# Progetto RTOS

Berselli Werther

Di Blasi Fabrizio

# Titolo:

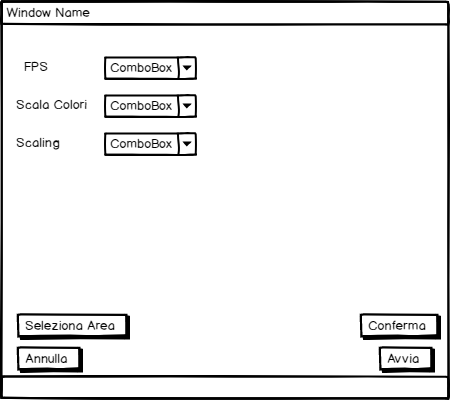
Image processing. Simulate a fixed camera looking at a given portion of an animated screen and write 4 periodic tasks performing different computations on the same image and displaying the results on other images. For example, task1 displays the histogram, task2 displays the image after applying a threshold, task3 displays the difference between consecutive frames, task4 displays the result of a filter. The user

must be able to vary some parameters of such tasks.

# Guida per l’utente:

Il programma ha il compito di analizzare uno stream video partendo da un riquadro tracciato a piacere dall’utente.

La schermata inziale è la seguente:



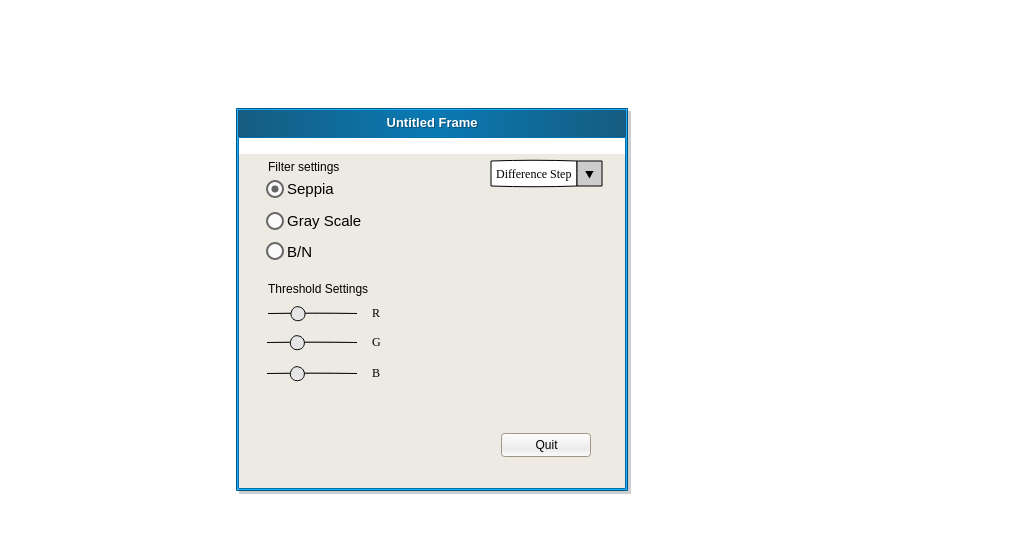
All’utente sarà permesso di scegliere alcuni iper parametri, che verranno utilizzati per la computazione.

All’inizio sarà “clickabile” solamente il tasto “**Seleziona Area**” che permetterà all’utente di selezionare la parte di schermo che si vuole processare.

Una volta selezionata la porzione di schermo, si potrà premere il tasto “**conferma**” il quale farà comparire a video le 5 aree di lavoro (videocamera, e 4 thread)

Infine col tasto **annulla** si ritorna alla scelta degli iper parametri, altrimenti si avvia la computazione.

Una volta premuto il tasto conferma, verrà visualizzato un’altro frame:



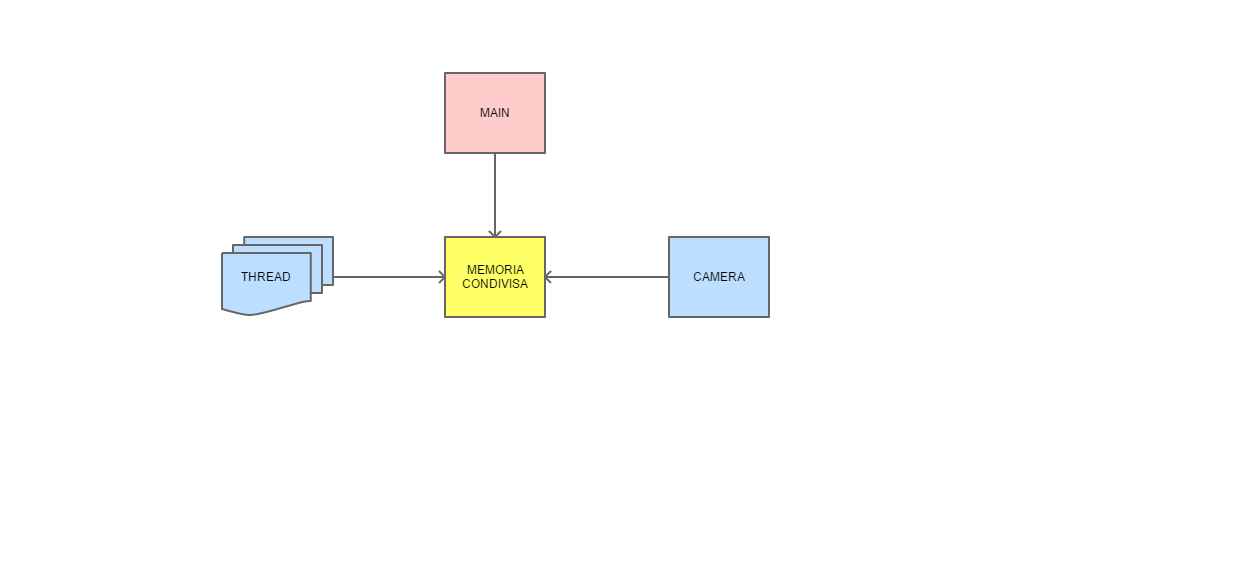
questo frame permette all’utente di impostare dei parametri a piacere.

È un’unico frame per tutti quanti i thread, poiché per avere una struttura più dinamica, utilizziamo le API messe a disposizione da openCV per mandare a video il fotogramma “ritoccato”.

Così facendo la gestione dello “scaling” risulta più semplice ed immediata.

È previsto un tasto di “quit” che fa arrestare tutta la computazione generale, senza dover ricorrere all’arresto da terminale

# Struttura del codice ed organizzazione della memoria condivisa:



Main:

* Inizializza il frame dell’interfaccia utente iniziale vista sopra
* Permette la scelta degli iper parametri nelle modalità già descritte
* Mette in esecuzione i thread

Camera:

* Gestisce la memoria condivisa tramite i semafori. Produce il frame, disattiva il suo semaforo ed attiva i semafori dei thread
* Inserisce l’ultimo frame all’interno della memoria condivisa
* Visualizza sul suo frame GUI il video che si sta analizzando
* Cattura il video ad “X” FPS scelti dall’utente nella fase iniziale

Memoria Condivisa:

* Semafori dei thread
* Mutex mutua esclusione
* FPS (iper parametro)
* Scala Colori (iper parametro)
* Scaling (iper parametro)
* Immagine catturata dalla camera **NON** RIDIMENSIONATA DI **DIMENSIONI FISSE**
  + Altezza
  + Larghezza
* Immagine catturata dalla camera **RIDIMENSIONATA DI DIMENSIONI FISSE**
  + Altezza
  + Larghezza

Thread:

* Parametri variabili impostabili dall’utente
* Variabili personali

## Organizzazione della memoria condivisa:

Il main mette in esecuzione il thread videocamera ed inizia a campionare il video.

Il fotogramma (campione) verrà inserito in due celle di memoria condivisa.

* Una cella conterrà il fotogramma a dimensioni originali non alterate
* L’altra conterrà il fotogramma a dimensioni ridotte in base alla volontà dell’utente

In questo modo è possibile far prendere il fotogramma ridimensionato per operazioni in cui è accettabile perdere informazioni, come ad esempio l’applicazione di un filtro.

Mentre quello a dimensioni originali sarà utilizzato da thread che faranno operazioni *non lossy* come ad esempio il tracciamento di un’istogramma

# Note di progetto:

Il progetto sarà organizzato in più file sorgenti .cpp e più librerie .h

Per i task periodici si utilizza la libreria “ptask” salvo eventuali errori di ubuntu, con la quale modelleremo i seguenti thread:

* Camera
* Thread istogramma
* Filtro a soglia
* Differenza tra fotogrammi

Grazie all’organizzazione su più file sarà possibile dividerci i compiti in modo più lineare, senza dover lavorare contemporaneamente sullo stesso file col rischio di conflitti.

La libreria conterrà le seguenti funzioni:

* Selezione riquadro
* Mostra frame glade

Variabili del main:

* Define
  + MAX THREAD
  + DIMENSIONI MASSIME RIQUADRO (800 x 600)
  + MAX\_FPS
* Fps
* Scala\_colori
* Scaling
* Ptask\_camera
* Ptask[i] thread
* Bool thread attivi

Si hanno dei vettori, poiché all’utente sarà possibile selezionare quali thread voler attivi e quali no.

## Funzioni richiamate dal main:

* On\_click\_seleziona\_area
* On\_click..

## File .c:

* Main
* Camera
* Histogram
* Filter
* Threshold
* Frame\_difference
* Image\_processing.h

# Convenzioni di scrittura adottate:

Oltre alle direttive indicateci a lezione utilizziamo:

* Commenti in italiano (da valutare se farli in inglese)
* Nomi variabili in inglese
* Nomi delle funzioni in inglese
* Nomi composti delle funzioni in questo modo “ funzioneDiProva ( int a )”
* Variabili con nomi composti unite da “ \_ “
* #DEFINE maiuscoli

compiti da dividerci:

- seleziona riquadro ( werther )

- camera ( fabrizio )

- mostrare frame glade (werther)

- main ( werther/fabrizio )

- istogramma (fabrizio)

- frame\_difference (werther)

- filter (fabrizio)

- threshold (werther)

frame glade :

- glade\_main (werther)

- settings (werther)

Nota bene:

camera: la camera visualizza il video da analizzare e ha un bottone per arrestare la computazione e tutto il programma. Gestire la cosa anche dal main

istogramma: non ha parametri, e analizza il video catturato dalla camera

filtro : permette di scegliere tra b/n, seppia, scala di grigi

threshold : 3 slider per R G B

difference: step di differenza tra frame. Il thread prende il primo frame, fa scattare il contatore ed andrà a “prendere” il frame successivo dopo “step volte”