BAB II
TEORI PENUNJANG

2.1 Citra

Sebuah citra adalah kumpulan piksel yang disusun dalam larik 2 dimensi. Indeks baris dan kolom (x,y) dari sebuah piksel dinyatakan dalam bilangan bulat. Piksel adalah unsur gambar atau representasi sebuah titik terkecil dalam sebuah gambar grafis yang dihitung per inchi. Piksel (0,0) terletak pada sudut kiri atas pada citra, indeks *x* bergerak ke kanan dan indeks *y* bergerak ke bawah. Konvensi ini dipakai merujuk pada cara penulisan larik yang digunakan dalam pemrograman komputer.

2.1.1 Citra Biner

Citra biner (*binary image*) adalah citra yang hanya mempunyai 2 nilai derajat keabuan yaitu hitam dan putih. Piksel objek bernilai 1 dan piksel *background* bernilai 0. Pada waktu menampilkan gambar, 0 adalah putih dan 1 adalah hitam. Karena hanya ada 2 kemungkinan nilai pada setiap piksel, maka yang diperlukan hanya 1 bit per piksel. Citra seperti ini sangat efisien untuk penyimpanan. Citra seperti ini bisa dimanfaatkan untuk representasi biner dari teks, tanda tangan, sidik jari, atau rancangan arsitektur.

2.1.2 Citra Grayscale

Citra abu-abu (citra *grayscale*) merupakan citra yang memiliki nilai intensitas dari 0 – 255, dimana 0 adalah putih dan 1 adalah hitam sedangkan nilai diantaranya merupakan level keabuan. Rentang nilai intensitas 0 (hitam) sampai 255 (putih) berarti bahwa setiap piksel dapat direpresentasikan oleh 8 bit, atau 1 *byte*. Citra *grayscale* bisa digunakan untuk merepresentasikan citra medis (sinar x), tulisan/buku, dan lain-lain. Citra *grayscale* dengan 256 level abu-abu dapat digunakan untuk mengenali kebanyakan objek.

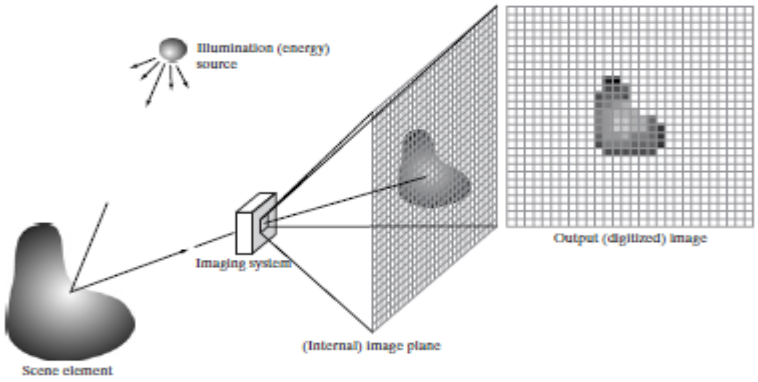
2.1.3 Citra RGB

Citra RGB (*Red, Green, Blue*) merupakan sebuah citra yang dimana setiap pikselnya memiliki suatu warna khusus. Warna RGB merupakan model warna *aditif* yaitu ketiga berkas cahaya yang ditambahkan bersama-sama dengan menambahkan panjang gelombang untuk membuat spektrum warna akhir. Pada model warna RGB setiap piksel mengandung minimal 24 bit data, dengan komposisi yaitu 8 bit untuk merah, 8 bit untuk hijau, dan 8 bit untuk biru. Jika setiap komponen warna tersebut memiliki rentang intensitas 0 - 255, maka terdapat sejumlah 255^3 = 16.777.216 kemungkinan jenis warna pada citra ini. Karena dibutuhkan 24 bit per piksel, maka citra ini disebut pula dengan citra warna 24 bit.

2.2 Pengolahan Citra Digital

Citra digital adalah citra kontinu yang diubah dalam bentuk diskrit, baik koordinat ruang maupun intensitas cahayanya. Proses digitalisasi terdiri dari 2 tahap yaitu pencuplikan (*sampling*) dan kuantisasi intensitas. Pengolahan citra digital adalah suatu pemrosesan sebuah citra yang bertujuan untuk memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin dalam hal ini komputer. Citra digital merupakan sebuah fungsi 2 dimensi f (x,y), di mana x dan y adalah koordinat spasial dan *amplitude* dari f pada sembarang pasangan koordinat (x,y) disebut *intensity* (intensitas) atau *gray level* (level keabuan) dari citra pada titik tersebut. Ketika x,y dan nilai intensitas dari f adalah semua terbatas, *discrete quantities*, kita sebut citra tersebut *digital image* (citra digital).

Citra digital terdiri dari sejumlah elemen tertentu, setiap elemen mempunyai lokasi dan nilai tertentu. Elemen-elemen ini disebut *picture elements*, *image elements, pels,* dan *pixels*. Piksel adalah istilah yang sudah digunakan secara luas untuk menyatakan elemen citra digital. Warna citra dibentuk oleh kombinasi citra 2-D individual. Misalnya, dalam sistem warna RGB, warna citra terdiri dari 3 komponen individu (*red, green, blue*).



Gambar 2.1 Pengolahan Citra Digital

2.3 Uang Koin

Uang koin rupiah yang digunakan dalam penelitian ini yaitu uang koin berbahan aluminium dimana setiap nominalnya memiliki ukuran jari-jari yang berbeda-beda. Jari-jari ini nantinya digunakan sebagai parameter utama pengenalan jenis nominal uang koin oleh aplikasi. Uang koin dengan nominal Rp 100 memiliki ukuran jari-jari 11,5 mm, uang koin dengan nominal Rp 200 memiliki ukuran jari-jari 12,5 mm, uang koin dengan nominal Rp 500 memiliki ukuran jari-jari 13,5 mm.

Tabel 2.1 Nilai Jari-jari Uang Koin Berbahan Aluminium

Jenis Nominal	Jari-jari
Rp 100	11,5 mm
Rp 200	12,5 mm
Rp 500	13,5 mm



Gambar 2.2 Uang Koin Rupiah Berbahan Aluminium

2.4 Kamera

Kamera adalah alat yang digunakan untuk menangkap citra, pada pembangunan aplikasi tugas akhir ini, kamera yang digunakan adalah kamera yang berasal dari sebuah *smartphone*.



Gambar 2.3 Kamera *Smartphone*

2.5 Tempat Untuk Menaruh Kamera Smartphone dan Alas Koin

Tempat untuk menaruh kamera *smartphone* yang dibuat memiliki ketinggian 50 cm dan terbuat dari kayu borneo. Sedangkan untuk alas tempat menaruh koin digunakan stiker polet berwarna hitam.



Gambar 2.4 Tempat Menaruh Kamera dan Alas Koin

2.6 Transformasi Hough

Hough transform pertama kali diperkenalkan oleh Paul Hough pada tahun 1962 untuk mendeteksi garis lurus. *Hough transform* adalah teknik transformasi citra yang dapat digunakan untuk mengisolasi atau dengan kata lain memperoleh fitur dari sebuah citra. Karena tujuan dari sebuah transformasi adalah mendapatkan suatu fitur yang lebih spesifik, *classical hough transform* merupakan teknik yang paling umum digunakan untuk mendeteksi objek yang berbentuk kurva seperti garis, lingkaran, elips dan parabola. Keuntungan utama dari transformasi Hough adalah dapat mendeteksi sebuah tepian dengan celah pada batas fitur dan secara relatif tidak dipengaruhi oleh derau atau *noise*.

Prinsip kerja dari metode transformasi Hough sendiri adalah dengan mencari bentuk geometri yang paling sesuai dengan kumpulan titik pada suatu citra. Jika objek yang dicari adalah berupa lingkaran, maka digunakan transformasi lingkaran Hough. Prosedur yang digunakan dalam mendeteksi lingkaran adalah sama dengan transformasi Hough pada objek garis, tapi

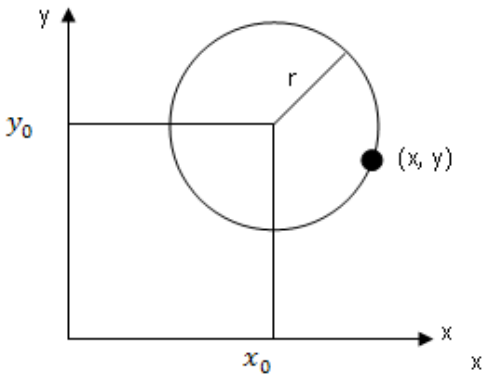
dikerjakan pada ruang dimensi yang lebih kompleks, yaitu dalam parameter ruang 3D (x_0,y_0,r) dimana x_0 dan y_0 merupakan koordinat pusat lingkaran dan r adalah jari-jari lingkaran seperti persamaan berikut :

$$(x-x_0)^2 + (y-y_0)^2 = r^2$$

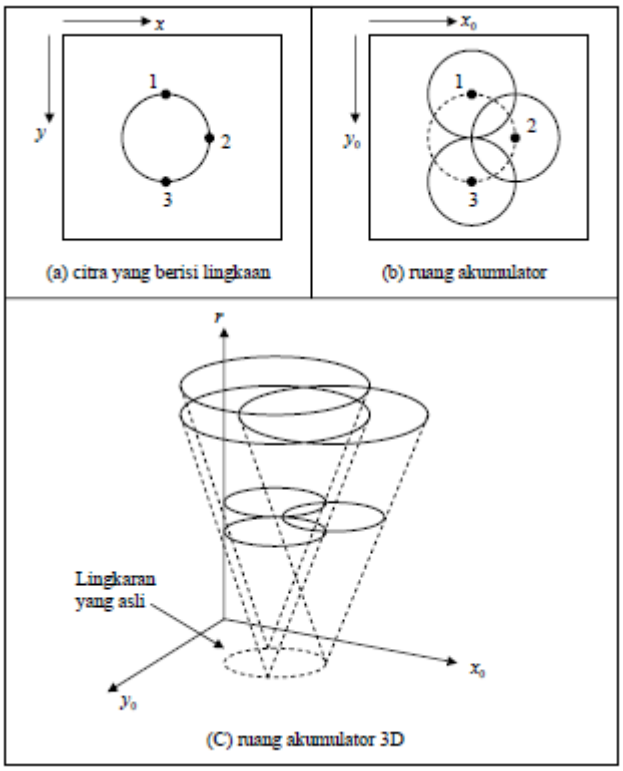
Sedangkan untuk vektor parameternya adalah :

$$\vec{p} = \begin{bmatrix} x_0 \\ y_0 \\ r \end{bmatrix}$$

Dan jika diilustrasikan dalam koordinat (x_0,y_0, r) adalah sebagai berikut :



Gambar 2.5 Peta Kartesian Koordinat Titik Tepi



Gambar 2.6 Ilustrasi Transformasi Hough

2.7 Deteksi Tepi

Deteksi tepi adalah suatu proses untuk menghasilkan tepi-tepi dari sebuah obyek pada sebuah citra, yang bertujuan untuk menandai bagian yang menjadi detail citra dan untuk memperbaiki detail citra yang kabur akibat dari proses akuisisi citra yang tidak baik. Suatu titik dikatakan sebagai sebuah tepi (edge) dari sebuah citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya.

2.7.1 Metode Canny

Salah satu algoritma deteksi tepi modern adalah deteksi tepi dengan menggunakan metode Canny. Deteksi tepi Canny dikembangkan oleh John F. Canny pada tahun 1986. Metode deteksi tepi Canny dibanding metode deteksi tepi

lainnya karena memiliki beberapa kelebihan dibanding metode deteksi tepi lain. Beberapa hal yang dapat dihasilkan metode deteksi tepi Canny antara lain :

- Kemampuan untuk meletakkan dan menandai semua tepi yang ada sesuai dengan pemilihan parameter-parameter konvolusi yang dilakukan. Sekaligus juga memberikan fleksibilitas yang sangat tinggi dalam hal menentukan tingkat deteksi ketebalan tepi sesuai yang diinginkan.

- Dengan Canny dimungkinkan dihasilkan jarak yang minimum antara tepi yang dideteksi dengan tepi yang asli.

- Hanya ada satu respon untuk tiap tepi. Sehingga mudah dideteksi dan tidak menimbulkan kerancuan pada pengolahan citra selanjutnya. Pemilihan parameter deteksi Canny sangat mempengaruhi hasil dari tepian yang dihasilkan. Seperti nilai standar deviasi *Gaussian* dan nilai *threshold*.

Dalam melakukan deteksi tepi Canny, terdapat beberapa tahapan yang dilakukan, yaitu sebagai berikut :

- Langkah pertama adalah menghilangkan *noise* yang terdapat pada citra dengan mengimplementasikan *Gaussian Filter*.

- Langkah kedua adalah melakukan deteksi tepi menggunakan salah satu operator deteksi tepi (dalam pembuatan tugas akhir ini digunakan metode Sobel) dengan melakukan pencarian secara horizontal (Gx) dan secara vertikal (Gy). *Mask* untuk Gx dan Gy adalah sebagai berikut :

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

- Hasil dari kedua operator tersebut digabungkan untuk mendapatkan hasil gabungan tepi vertikal dan horizontal. Gambar dengan gradien yang besar sering menunjukkan tepian yang cukup jelas. Namun tepian biasanya luas dan dengan demikian tidak dapat menunjukkan persis dimana tepian yang sebenarnya ini, arah tepian harus ditentukan dan disimpan. Berikut merupakan persamaan untuk mendapatkan nilai gradien (G) dan arah dari gradien tersebut (ɵ).

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan(\frac{G_y}{G_x})$$

- Langkah ketiga adalah melakukan *non-maximal supression* dari gradien citra untuk melokalisasi *edge* (tepi) secara presisi. *non-maximal suppression* bertujuan untuk membuang potensi gradien di suatu piksel dari kandidat *edge* (tepi) jika piksel tersebut bukan merupakan maksimal lokal pada arah *edge* (tepi) di posisi tersebut. Hal ini akan menghilangkan piksel yang dianggap bukan bagian dari tepi suatu citra.

- Langkah ke-empat adalah *hysteresis thresholding* untuk melakukan klasifikasi akhir. Canny menggunakan 2 *threshold* (*upper* dan *lower*). Jika sebuah gradien piksel lebih tinggi dibandingkan *upper threshold*, piksel tersebut akan dianggap sebagai sebuah *edge* (tepi). Jika sebuah nilai gradien piksel dibawah *lower threshold*, maka piksel tersebut akan diabaikan. Jika gradien piksel berada diantara 2 *threshold*, maka akan dianggap sebagai tepi jika terhubung pada sebuah piksel yang berada

diatas *upper threshold*. Rasio yang direkomendasikan untuk sebuah *upper* dan *lower* adalah 2:1 dan 3:1.

2.8 Bahasa Pemrograman C/C++

Bahasa pemrograman C adalah bahasa yang berada diantara bahasa tingkat rendah dan bahasa tingkat tinggi. Bahasa tingkat rendah artinya bahasa yang berorientasi pada mesin seperti *assembler* dan bahasa tingkat tinggi yang berorientasi pada manusia. Diciptakan oleh Brian W. Kernighan dan Denis M. Ritchi, pada tahun 1972. Penulisan program dalam bahasa C dilakukan dengan membagi dalam blok-blok, sehingga bahasa C disebut bahasa terstruktur. Berikut merupakan contoh penulisan program bahasa C.

```
#include <at89c51.h>
main ()
{
.....
.....
}
```

Gambar 2.7 Contoh Penulisan Sintak Bahasa Pemrograman C/C++

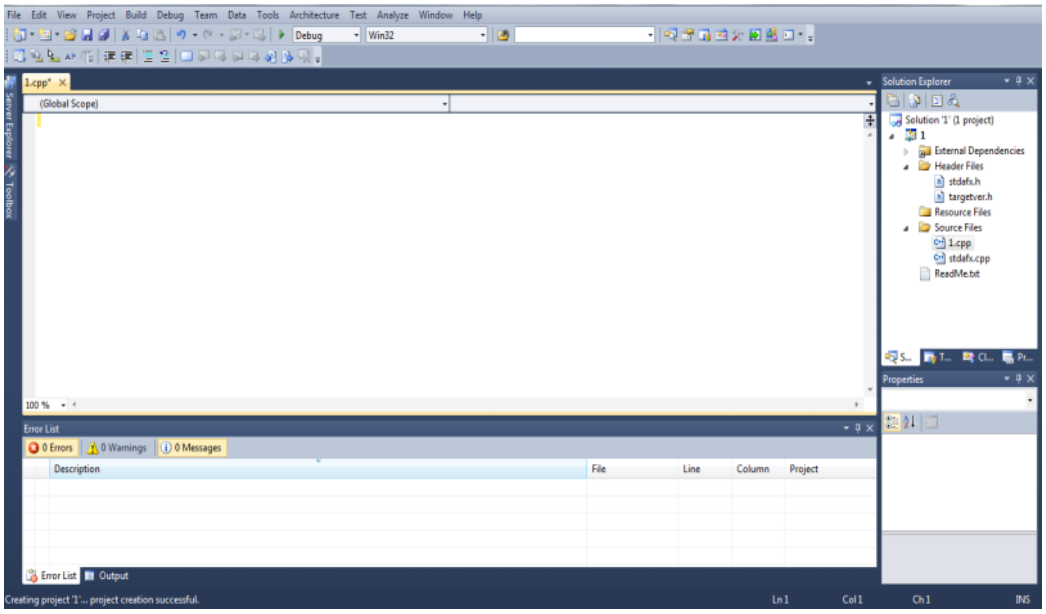
C++ adalah bahasa pemrograman komputer yang merupakan pengembangan dari bahasa C. Bahasa C++ diciptakan satu dekade setelah bahasa C yaitu pada tahun 1983 oleh Bjarne Stroustrup dari Laboratorium Bell, AT&T. Nama C++ sendiri diberikan oleh Rick Mascitti yang terinspirasi dari operator *increment* pada bahasa C. Bahasa C dijadikan sebagai landasan dari bahasa C++, mengingat bahasa C memungkinkan untuk diterapkan pada PC hingga *mainframe*, serta pada berbagai sistem operasi (DOS, UNIX, VMS, dan sebagainya). Untuk mendukung fitur pada C++, dibangun efisiensi dan sistem *support* untuk pemrograman tingkat rendah. Pada C++ ditambahkan konsep baru seperti *class* dengan sifat-sifatnya seperti *inheritance* dan *overloading*. Salah satu perbedaan yang paling mendasar dengan bahasa C adalah dukungan terhadap konsep pemrograman berorientasi objek.

Perbedaan antara bahasa pemrograman C dengan C++ meskipun kedua bahasa tersebut menggunakan sintak yang sama tetapi mereka memiliki perbedaan, C merupakan bahasa pemrograman prosedural, dimana penyelesaian suatu masalah dilakukan dengan membagi masalah tersebut kedalam sub masalah yang lebih kecil, C++ merupakan bahasa pemrograman berorientasi objek, untuk menyelesaikan masalah, C++ melakukan langkah pertama dengan menjelaskan beberapa *class* yang merupakan anak *class* yang dibuat sebelumnya sebagai abstraksi dari objek-objek fisik, *class* tersebut berisi keadaan objek, anggota-anggotanya dan kemampuan objeknya, setelah beberapa *class* dibuat kemudian masalah dipecahkan dengan *class*.

2.9 Microsoft Visual Studio 2010

Microsoft Visual Studio merupakan sebuah perangkat lunak yang dapat digunakan untuk pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi *console*, aplikasi *windows*, ataupun aplikasi web. Visual Studio mencakup *compiler*, *Software Development Kit* (SDK), *Integrated Development Environment* (IDE), dan dokumentasi (umumnya berupa MSDN Library). *Compiler* 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* di atas .NET Framework). Selain itu, Microsoft Visual Studio juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *windows mobile* (yang berjalan di atas .NET Compact Framework).



Gambar 2.8 Tampilan Microsoft Visual Studio 2010

2.10 OpenCV

OpenCV (*Open Computer Vision*) adalah sebuah *library* bersifat *open source* dibawah lisensi BSD (*Berkeley Software Distribution*) yang memiliki sejumlah algoritma *computer vision* atau pengolahan citra didalamnya. *Computer vision* sendiri merupakan salah satu cabang dari bidang ilmu pengolahan citra (*Image Processing*) yang memungkinkan komputer untuk dapat melihat seperti halnya manusia. Dengan *vision* tersebut komputer dapat mengambil keputusan, melakukan aksi, atau mengenali suatu objek. Beberapa implementasi dari *computer vision* adalah *face recognition, face detection, face/object tracking, road tracking*, dan lain sebagainya. OpenCV ditulis dalam bahasa C++. OpenCV pertama kali diluncurkan secara resmi pada tahun 1999 oleh Intel Research sebagai lanjutan dari proyek bertajuk aplikasi intensif dan saat ini dikembangkan oleh Willow Garage dan Itseez. *Library* OpenCV merupakan *library* lintas *platform*, bisa digunakan untuk bahasa pemrograman C, C++, Java, Phyton.

OpenCV memiliki struktur modular yang berarti setiap paket OpenCV terdiri dari sejumlah *library* statis dan *library* yang di-*share*. Beberapa modul yang terdapat dalam OpenCV adalah sebagai berikut :

- core

  Sebuah modul yang mendefinisikan struktur data standar dan fungsi standar *image processing*, termasuk array multi dimensional Mat dan fungsi standar yang digunakan oleh semua modul lainnya.

- imgproc

  Sebuah modul yang meliputi algoritma standar *image processing* meliputi *image filtering, image transformation, color space conversions*, dan lain sebagainya.

- video

  Sebuah modul analisis video yang meliputi estimasi gerakan, subtraksi latar belakang, dan algoritma *tracking* objek.

- calib3d

  Algoritma geometri *multiple-view* standar, *single* dan kalibrasi kamera stereo, estimasi gaya objek, algoritma korespondensi stereo, dan elemen rekonstruksi 3D.

- features2d

  Sebuah modul untuk keperluan *feature detectors*, *descriptors*, dan *descriptor matchers*.

- objdetect

  Sebuah modul yang meliputi deteksi objek dan algoritma pengenalan untuk objek standar.

- highgui

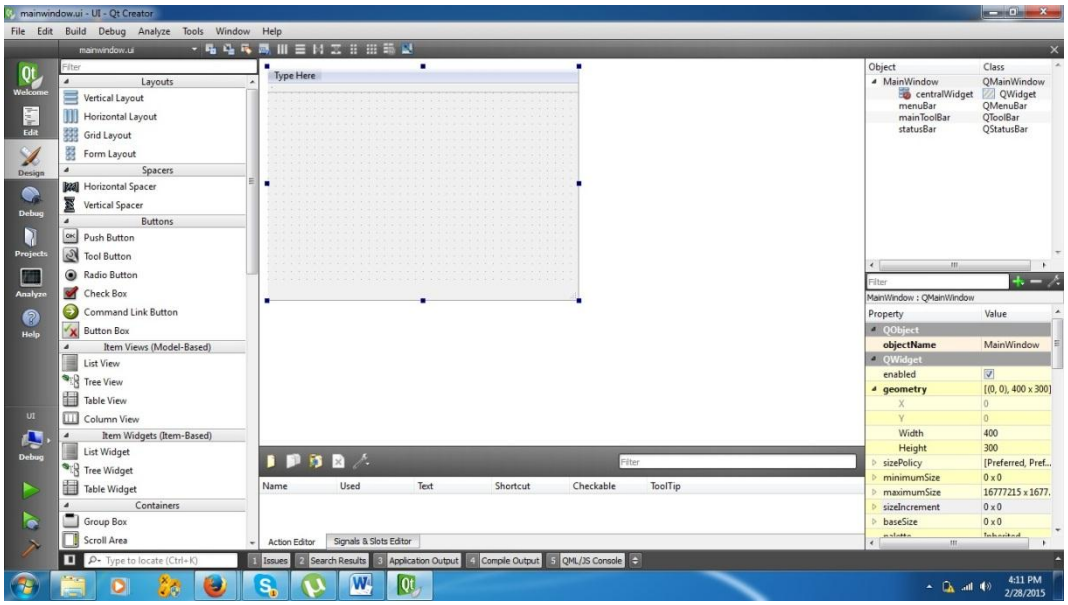
  Sebuah modul yang menyediakan kapabilitas *interface* pengguna yang sederhana, sejumlah *codec image* dan video, kapabilitas *image* dan *video capturing*, *manipulating image windows*, menangani *event mouse* dan *track bar*, dan lain sebagainya.

- gpu

  Algoritma GPU yang diakselerasi dari berbagai modul OpenCV.

2.11 QtCreator

QtCreator adalah sebuah IDE (*Integrated Development Software*) multi *platform*, termasuk juga sebuah *visual debugger* dan sebuah *integrated GUI layout and forms designer* untuk mendesain dan membangun *graphical user interface* (GUI) dari Qt widgets. Qt creator mendukung pembuatan aplikasi *desktop* (windows, linux, freeBSD dan mac os) dan aplikasi *mobile*. Namun dalam pembuatan aplikasi tugas akhir ini, QtCreator digunakan sebagai *software* untuk membangun GUI (*Graphical User Interface*) aplikasi. Qt designer adalah sebuah *tool* untuk mendesain dan membangun *graphical user interface* (GUI) dari qt widget. Hal ini memungkinkan untuk merancang *widget* dan mencobanya menggunakan tipe tampilan dan resolusi secara langsung di editor.



Gambar 2.9 Tampilan QtCreator Pada *Design Mode*