon.samplingvideo * 100);
            } catch (Exception e) { }
            System.out.println("perubahan (thread):"
              + ckameraPengintai.
              detektor.getPerubahan());
            if (ckameraPengintai.detektor.getPerubahan()> motion) {
                if (!merekam) {
                    merekam = true;
                    rekamVideo();
                }
            if (!memfoto) {
                    memfoto = true;
                    ambilFoto();
                }
            }
            counter++;
        }
    }
```

Modul Program 3. Method run() Milik class CThread

```
public void rekamVideo() {
  Thread t = new Thread() {
    public void run() {
       int i = 0;
       ckameraPengintai.rekam();

       System.out.println(ckameraPengintai.detektor.getPerubahan());
       System.out.println("rekaman mulai");
       while (merekam) {
            try {
                 sleep(1000);
            }
            catch (Exception e) {
            }
```

Modul Program 4. Method rekamVideo() class CThread

```
            //hentikan jika tidak ada pergerakan
            if (wakturekam == 0 && ckameraPengintai.
               detektor.getPerubahan() < motion) {
               merekam = false;
               ckameraPengintai.stopRecording();
               System.out.println("dihentikan pergerakan");
            }
            //hentikan jika waktu rekam lewat
            if (i > wakturekam) {
               System.out.println("dihentikan waktu");
               merekam = false;
               ckameraPengintai.stopRecording();
            }
            System.out.println("masih merekam");
            i++;
            System.out.println(i);
        }//akhir while
        System.out.println("rekaman berhenti");
        //hentikan rekaman
        ckameraPengintai.stopRecording();
      }//method run()
   }; //akhir thread
   t.start();
}
```

Modul Program 5. Lanjutan Method rekamVideo() class CThread

```
public void ambilFoto() {
    tgl = new CTanggal();
    Thread t = new Thread() {
      int counter = 1;
      public void run() {
          System.out.println("pengambil foto mulai");
          while (memfoto) {
             try {
               sleep(ckon.samplingfoto * 100);
             }
             catch (Exception e) {
             }
             //ambil foto
             System.out.println("Ambil foto " + counter + " "
             +ckon.samplingfoto);

             ckameraPengintai.ambilFoto();
             if (ckameraPengintai.detektor.getPerubahan()< motion) {
                memfoto = false;
                System.out.println("dihentikan pergerakan");
             }
             if (!nomorponsel.equals("") && counter == 2) {
                //kirim mms
                try {
                  String cmdarray[] ={"run.bat",nomorponsel,"kirim-mms-"
                  +ckameraPengintai.detektor.namaFileMms ,"C:/hasil/"
                  +ckameraPengintai.detektor.namaFileMms };
                   Runtime r = Runtime.getRuntime();
                   r.exec(cmdarray);
                   c.simpanReport("Mengirim data foto "
                                  +ckameraPengintai.detektor.namaFileMms
                                  + " melalui mms pada " +tgl.waktu);
```

Modul Program 6. Method ambilFoto() class CThread

```
                    System.out.println("Mengirim data foto "
                                  +ckameraPengintai.detektor.namaFileMms
                                  + " melalui mms pada "
                                  +tgl.waktu);
                    System.out.println("mengirim foto ke" + counter);
                    bd = new CBasisData();
                    String query = "update hasil set statusMMS = 1
                           where nama= '"+ckameraPengintai.namaFileEks+"'
                           and tipe='Foto'";
                    bd.setQuery(query);
                    bd.store();
                    //simpan report
                    c.simpanReport("Mengirim data foto "
                                +ckameraPengintai.namaFileEks
                                +" melalui mms pada "
                                +ckameraPengintai.tgl);
                 }
                 catch (Exception e) {
                    System.err.println("GAGAL");
                    e.printStackTrace();
                 }
              }// akhir if
              counter++;
           }//akhir while
          System.out.println("pengambil foto selesai");
       }// akhir method run()
     };// akhir thread
     t.start();
}// akhir method ambilFoto()
```

Modul Program 7. Lanjutan Method ambilFoto() class CThread

Berikut ini adalah hasil penangkapan ketika dideteksi ada pergerakan yaitu didapatkan beberapa foto yang tertangkap.

Gambar 13 Tampilan Hasil Foto 13.10.2008-9.26.17.jpeg

Hanya satu foto yang dikirim melalui mms dari beberapa foto yang tertangkap, yaitu foto dengan nama 13.10.2008-9.26.16.jpeg. Berikut ini adalah tampilan MMS yang berhasil diterima oleh *handphone* penerima.

Gambar 14 Tampilan MMS yang Berhasil Dikirim

5. KESIMPULAN

Dari semua tahap yang telah dilakukan maka dapat disimpulkan bahwa telah berhasil dibangun aplikasi kamera video pemantau keadaan suatu ruangan, yang memiliki kemampuan:

a. Dapat mendeteksi adanya pergerakan yang terjadi di dalam area pemantauan.
b. Dapat merekam video selama terdeteksi adanya pergerakan.
c. Dapat mengirim foto hasil penangkapan selama terjadi pergerakan melalui mms.
d. Dapat menampilkan laporan kegiatan yang dilakukan selama aplikasi dijalankan.

6. DAFTAR PUSTAKA

Fowler, Martin, 2005, *UML Distillled Edisi 3*, *Panduan Singkat Bahasa Pemodelan Objek Standar,* Andi Offset, Yogyakarta.

Hermawan, Benny, 2004, *Menguasai Java 2 dan Object Oriented Programming*, Andi Offset, Yogyakarta.

Kadir, Abdul, 2004 , *Dasar Pemrograman Java 2*, Penerbit Andi Offset Yogyakarta.

Kirillov, Andrew, 2005, *Motion Detection Algorithms.* <http://www.codeproject.com/KB/audio-video/Motion_Detection.aspx>, (diakses tanggal 18 September 2007).

Kruchten, P., dan Kroll, P., 2003, *Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, The, Addison Wesley.

Le Bodic, Gwenael, 2003, *Mobile Messaging Technologies and Services SMS, EMS, and MMS*, Chichester, John Wiley & Sons, Ltd.

Munawar, 2005, *Pemodelan Visual dengan UML*, Graha Ilmu, Yogyakarta.

Munir, Rinaldi, 2004, *Pengolahan Citra Digital dengan Pendekatan Algoritmik*, Informatika, Bandung.

Subrahmanian, V. S., 1998, *Principles of Multimedia Database Systems*, Morgan Kaufman Publishers, Inc., San Fransisco, California.