Università di Parma

Corso di Laurea Magistrale in Ingegneria Informatica

Fondamenti di Visione Artificiale

a.a. 2019/19

PROVA PRATICA 29-01-2019

NOME:

COGNOME:

MATRICOLA:

WORKSTATION N°:

Non è consentito scambiarsi materiale via rete (ovviamente).

E' consentito 'uso di funzioni OpenCv di alto livello come `at()` e similari.

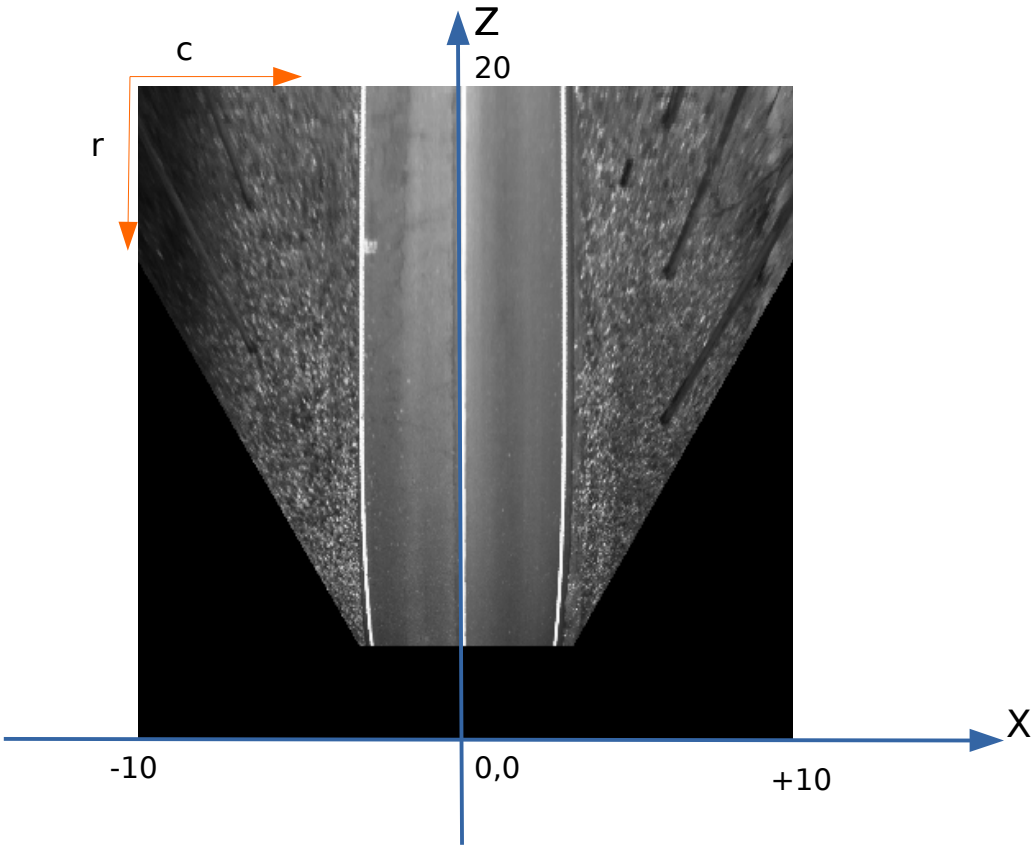
Salvare l'esame in un file COGNOME_MATRICOLA.zip.

FIRMA

ES1

Data l'immagine "mono.pgm" e i suoi parametri di calibrazione "mono.dat":

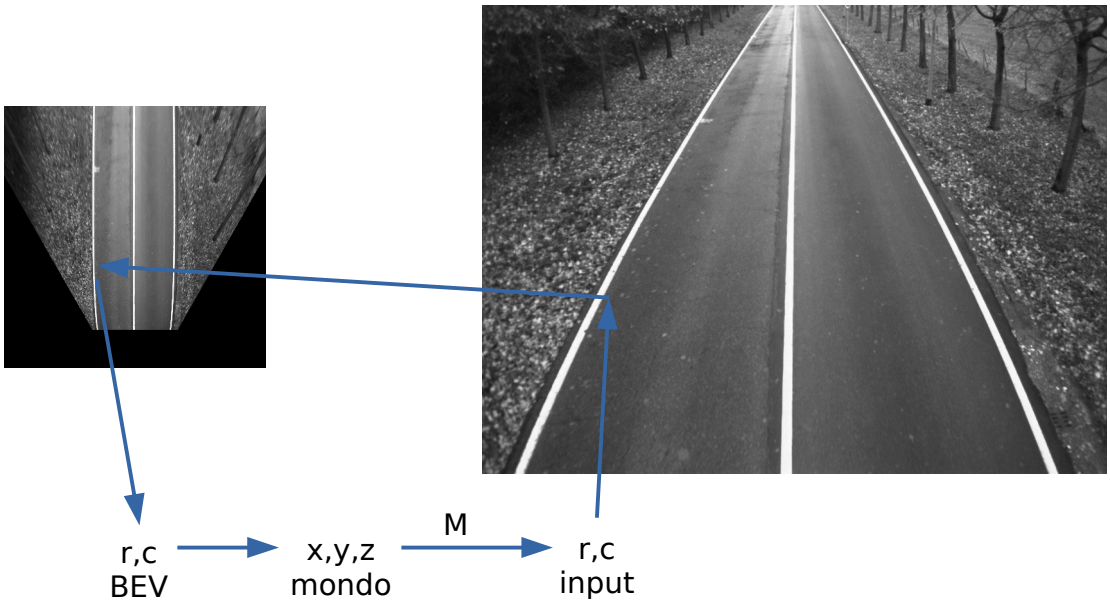
Creare la vista dall'alto (detta anche Bird Eye View, abbreviato BEV) della regione di mondo di fronte alla telecamera, limitandosi ad un'area di **20m x 20m**:



1. l'immagine BEV e' 400x400 pixel
2. ogni pixel dell'immagine BEV *corrisponde ad una particolare coordinata (x,y,z) del mondo*, con il *vincolo di y = 0 sempre*
3. l'angolo in *alto a sinistra* (r,c)=(0,0) dell'immagine BEV corrisponde al punto (x,y,z) = (-10, 0, 20) nel mondo
4. l'angolo in *alto a destra* (r,c)=(0,399) dell'immagine BEV corrisponde al punto (x,y,z) = (+10, 0, 20) nel mondo
5. l'angolo in *basso a sinistra* (r,c)=(399,0) dell'immagine BEV corrisponde al punto (x,y,z) = (-10, 0, 0) nel mondo
6. l'angolo in *basso a sinistra* (r,c)=(399,399) dell'immagine BEV corrisponde al punto (x,y,z) = (+10, 0, 0) nel mondo

HINTS:

1. nella BEV abbiamo 400 colonne per rappresentare 20 metri in orizzontale, 400 righe per rappresentare 20 metri in verticale. La *x* varia tra -10m e +10m, la *z* tra 0m e 20m. Come detto, la *y* e' sempre 0.
2. dato dunque un generico pixel (r,c) *dell'immagine BEV*, a che punto (x,y,z) nel *mondo* corrisponde?
3. data la calibrazione *dell'immagine di partenza "mono.pgm"*, calcolare la matrice di trasformazione prospettica mondo→immagine **M**
4. dato un punto mondo (x,y,z), con **M** posso ottenere la riga e la colonna (r_i,c_i) del corrispondente pixel sull'immagine **input** (trasformazione di coordinate omogenee da mondo a immagine...)
5. *copiare il pixel dell'immagine di input (r_i,c_i) cosi' trovato nel corrispondente (r,c) BEV*

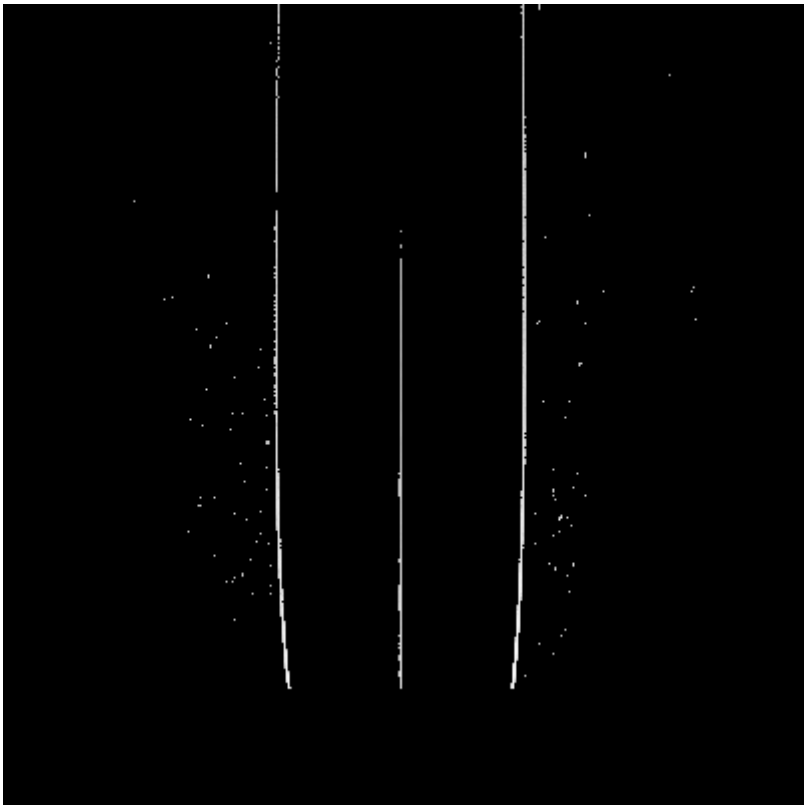


Esecuzione del codice di esempio:

./pratico ../images/mono.pgm ../calib/mono.dat

ES2

Partendo dalla BEV del passo precedente, come posso evidenziale le *linee bianche della carreggiata*? Ad esempio:



HINTS:

1. utilizzare una **convoluzione** con **kernel orizzontale** di 5 elementi `cv::Mat hlh = (cv::Mat_<float>(1, 5) << ?, ?, ?, ?, ?);`
2. Le linee bianche sono caratterizzate da un pattern dei toni di grigio del tipo: basso (asfalto), alto (linea bianca), basso (asfalto). Come dovrebbe essere fatto questo kernel per evidenziale *in particolare* questo tipo di andamento?
3. Far seguire l'applicazione del kernel ad una **binarizzazione**