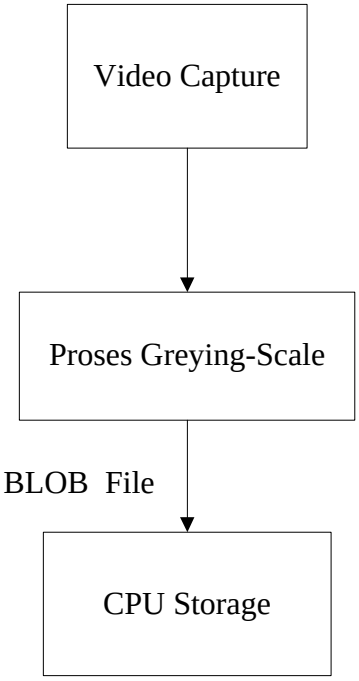
3. PERENCANAAN SISTEM

Pengerjaan Tugas Akhir ini terdiri dari beberapa bagian sistem, antara lain : sistem database, metode penyimpanan data *training* dan metode meng-*upload* data *training* ke memori untuk proses *recognition*, sistem perhitungan matriks yang akan digunakan dalam *training* maupun *recognition*, dan kontrol webcam agar dapat mengambil gambar dengan baik

Database berfungsi menyimpan *image-image* yang digunakan untuk *training* dan *image-image* yang digunakan untuk *recognition*. Secara garis besar, proses *saving image* ke database dapat digambarkan pada gambar 3.1. berikut ini :



Gambar 3.1. Blok Diagram Proses Input Data

3.1. Perancangan Database

Database menggunakan system *database Paradox* yang disediakan oleh *Borland Delphi 5.0*. System *Paradox* ini dipilih karena sudah tersedia dalam satu paket dengan *Delphi 5.0*, dan system Paradox memiliki kecepatan akses yang tinggi.

Software tugas akhir ini akan menggunakan dua buah tabel; yaitu '*tabelorang'*, dan **'***tabelgambar'***. '***tabelgambar'* digunakan untuk menyimpan *image – image* yang akan di-*training*, '*tabelorang'* digunakan untuk mengelompokkan *image-image training* ke dalam kelasnya masing-masing. Semua tabel ini disimpan ke dalam satu direktori, dan pada *database desktop* diberi *alias name* 'TA'. *Alias name* ini harus diperhatikan supaya benar – benar menunjuk pada tabel yang dimaksud. *Alias name* dapat diatur dengan menggunakan program *Database Desktop* yang juga telah disediakan oleh *Borland Delphi 5.0*. *Field-field* dari tiap tabel itu dijabarkan sebagai berikut :

3.1.1. *Field-field* pada tabel tabelgambar

Tabel 3.1. Field Tabel *tabelgambar*

Field	**Type**	**Size**	**Keterangan**
NamaOrang	Alpha	255	Menyimpan identitas (nama) dari image yang dimaksud
NamaFile	Alpha	255	Menyimpan *path* dari image training
Gambar	Graphics	-	Menyimpan file gambar dari image training.

Tabel 'tabelgambar' disimpan dengan nama *tabelgambar.db.* Khusus untuk field gambar, menggunakan tipe field *Graphics*. Tujuannya,supaya tidak perlu lagi menyimpan file-file gambar (*.bmp), karena semua file tersebut sudah disimpan di dalam *database*. *Field* ini tidak perlu memiliki *size* yang *fixed*, karena file-file dalam *field* ini disimpan dalam file yang berbeda, yaitu *tabelgambar.mb*. Isi dari *field* ini tidak bisa begitu saja dilihat. Jika isi *field* dibuka

pada program *Database Desktop* dari *Delphi*, yang ditampilkan sebagai isi *field* ini hanyalah tulisan *(BLOB File; Binary LargeObject).* Untuk melihat isi *field* perlu dibuat sebuah program khusus dengan menggunakan *object* dari *Delphi* yang disebut *dbimage*. *Object* ini berfungsi menampilkan isi dari *BLOB File* (*Binary Large Object*) begitu *record* yang bersangkutan diaktifkan.

Tabel 'tabelgambar' diindex berdasarkan *field* 'namaorang' dan 'namafile' , tujuannya supaya jika ada *record* yang dihapus atau ditambahkan, semua data akan diurutkan kembali berdasarkan *index* tersebut.

3.1.2. *Field-field* pada tabel *tabelorang*

Tabel 3.2. Field tabel *tabelorang*

Field	**Type**	**Size**	**Keterangan**
NamaOrang	Alpha	255	Menyimpan identitas (nama) dari *class*
JumlahSample	Integer	-	Menyimpan jumlah *image training* yang dimiliki *class* tersebut.
Dari	Integer	-	Menyimpan nomor urut awal dari *image-image training* yang dimiliki kelas tersebut dalam distribusi semua *image training*.
Sampai	Integer	-	Menyimpan nomor urut akhir dari *image-image training* yang dimiliki kelas tersebut dalam distribusi semua *image training*

Field 'JumlahSample', 'Dari' , dan 'Sampai' digunakan untuk mempermudah perhitungan matriks; khususnya pada metode LDA. Dalam metode LDA, ada pengelompokan *image-image* dari *class* yang sama. Karena itu perlu menggunakan *field – field* tersebut saat training dilakukan.

$$S_b = \sum_{i=1}^{c} N_i (m_i - m)(m_i - m)^T \quad (3.1.)$$

$$S_w = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - m_i)(x_k - m_i)^T \quad (3.2.)$$

Pada rumusan untuk mencari 'Sw' dan 'Sb'[1] di atas, bisa diambil contoh sebagai berikut :

Tabel 3.3. Contoh Isi Tabel *tabelorang*

NamaOrang	**JumlahSample**	**Dari**	**Sampai**
Vilas01	1	0	0
Vilas02	1	1	1

Variabel *c*, yaitu jumlah kelas, adalah jumlah record yang ada pada 'tabelorang', dalam hal ini adalah 2 (dua), karena hanya ada dua kelas.

N_1 dan N_2 masing-masing diberi nilai yang ada pada field JumlahSample, yaitu $N_1 = 1$ dan $N_2 = 1$.

Untuk menghitung ì1 dapat dilakukan dengan cara mencari rata-rata *image* 2 dapat dilakukan dengan mencari rata-rata *image* dengan nomor urut 1 sampai 1. Setelah itu, 'Sb' dapat dicari dengan mengurangkan masing-masing ì i dengan rata-rata total LDA (ì).

Untuk mencari perhitungan $(x_k - m_i)$ { yaitu mengurangi *image-image* yang termasuk kelas ke-i dengan rata-rata kelas ke-I } dapat dilakukan dengan mengurangi *image* dengan nomor urut 0 sampai 0 (karena *software* didesain satu nama hanya memiliki satu *image*) dengan rata-rata kelas ke-1, dan *image* dengan nomor urut 1 sampai 1 dengan rata-rata kelas kedua. Cara ini akan mempermudah perhitungan matriks pada LDA. Secara garis besar, *field* 'Dari' dan 'Sampai' digunakan sebagai *counter* untuk menghitung data pada kelas-kelas yang berbeda. Untuk lebih jelasnya dapat dilihat pada *listing program* yang terlampir pada laporan ini.

[1] Diambil dari: Raymond "Face Recognition Menggunakan Metode Linear DiscriminantAnalysis (LDA)" Universitas Krister Petra, 2002

Tabel '*tabelorang*' ini akan berubah jika terdapat penambahan atau pengurangan *image training* pada tabel '*tabelgambar*' . *Software* tinggal mengambil nilai yang ada pada *field* 'JumlahSample' untuk menentukan berapa jumlah *image training* pada *class* yang bersangkutan; dan nilai pada *field* 'Dari' dan 'Sampai' untuk mengetahui pada nomor urut berapa *image-image* tersebut berada di tabel '*tabelgambar*' .

3.2. Penyimpanan / Pemanggilan Data Yang Diperlukan

Pada program tugas akhir ini terdapat beberapa data matriks yang perlu disimpan. Data matriks ini perlu disimpan agar proses *training* tidak dilakukan berulang-ulang. Cukup meng-*upload* data tersebut ke memori dan kemudian program bisa membentuk matriks yang diinginkan berdasarkan file yang dimaksud.

Data matriks yang perlu disimpan adalah : Rata-rata total PCA (û), V_{PCA}, *feature* PCA, Rata-rata total LDA (î), V_{LDA}, dan *feature* LDA.

Berikut ini adalah nama - nama file yang digunakan untuk menyimpan data matriks di atas.:

Tabel 3.4. Daftar File yang Disimpan

Nama File	Data Yang Disimpan
FeatureLDA.txt	*Feature* LDA
FeaturePCA.txt	*Feature* PCA
Rata2TotalLDA.txt	Rata-rata total LDA
Rata2TotalPCA.txt	Rata-rata total PCA
V_LDA.txt	*Eigen Vector* dari LDA
V_PCA.txt	*Eigen Vector* dari PCA

Semua file di atas adalah file *text*. Penyimpanan dilakukan setelah matriks selesai dihitung. Perhitungan matriks lebih detail akan dijabarkan lebih lanjut pada akhir bab ini. Setelah hasil matrix yang bersangkutan diketemukan, semua data yang ada di matriks tersebut akan disimpan ke dalam file yang bersangkutan dengan variabel *string*.

Sebagai contoh, pada matriks *feature* PCA; seandainya diketemukan sebuah matriks berdimensi 100 x 100 (untuk 100 buah *image training*), maka akan diketemukan sebanyak 10.000 buah nilai. Nilai-nilai ini kemudian akan disimpan ke file *FeaturePCA.txt* dengan urutan per-baris dan per-kolom. Jadi file *FeaturePCA.txt* akan memiliki 10.000 baris.

Untuk memanggil data matriks kembali ke memori, *software* akan mengambil 100 baris pertama dan memasukkan datanya ke kolom I dari matriks *feature* PCA, 100 baris kedua ke kolom II, dst. Proses yang sama juga dilakukan untuk *file* yang lain.

3.3. Perancangan *Software*

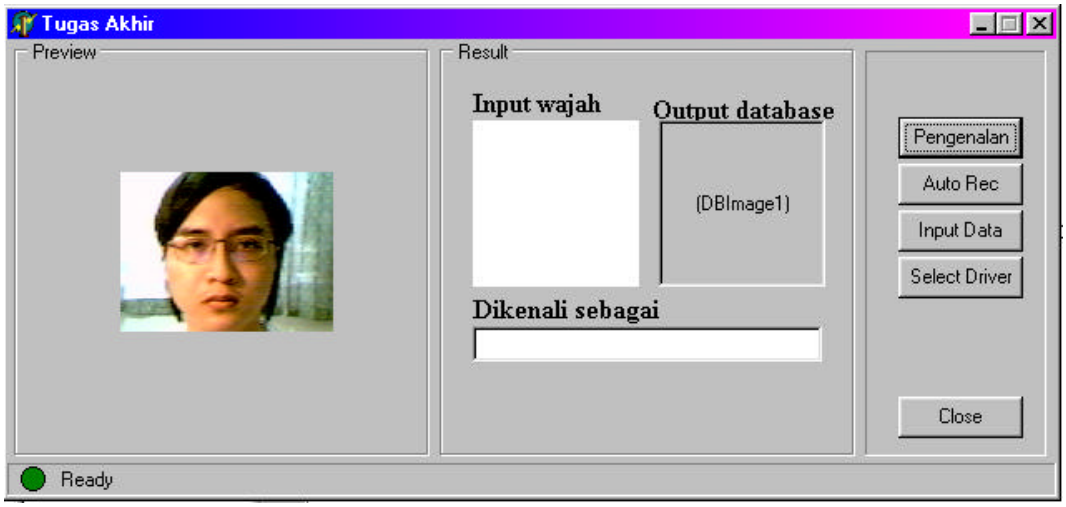
Software Tugas Akhir ini dirancang berbasis pemrograman Borland *Delphi 5.0*. Alasan digunakannya *software* ini, karena bahasa pemrogramannya yang relatif sederhana (bahasa pemrogaman tingkat tinggi), memiliki kecepatan dan ketepatan akses yang cukup tinggi, memiliki kemudahan dalam pengolahan *database*, serta dapat mengeksekusi komponen *dsgnintf* yang diperlukan untuk proses *capture image*. Kesulitan yang terdapat pada bahasa *Delphi* ini, adalah karena tidak adanya operasi-operasi *internal* dalam perhitungan matriks, contohnya mencari rata-rata matriks, invers, transpose, dan sebagainya. Kesulitan lainnya adalah belum adanya komponen untuk mengendalikan webcam; misalnya, bagaimana agar webcam dapat meng–capture gambar dan disimpan pada harddisk, bagaimana menampilkan *preview* sehingga dapat dilihat wajah orang yang akan di–*recognize* sebelum proses *recognition* dijalankan (mengatur posisi wajah sebelum proses dijalankan). Semua *procedure* untuk melakukan operasi matriks dan kendali *webcam* harus dibuat sendiri.

Struktur *software* ini terbagi menjadi 2 bagian penting, yaitu program *capture gambar*, dan program *training*. Program *capture gambar* mengendalikan webcam, antara lain: pemilihan driver (jika pada komputer diinstall *driver webcam* lebih dari satu jenis / tipe), *preview* gambar, serta pengambilan gambar (*image*) untuk di–*training*.

Program *training* berisi operasi-operasi matematika yang berfungsi untuk memasukkan *image-image training*, mengolah *image-image* tersebut, serta

mengurutkannya. Hasil yang ingin dicapai adalah mendapatkan data-data *training* untuk disimpan ke dalam file dan digunakan pada proses *recognition*.

Saat program pertama kali dijalankan, akan keluar tampilan seperti gambar 3.2.



Gambar 3.2. Tampilan awal *Program Recognition*

Dalam tampilan *main menu* ini terdapat 5 buah tombol. Tombol *Pengenalan* berfungsi untuk mengaktifkan program *training* dan *program recognition* sekaligus menampilkan hasil *recognition*. Tombol *Auto Rec* untuk melakukan proses *recognition* secara otomatis (tidak perlu berulang – ulang menekan tombol *Pengenalan*). Tombol *Input Data* berfungsi untuk memasukkan data baru ke dalam data base (berupa *image* dan nama orang yang baru). Tombol *Select Drive* berfungsi untuk memilih *driver webcam*; jika pada komputer diinstall webcam lebih dari satu jenis atau tipe. Tombol *Close* berfungsi untuk menghentikan program *recognition* dan keluar (exit) dari program *recognition*.

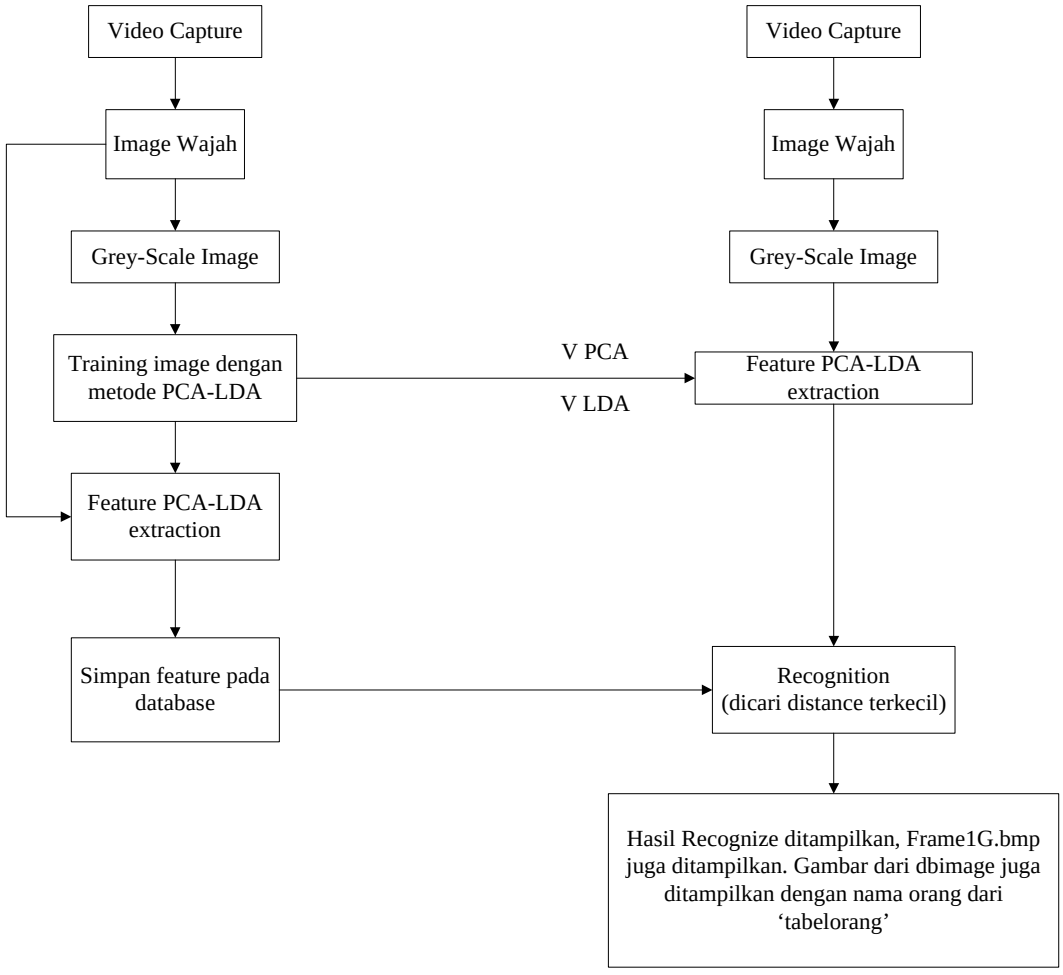
3.3.1. Program *Capture Gambar*

Proses *capture* gambar ini dikerjakan pada setiap kali proses *input data* dan proses *training* diaktifkan. Maksudnya, setiap kali tombol *Input Data*, tombol *Pengenalan*, dan tombol *Auto Rec* diaktifkan; proses *capture* gambar pasti diaktifkan juga. Saat proses *training* atau proses *capture* gambar berlangsung ditandai dengan lingkaran sebelah kiri bawah yang berkedip – kedip merah-

kuning. Berikut ini adalah cuplikan listing program yang berfungsi untuk meng-capture [2] gambar :

```
Video1.SingleImageFile:='Frame'+IntToStr(i)+'.bmp';
Video1.GrabFrameNoStop;
```

Image yang baru di-*capture* akan disimpan pada direktori dimana program face *recognition* dijalankan dan diberi nama ‘Frame1.bmp’. ‘i’ adalah variabel yang berfungsi sebagai *counter*. Saat *counter* mencapai nilai dua, maka program akan berhenti melakukan *capture* gambar, dan *counter* direset. Karena itulah, pada prakteknya file yang tersimpan hanya sampai ‘Frame1.bmp’. Perintah ‘*+IntToStr(i)*’ berfungsi untuk mengubah ‘i’ yang bertipe integer menjadi bertipe string.

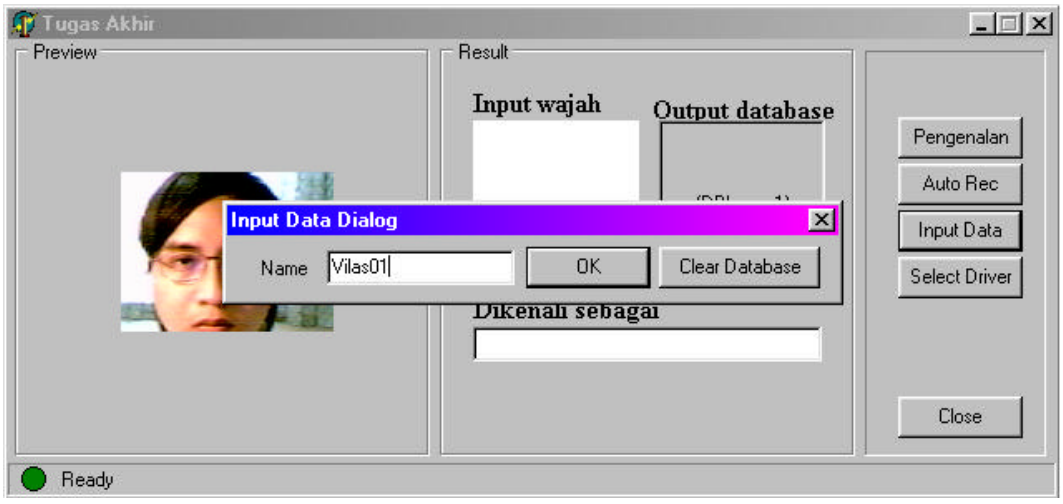


Gambar 3.3. Proses Training dan Recognition Secara Umum

[2] http://www.delphi32.com/vcl/lists/scl_n_4.asp

Gambar 3.4 berikut ini akan menunjukkan beberapa tampilan window pada saat program dijalankan.

Gambar 3.4. Tampilan Awal Program



Gambar 3.5. Tampilan Saat Memasukkan Nama

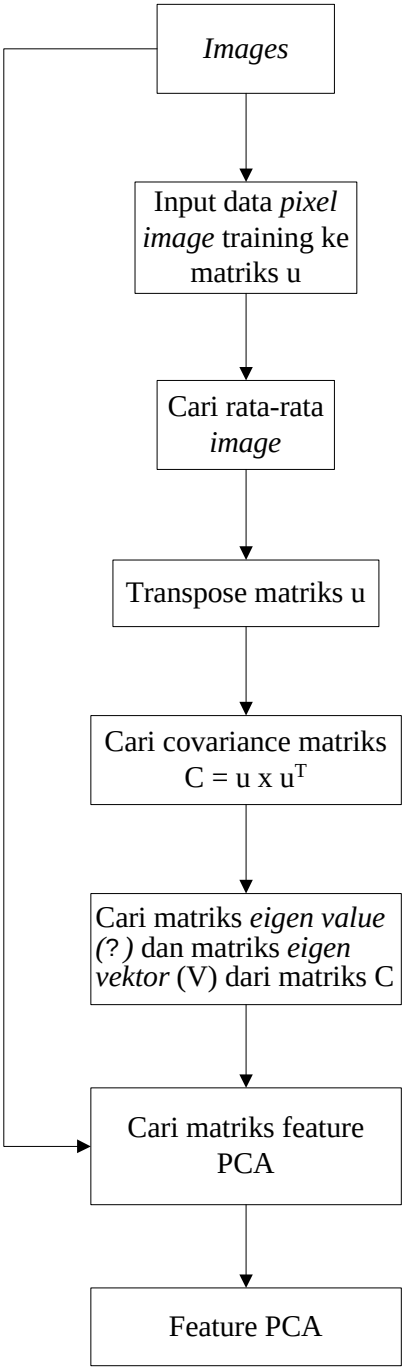
MatRataRata" yang disimpan pada file *Rata2TotalPCA.txt*.

2. Langkah selanjutnya adalah mencari *covariance matrix* dari metode PCA. Proses yang berlangsung dalam pencarian c*ovariance matrix* ini adalah :
    - Program memasukkan data masing-masing *image training* ke dalam matriks. Hasilnya adalah matriks berdimensi 10.000 x m, dimana m adalah jumlah *image training*. Hasilnya disimpan pada matriks "*MatrixKeseluruhan"*.

VPCA". Masing-masing berdimensi m x 1 dan m x m.

3. Setelah mencari *covariance matrix* dari metode PCA, maka dicari *feature* dari PCA berdasarkan *image-image training* yang sudah di – capture. Proses yang berlangsung selama pencarian *feature* PCA ini adalah :
    - Program akan mengurangkan data masing-masing *image training* yang tersimpan pada matriks "*MatrixKeseluruhan*" dengan rata-rata total PCA (û). Hasil perhitungan akan mengubah nilai "*MatrixKeseluruhan*". Matriks menjadi berdimensi 10.000 x m.
    - Hasil dari matriks di atas dikalikan dengan matriks "*VPCA*" yang berdimensi m x m. Hasil perkalian ini disimpan pada matriks "*VExpand*", yang berdimensi 10.000 x m. Matriks *VExpand*" ini disimpan pada file *V_PCA.txt*.
    - Matriks "*MatrixKeseluruhan*" yang sudah dikurangkan dengan *MatRataRata*" di atas kemudian ditranspose. Hasilnya adalah *MatImageT*" yang berdimensi m x 10.000. Matriks ini dikalikan dengan matriks "*VExpand*", untuk menghasilkan matriks "*FeaturePCA*" berdimensi m x m. Matriks ini disimpan ke file *FeaturePCA.txt*.

- Matriks "*FeaturePCA*" kemudian ditranspose agar dapat dipergunakan oleh metode LDA. Hasilnya disimpan pada matriks "*FeaturePCATranspose*". Gambar 3.7 adalah blok diagram proses *training* PCA



Gambar 3.7. Blok Diagram Proses T*raining* PCA

4. Langkah selanjutnya adalah mencari rata-rata total dari LDA. Tadi telah dijelaskan, bahwa *feature* dari PCA akan berfungsi sebagai input bagi LDA. *Feature* PCA yang sudah dihitung akan digunakan sebagai input metode LDA. Bagi metode LDA, seakan – akan ada m buah *image training*; masing-masing berdimensi m x m. Perhitungan yang dilakukan adalah mencari rata-rata dari matriks "*FeaturePCATranspose*". Hasil perhitungan disimpan dalam matriks *MatRataRataLDA*" yang berdimensi m x 1.

5. Selanjutnya dicari nilai rata-rata tiap kelas dari LDA. Proses yang berlangsung adalah :
   - Program akan mencari rata-rata dari kelas ke-i. Proses ini dibantu dengan nilai dari *field* 'Dari' dan 'Sampai' pada tabel *tabelorang* seperti yang telah dijabarkan di sub-bab 3.1.2. Sebagai contoh, pada sub-bab 3.1.2, program akan dibuat suatu matriks bernama "*MatRataRataKelasTemp*", dan diisi dengan data-data dari matriks "*FeaturePCATranspose"* dari kolom ke 0 sampai dengan 0. Matriks "*MatRataRataKelasTemp*" ini kemudian dicari rata-ratanya dan dimasukkan ke kolom I dari matriks "*MatRataRataKelas*", sebagai rata-rata kelas I. Proses ini diulang sebanyak *p* kali, dimana p adalah jumlah record yang ada pada tabel *tabelorang*, yang juga menyatakan jumlah kelas yang ada pada *training*. Hasil akhir dari matriks "*MatRataRataKelas*" adalah matriks berdimensi m x p

6. Langkah selanjutnya adalah mencari Sb (*Between Class Scatter Matrix*). Proses yang dikerjakan adalah :
   - Dibuat sebuah matriks "*Bobot*" berdimensi 1 x c, yang diisi dengan nilai yang terdapat pada *field* 'JumlahSample' pada tabel *tabelorang*.

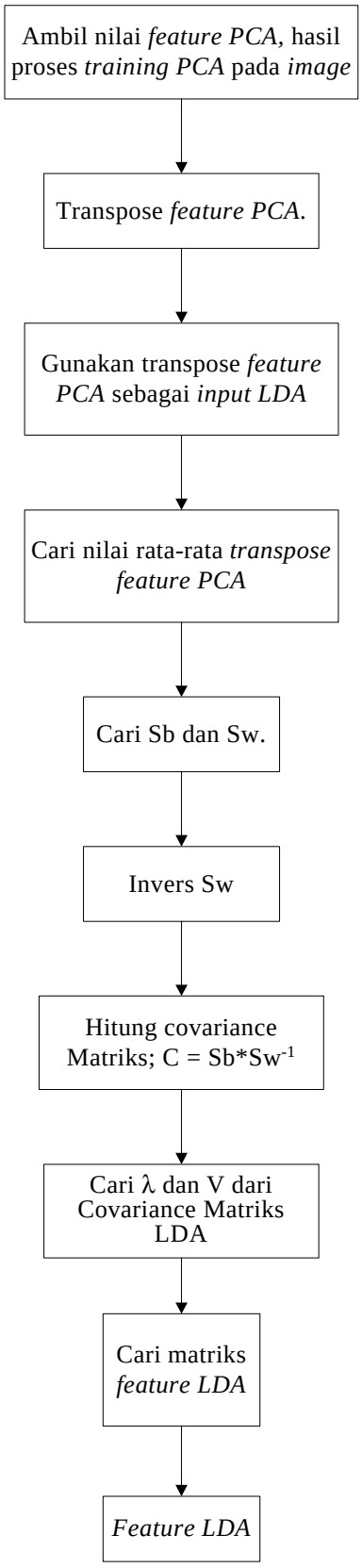
- Kemudian program akan mengurangkan nilai-nilai pada matriks "*MatRataRataKelas*" dengan nilai-nilai pada matriks "*MatRataRataLDA*". Hasil perhitungan disimpan pada matriks *MatRata2KelasMinRata2Total*" yang berdimensi m x c.
- Kemudian matriks "*MatRata2KelasMinRata2Total*" ini di-*transpose*, hasilnya akan disimpan pada matriks "*MatRata2KelasMinRata2TotalT*" yang berdimensi c x m.
- Matriks Sb dihasilkan dengan mengalikan nilai pada matriks "*MatRata2KelasMinRata2TotalT*" dengan nilai pada matriks "*MatRata2KelasMinRata2Total*". Hasil perhitungan disimpan pada matriks "*Sb*".
- Matriks "*Sb*" ini kemudian dikalikan dengan bobotnya masing-masing. Bobot disini berarti jumlah *image training* yang dimiliki oleh kelas yang bersangkutan. Caranya adalah seperti contoh di atas, mengalikan kolom ke-0 sampai 0 dengan bobot I, kolom ke – 1 sampai 1 dengan bobot II.

7. Setelah matriks "Sb" dihasilkan, selanjutnya dicari matriks "Sw" (*Within Class Scatter Matrix*). Proses yang berlangsung adalah :
    - Mengurangkan data masing-masing *image training* dengan rata-rata kelas dimana *image training* itu berada. Pada proses perhitungan ini tidak perlu lagi digunakan counter yang terdapat pada tabel *tabelorang*, cukup mengurangkan data pada matriks "*FeaturePCA*" dengan data pada matriks "*MatRataRataKelasExpand*". Hasil perhitungan disimpan pada *SwTemp*".
    - Lalu hitung transpose dari matriks "*SwTemp*" dan disimpan pada matriks "*SwTempTranspose*".
    - Sw didapat dari perkalian antara matriks "*SwTemp*" dan matriks "*SwTempTranspose*". Hasil perhitungan disimpan pada matriks "*Sw*".

LambdaLDA" dan "*VLDA*". Masing-masing berdimensi m x 1 dan m x m.

- Matrix "*VLDA*" ini disimpan pada file *V_LDA.txt*

9. Langkah yang berikutnya adalah mencari *feature* LDA. Proses yang berlangsung adalah :
    - Meng-*copy* isi dari matriks "*FeaturePCATranspose*" ke matriks "*FeaturePCAUntukLDA*". Hal ini perlu dilakukan agar nilai dalam matriks "*FeaturePCATranspose*" tidak berubah.
    - Proses mengurangi isi matriks "*FeaturePCAUntukLDA*" dengan isi matriks "*MatRataRataLDA*".
    - Hitung transpose dari matrix "*FeaturePCAUntukLDA*" dan disimpan pada matrix "*FeaturePCAUntukLDATranspose*".
    - Matrix ini kemudian dikalikan dengan matriks "*VLDA*", untuk mendapatkan matriks "*FeatureLDA*", yang berdimensi m x m.
    - Data-data pada matriks "*MatRataRataLDA*", "*VLDA*", dan "*FeatureLDA*" akan disimpan ke file untuk kemudian bisa dipanggil kembali.
    - Dari sini sudah bisa dilakukan proses *recognition* dengan metode PCA + LDA. (metode yang digunakan untuk proses recognition pada Tugas Akhir ini; karena telah terbukti bahwa perpaduan metode PCA yang dilanjutkan dengan metode LDA prosesnya paling cepat dengan hasil yang cukup presisi).

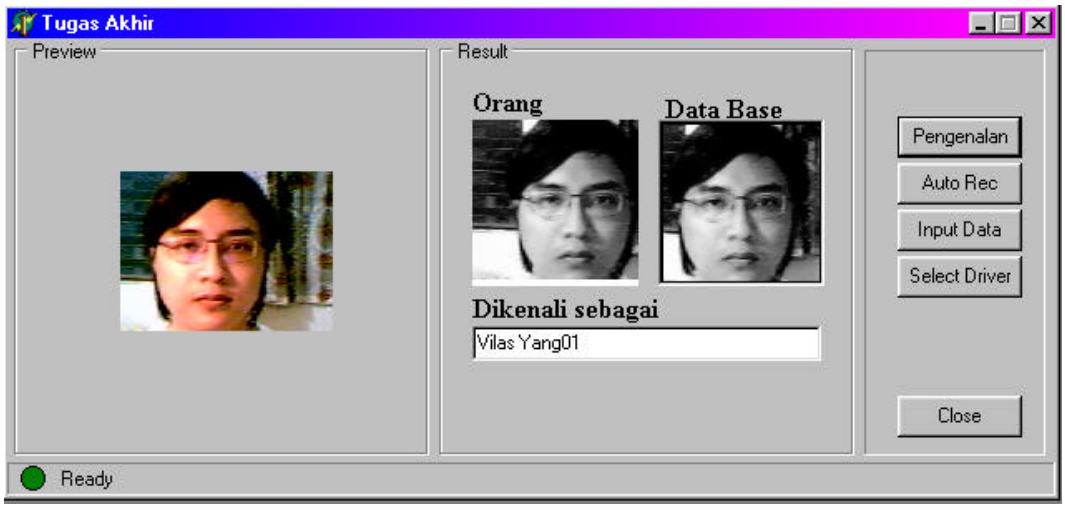


Gambar 3.8. Blok Diagram Proses *Training* PCA + LDA

10. Hingga sampai pada tahap ini; dapat dianggap bahwa proses training sudah selesai. Berikutnya adalah proses *recognition*. Pada proses recognition,hal yang pertama di kerjakan adalah meng-*upload* data yang telah disimpan dari harddisk ke memori komputer. Data-data itu yang di – up load adalah:
    - Rata-rata total PCA
    - Rata-rata total LDA
    - *Eigen Vector* PCA
    - *Eigen Vector* LDA
    - *Feature* PCA
    - *Feature* LDA

    Proses *upload* ini akan memakan waktu beberapa saat. Setelah selesai proses, berarti matriks-matriks yang akan digunakan dalam operasi *recognition* sudah selesai dibentuk (walau tidak ditampilkan di monitor, tapi disimpan pada memori).
11. Setelah matriks – matriks yang diperlukan telah terbentuk, maka program siap untuk melakukan proses *recognition*. Proses *recognition* pada Tugas Akhir ini menggunakan gabungan metode PCA dan LDA. Tahapan – tahapan yang dilakukan oleh software pada proses recognition ini dalah sebagai berikut:
    - Memasukkan data tiap *pixel* dari *image test* ke dalam matriks "*MatI*".
    - Mengurangkan nilainya dengan nilai rata-rata total PCA yang terdapat pada matriks "*MatRataRata*". Hasilnya disimpan pada matriks "*MatIMinU*".
    - Mencari transposenya dan menyimpan di matriks "*MatITranspose*".
    - Hasil transpose itu kemudian dikalikan dengan matriks "*VPCA*", dan didapat *feature* dari *image test*, disimpan pada matriks "*Featur*".
    - Dicari transpose dari matriks "*Featur*" dan disimpan pada matriks "*FeaturT*".

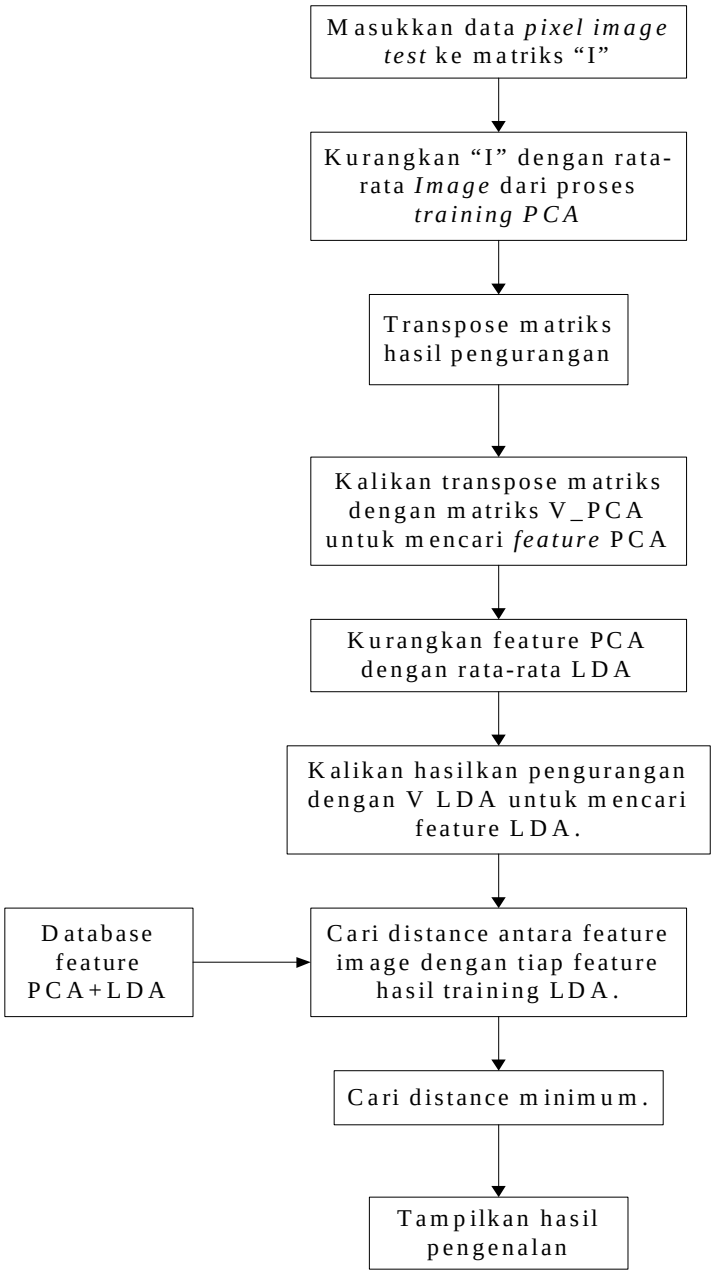
- Matriks "*FeaturT*" ini kemudian dikurangkan dengan matriks "*MatRataRataLDA*", kemudian ditranspose menjadi matriks *MatITransposeLDA*".
- Matriks ini dikalikan dengan matriks "*VLDA*", untuk mendapatkan *feature image* dalam LDA, disimpan pada matriks "*FeaturLDA*".
- Dicari matriks "*DistanceDari*", yang didapat dari k-*feature* pertama matriks "*FeaturLDA*". Nilai k didapat dari jumlah *feature* yang diambil.
- Matriks "*DistanceSampai*" didapatkan dari matriks "*FeatureLDA*" yang diambil sebanyak k-*feature* pertama, seperti di atas.
- Dihitung *distance* dari *image test* terhadap semua *image training* dengan rumusan *distance* yang telah dijabarkan di bab II.
- Program akan mencari nilai *distance* minimum dari semua nilai yang ada, kemudian menampilkan nama orang yang dimaksud dan *image training* yang dianggap memiliki *distance* minimum. Contoh hasil dari proses recognition adalah seperti gambar 3.9.



Gambar 3.9.Tampilan Saat Penekanan Tombol *Periksa* Selesai Proses .

Pada *GroupBox* sebelah kanan (window *Result*); gambar sebelah kiri adalah hasil capture webcam yang terbaru (diberi keterangan ‘Orang’). Sedangkan gambar sebelah kanan adalah gambar yang tersimpan pada data base. (data yang tersimpan dalam bentuk BLOB file).

Proses perhitungan yang panjang diatas dapat dilihat pada blok diagram gambar 3.10:



Gambar 3.10. Blok Diagram Proses Recognition PCA+LDA.