Traffic Tracker

Generato da Doxygen 1.9.3

1 Indice dei tipi composti

1.1 Elenco dei tipi composti

Queste sono le classi, le struct, le union e le interfacce con una loro breve descrizione:

2 Indice dei file

2.1 Elenco dei file

Questo è un elenco dei file documentati con una loro breve descrizione:

3 Documentazione delle classi

3.1 Riferimenti per la classe camera::Camera

Classe che implemente e gestisce le telecamere per il riconoscimento e e il tracciamento degli oggetti.

```
#include <camera_threads.hpp>
```

Membri pubblici

- void **print_status** ()

  Stampa lo status della telecamera.
- void **start** ()

  Metodo che avvia la telecamera e l'analisi dei dati in input.
- void **stop** ()

  Metodo che ferma il thread della telecamera.
- void **camera_thread** ()

  Analisi e riconoscimento delle immagini in input Metodo che viene lanciato come thread separato in modalita' detached e che si occupa di recuperare l'input video dal source, di analizzarlo e di scrivere i risultati nel database.
- Camera (unsigned int id, std::string source_file, int traffico, int traffico_intenso, std::string name="", bool cuda=false, bool debug=false, int fps=-1, bool is_colored=false)

  Costruttore della clasee Camera.

3.1.1 Descrizione dettagliata

Classe che implemente e gestisce le telecamere per il riconoscimento e e il tracciamento degli oggetti.

3.1.2 Documentazione dei costruttori e dei distruttori

3.1.2.1 Camera()

```
camera::Camera::Camera (
        unsigned int id,
        std::string source_file,
        int traffico,
        int traffico_intenso,
        std::string name = "",
        bool cuda = false,
        bool debug = false,
        int fps = -1,
        bool is_colored = false )  [inline]
```

Costruttore della clasee Camera.

Parametri

in	*id*	L'id della telecamera, riferimento della telecamera nel database
in	*source_file*	La/il (pipe || source ID || file || data stream || file descriptor) che identifica l'input video
in	*traffico*	Il numero di veicoli a schermo sopra la quale si considera la strada trafficata
in	*traffico_intenso*	Il numero di veicoli sopra la quale si considera la strada molto trafficata
in	*name*	Il nome da associare alla telecamera (Nome fittizio solo per riconoscerla meglio). default: ""
in	*cuda*	Identifica se si intende utilizzare CUDA per effettuare i calcoli della rete neurale sulla GPU. default: false
in	*debug*	Identifica se si intende abilitare i messaggi di debug(in questa versione non sono ancora inplementati). default: false
in	*fps*	Identifica il limite massimo di fps a cui viene analizzato il file/source video. In caso di connessione diretta con telecamere lasciare a -1. default: -1
in	*is_colored*	Se vero lo status della telecamera viene stampato a colori e con caratteri UTF-8. In caso contrario viene stampato in ascii. default: false

La documentazione per questa classe è stata generata a partire dal seguente file:

- Core/includes/camera_threads.hpp

3.2 Riferimenti per la struct utilities::colors

`#include <hub_utilities.hpp>`

Attributi pubblici

- std::string **red** = "\033[31m"
- std::string **blue** = "\033[34m"
- std::string **green** = "\033[32m"
- std::string **magenta** = "\033[35m"
- std::string **cyan** = "\033[36m"
- std::string **yellow** = "\033[33m"
- std::string **defaul** = "\033[0m"

3.2.1 Descrizione dettagliata

Struct colors. Struct che contiene il codice ANSI dei colori

La documentazione per questa struct è stata generata a partire dal seguente file:

- Core/includes/hub_utilities.hpp

3.3 Riferimenti per la classe Database

Classe che implementa le funzionalita' richieste per interagire con il database.

`#include <database_utilities.hpp>`

3.3.1 Descrizione dettagliata

Classe che implementa le funzionalita' richieste per interagire con il database.

La documentazione per questa classe è stata generata a partire dal seguente file:

- Core/includes/database_utilities.hpp

3.4 Riferimenti per la classe Database::Database

Membri pubblici

- Database (std::string config_path)

  Costruttore della classe Database.
- bool check_conneciton ()

  Metodo che verifica se il Database e' raggiungibile.
- bool insert_registrazione (unsigned int id, unsigned int state)

  Inserisce una nuova registrazione nel Database.
- bool insert_telecamere (float lat, float lng)

  Inserische una nuova telecamera nel Database con ID autogenerato.
- bool insert_telecamere_by_id (int id, float lat, float lng)

  Inserische una nuova telecamera nel Database.
- bool start_telecamera (int id)

  Aggiorna lo stato della telecamera sul Database a ON.
- bool **stop_telecamera** (int id)
- int get_cameras_count ()

  Ritorna il numero di telecamere nel Database.
- std::vector< int > get_cameras_ids ()

  Funzione che ritona un std::vector contenente gli id delle telecamere presenti nel database.

3.4.1 Documentazione dei costruttori e dei distruttori

3.4.1.1 Database() `Database::Database::Database ( std::string config_path ) [inline]`

Costruttore della classe Database.

Parametri

in	*config_path*	Indirizzo in memoria del file di configurazione

3.4.2 Documentazione delle funzioni membro

3.4.2.1 check_conneciton() `bool Database::Database::check_conneciton ( ) [inline]`

Metodo che verifica se il Database e' raggiungibile.

Restituisce

true se il Database e' raggiungibile, false altrimenti

3.4.2.2 get_cameras_count() `int Database::Database::get_cameras_count ( ) [inline]`

Ritorna il numero di telecamere nel Database.

Restituisce

Il numero di telecamere inserite nel Database

3.4.2.3 get_cameras_ids() `std::vector< int > Database::Database::get_cameras_ids ( ) [inline]`

Funzione che ritona un std::vector contenente gli id delle telecamere presenti nel database.

Restituisce

std::vector contenente gli id delle telecamere

3.4.2.4 insert_registrazione()

```
bool Database::Database::insert_registrazione (
        unsigned int id,
        unsigned int state ) [inline]
```

Inserisce una nuova registrazione nel Database.

Parametri

in	*id*	L'id della telecamera che ha effettuato la rilevazione
in	*state*	Lo stato della registrazione

Restituisce

true se l'inserimento e' andato a buon fine, false altrimenti

3.4.2.5 insert_telecamere()

```
bool Database::Database::insert_telecamere (
        float lat,
        float lng ) [inline]
```

Inserische una nuova telecamera nel Database con ID autogenerato.

Parametri

in	*lat*	La latitudine della telecamera
in	*lng*	La longitudine della telecamera

Restituisce

true se l'inserimento nel Database e' andato a buon fine, false altrimenti

3.4.2.6 insert_telecamere_by_id()

```
bool Database::Database::insert_telecamere_by_id (
        int id,
        float lat,
        float lng )  [inline]
```

Inserische una nuova telecamera nel Database.

Parametri

in	*id*	L' ID della telecamera
in	*lat*	La latitudine della telecamera
in	*lng*	La longitudine della telecamera

Restituisce

true se l'inserimento nel Database e' andato a buon fine, false altrimenti

3.4.2.7 start_telecamera()

```
bool Database::Database::start_telecamera (
        int id )  [inline]
```

Aggiorna lo stato della telecamera sul Database a ON.

Aggiorna lo stato della telecamera sul Database a OFF.

Parametri

in	*id*	L'id della telecamera

Restituisce

true se l'operazione e' andata a buon fine, false altrimenti

La documentazione per questa classe è stata generata a partire dal seguente file:

- Core/includes/database_utilities.hpp

3.5 Riferimenti per la classe stringParser::Parser

Classe che implementa il manipolatore di stringhe.

`#include <string_parser.hpp>`

Membri pubblici

- int parser_init (std::string input)

  Funzione di parsing.
- int print_buffer ()

  Stampa il contenuto del buffer.
- std::queue< std::string > get_buffer ()

  Ritorna il buffer.
- **Parser** ()

  Costruttore della classe Parser Costruttore di default, non e' necessario compiere azioni al momento dell'inizializzazione di un nuovo oggetto Parser.
- ~**Parser** ()

  Distruttore dela classe Parser Il distruttre svuta i contenuto del buffer prima di eliminare l'oggetto. Questa azione non dovrebbe essere necessaria essendo il buffer inizializato staticamente, ma come si dice il divolo e' tanto fino e i puntatori son figli suoi.

3.5.1 Descrizione dettagliata

Classe che implementa il manipolatore di stringhe.

3.5.2 Documentazione delle funzioni membro

3.5.2.1 get_buffer() `std::queue< std::string > stringParser::Parser::get_buffer ( ) [inline]`

Ritorna il buffer.

Restituisce

Il buffer dell'oggetto allo stato attuale Funzione "getter" che ritorna il buffer dell'oggetto allo stato attuale

3.5.2.2 parser_init() `int stringParser::Parser::parser_init ( std::string input ) [inline]`

Funzione di parsing.

Parametri

in	*input*	La stringa in formato std::string di cui fare il parsing

Restituisce

0 se tutto e' andato a buon fine, -1 altrimenti Funzione che preso in input una stringa ne fa il parsing e salva il contenuto nella queue buffer dell'oggetto

3.5.2.3 print_buffer() `int stringParser::Parser::print_buffer ( ) [inline]`

Stampa il contenuto del buffer.

Restituisce

0 se tutto e' andato a buon fine, -1 altriemnti Funzione che stampa il contenuto della queue di buffer sullo standart output. Se la funzione e' andata a buon fine il buffer sara' identico a come era prima dell'invocazione, in caso di fallimento non e' invece garantita questa consistenza.

La documentazione per questa classe è stata generata a partire dal seguente file:

- Core/includes/string_parser.hpp

3.6 Riferimenti per la classe utilities::Prompt

Classe contenitore per tutti i metodi usati nell'interazione grafica con l'utente.

`#include <hub_utilities.hpp>`

Membri pubblici

- std::string get_prompt ()
  Visualizza il prompt a schermo e richiede in inputun comando all'utente.
- void **prompt_help** ()
  Stampa a schermo la guida ai comandi.
- void command_error (std::string command)
  Stampa il messaggio di errore per l'inserimento di un comando non riconosciuto.
- void camera_error (std::string command)
  Stampa il messaggio di errore per l'inserimento di un parametro non riconosciuto al comando 'camera'.
- void camera_status_error (std::string command)
  Stampa il messaggio di errore per l'inserimento di un parametro non riconosciuto al comando 'camera status'.
- void camera_start_error (std::string command)
  Stampa il messaggio di errore per l'inserimento di un parametro non riconosciuto al comando 'camera start'.
- void camera_stop_error (std::string command)
  Stampa il messaggio di errore per l'inserimento di un parametro non riconosciuto al comando 'camera stop'.
- void camera_help (unsigned int help_code)
  Stampa l'help.
- Prompt (bool colors=false)
  Costruttore per la classe Prompt.

3.6.1 Descrizione dettagliata

Classe contenitore per tutti i metodi usati nell'interazione grafica con l'utente.

3.6.2 Documentazione dei costruttori e dei distruttori

3.6.2.1 Prompt() `utilities::Prompt::Prompt ( bool colors = false ) [inline]`

Costruttore per la classe Prompt.

Parametri

in	*colors*	Specifica se il terminale in uso supporta i colori oppure no Inizializza un oggetto che stampera' usando i colori ed i caratteri Unicode oppure con colori e caratteri base

3.6.3 Documentazione delle funzioni membro

3.6.3.1 camera_error() `void utilities::Prompt::camera_error ( std::string command ) [inline]`

Stampa il messaggio di errore per l'inserimento di un parametro non riconosciuto al comando 'camera'.

Parametri

in	*command*	Il parametro non riconosciuto

3.6.3.2 camera_help() `void utilities::Prompt::camera_help ( unsigned int help_code ) [inline]`

Stampa l'help.

Parametri

in	*help_code*	Il codice del comando di cui stampare l'help

3.6.3.3 camera_start_error() `void utilities::Prompt::camera_start_error ( std::string command ) [inline]`

Stampa il messaggio di errore per l'inserimento di un parametro non riconosciuto al comando 'camera start'.

Parametri

in	*command*	Il parametro non riconosciuto

3.6.3.4 camera_status_error() `void utilities::Prompt::camera_status_error ( std::string command ) [inline]`

Stampa il messaggio di errore per l'inserimento di un parametro non riconosciuto al comando 'camera status'.

Parametri

in	*command*	Il parametro non riconosciuto

3.6.3.5 camera_stop_error() `void utilities::Prompt::camera_stop_error ( std::string command ) [inline]`

Stampa il messaggio di errore per l'inserimento di un parametro non riconosciuto al comando 'camera stop'.

Parametri

in	*command*	Il parametro non riconosciuto

3.6.3.6 command_error() `void utilities::Prompt::command_error ( std::string command ) [inline]`

Stampa il messaggio di errore per l'inserimento di un comando non riconosciuto.

Parametri

in	*command*	Il comando non riconosciuto

3.6.3.7 get_prompt() `std::string utilities::Prompt::get_prompt ( ) [inline]`

Visualizza il prompt a schermo e richiede in inputun comando all'utente.

Restituisce

Il comando inserito dall'utente Questa funzione e' impelementata usando la libreria GNU readline/readline.h per una migliore interazione con l'utente

La documentazione per questa classe è stata generata a partire dal seguente file:

- Core/includes/hub_utilities.hpp

4 Documentazione dei file

4.1 Riferimenti per il file Core/includes/camera_threads.hpp

```
#include "hub_utilities.hpp"
#include "database_utilities.hpp"
```

```
#include <array>
#include <cmath>
#include <cstdlib>
#include <exception>
#include <ios>
#include <iostream>
#include <thread>
#include <time.h>
#include <vector>
#include <fstream>
#include <string>
#include <chrono>
#include <opencv2/core.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/core/mat.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2/dnn.hpp>
#include <opencv2/core/types.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/core/utility.hpp>
#include <opencv2/core/hal/interface.h>
#include <opencv2/video/background_segm.hpp>
```

Composti

- class camera::Camera

  Classe che implemente e gestisce le telecamere per il riconoscimento e e il tracciamento degli oggetti.

Variabili

- std::vector< bool > camera::camera_stopper

4.1.1 Descrizione dettagliata

Autore

Andrea Valenzano

Data

2022

Copyright

AGPLv3: Affero General Public License version 3

Precondizione

Libreria per la gestione delle telecamere

4.1.2 Documentazione delle variabili

4.1.2.1 camera_stopper `std::vector<bool> camera::camera_stopper`

Vettore usato per fermare i thread delle telecamere

4.2 camera_threads.hpp

Vai alla documentazione di questo file.

```
1
9 /* Copyright (C) 2022  Andrea Valenzano
10  *
11  * This program is free software: you can redistribute it and/or modify
12  * it under the terms of the GNU Affero General Public License as published by
13  * the Free Software Foundation, either version 3 of the License, or
14  * (at your option) any later version.
15  *
16  * This program is distributed in the hope that it will be useful,
17  * but WITHOUT ANY WARRANTY; without even the implied warranty of
18  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  * GNU General Public License for more details.
20  *
21  * You should have received a copy of the GNU General Public License
22  * along with this program.  If not, see <http://www.gnu.org/licenses/>.
23  */
24
25 #ifndef CAMERA_THREADS_HPP
26 #define CAMERA_THREADS_HPP
27
28 #include "hub_utilities.hpp"
29 #include "database_utilities.hpp"
30
31 #include <array>
32 #include <cmath>
33 #include <cstdlib>
34 #include <exception>
35 #include <ios>
36 #include <iostream>
37 #include <thread>
38 #include <time.h>
39 #include <vector>
40 #include <fstream>
41 #include <string>
42 #include <chrono>
43
44 #include <opencv2/core.hpp>
45 #include <opencv2/highgui.hpp>
46 #include <opencv2/core/mat.hpp>
47 #include <opencv2/imgcodecs.hpp>
48 #include <opencv2/opencv.hpp>
49 #include <opencv2/dnn.hpp>
50 #include <opencv2/core/types.hpp>
51 #include <opencv2/imgproc.hpp>
52 #include <opencv2/videoio.hpp>
53 #include <opencv2/core/utility.hpp>
54 #include <opencv2/core/hal/interface.h>
55 #include <opencv2/video/background_segm.hpp>
56
57 using namespace cv;
58 using namespace  dnn;
59
60 namespace camera{
61
62     std::vector<bool> camera_stopper;
68     class Camera{
69
70         private:
71
72             unsigned int id;
73             std::string name;
74
75             bool is_colored;
76             utilities::colors colors;
77
78             std::string status;
```

```
79              bool CUDA;
80              bool DEBUG;
81              int FPS_TARGET;
82              std::string source_file;
83
84              int traffico;
85              int traffico_intenso;
86
87              time_t start_time;
88
89          public:
94              void print_status(){
95                  if(is_colored){
96                      std::cout << std::endl << (status=="active"?colors.green+"
    "+colors.defaul+"(":colors.red+"  "+colors.defaul+") << id << ") "<< name << std::endl;
97                      std::cout << " mode:    " << (CUDA?"GPU":"CPU") << std::endl;
98                      std::cout << " max FPS: "; FPS_TARGET==-1? std::cout << "Unlimited": std::cout <<
    FPS_TARGET; std::cout <<  std::endl;
99                      std::cout << " status:  " << (status=="active"?colors.green:colors.red) << status <<
    colors.defaul << std::endl;
100                     std::cout << " uptime:  " ;
101                     if(status == "active"){
102                         time_t uptime = time(NULL) - start_time;
103                         std::cout << (int)(uptime/60)/60 << " H : " << (int)uptime/60 << " M : " <<
    (int)uptime%60 << " S" << std::endl;
104                     } else {
105                         std::cout << std::endl;
106                     }
107                     std::cout << std::endl;
108                     return;
109                 }
110
111                 std::cout << std::endl << " - (" << id << ") "<< name << std::endl;
112                 std::cout << " mode:    " << (CUDA?"GPU":"CPU") << std::endl;
113                 std::cout << " max FPS: "; FPS_TARGET==-1? std::cout << "Unlimited": std::cout <<
    FPS_TARGET; std::cout <<  std::endl;
114                 std::cout << " status:  " << status << std::endl;
115                 std::cout << " uptime:  " ;
116                 if(status == "active"){
117                     time_t uptime = time(NULL) - start_time;
118                     std::cout << (int)(uptime/60)/60 << " H : " << (int)uptime/60 << " M : " <<
    (int)uptime%60 << " S" << std::endl;
119                 } else {
120                     std::cout << std::endl;
121                 }
122                 std::cout << std::endl;
123             }
124
129             void start(){
130                 if(camera_stopper.at(id) == true) return;
131                 camera_stopper.at(id) = true;
132                 status = "active";
133                 start_time = time(NULL);
134                 try{
135                     std::thread this_camera_thread(&Camera::camera_thread, this);
136                     this_camera_thread.detach();
137                 } catch(std::exception e){
138                     std::cout << e.what() << std::endl;
139                 }
140             }
141
146             void stop(){
147                 camera_stopper.at(id) = false;
148                 status = "stopped";
149             }
150
157             void camera_thread(){
158
159                 Database::Database database("database_config.yaml");
160
161                 database.start_telecamera(id);
162
163                 float FPS_TARGET_FREQ = 1.0/FPS_TARGET;
164                 std::string MODE;
165                 unsigned int FRAME_COUNT = 0;
166                 unsigned int CAR_COUT = 0;
167                 bool VISUALIZE = true;
168                 short FPS;
169                 float FPS_CURR_FREQ;
170
171                 int scrolling = 0;
172                 database.insert_registrazione(id, scrolling);
173
174                 Mat frame;
175
176                 // Tracking
177                 std::vector<std::array<int, 3> tracker_1;
```

```
178                 std::vector<std::array<int, 3» tracker_2;
179                 std::vector<std::array<int, 3» tracker_3;
180 
181 
182                 std::vector<int> classIds;
183                 std::vector<float> confidences;
184                 std::vector<Rect> boxes;
185 
186                 std::vector<std::string> classes;
187                 std::ifstream file("dnn_model/classes.txt");
188                 std::string line;
189                 while (std::getline(file, line)){
190                     classes.push_back(line);
191                 }
192                 file.close();
193 
194                 Net net = readNetFromDarknet("dnn_model/traffictracker.cfg",
    "dnn_model/traffictracker.weights");
195                 if (CUDA){
196                     try{
197                         net.setPreferableBackend(DNN_BACKEND_CUDA);
198                         net.setPreferableTarget(DNN_TARGET_CUDA_FP16);
199                         MODE = "GPU";
200                     } catch (const Exception e){
201                         MODE = "CPU";
202                     }
203                 }
204                 DetectionModel model = DetectionModel(net);
205                 model.setInputParams(1.0/255.0, Size(320, 320), Scalar(), true);
206 
207                 VideoCapture source = VideoCapture(source_file);
208                 source.set(CAP_PROP_FPS, 30);
209                 source.set(CAP_PROP_FRAME_WIDTH, 640.0);
210                 source.set(CAP_PROP_FRAME_HEIGHT, 480.0);
211 
212                 int64 t_start;
213                 int64 t_end;
214 
215 
216                 for (;camera_stopper.at(id);){
217 
218                     t_start = getTickCount();
219 
220                     //Catturo il frame dal video
221                     try{
222                         source.read(frame);
223                         FRAME_COUNT++;
224                     } catch(const Exception e){
225                         destroyAllWindows();
226                         break;
227                     }
228 
229                     // Faccio il riconoscimento
230                     model.detect(frame, classIds, confidences, boxes, 0.6, 0.6);
231 
232                     std::array<int, 3> point_aux;
233 
234                     // Tracker
235                     std::vector<std::array<int, 3» tracker_aux;
236 
237                     for (int i=0; i<classIds.size(); i++){
238 
239                         point_aux[0] = (boxes[i].x*2+boxes[i].width)/2;
240                         point_aux[1] = (boxes[i].y*2+boxes[i].height)/2;
241 
242                         float distance = 20.0;
243 
244                         for (auto traker : {tracker_1, tracker_2, tracker_3}){
245                             for (auto t_point : traker){
246                                 //controllo la distanza tra il punto centrale della box e quelli
    registrati nei frame precedente
247                                 if( sqrt(pow(t_point[0]-point_aux[0],2) + pow(t_point[1] - point_aux[1],
    2) * 1.0) < distance){
248                                     // Se si sono punti che rappresentano lo stesso oggetto
249                                     point_aux[2] = t_point[2];
250                                     goto point_found;
251                                 }
252                             }
253                             // In caso negativo passo al traker del frame precedente aumentando la
    distanza (purtroppo aggiungendo frame si perde precisione)
254                             distance += 20.0;
255                         }
256 
257                         // Se non trovo nulla in nessuno dei frame precedenti vuol dire che ho un nuovo
    oggetto!
258                         CAR_COUT++;
259                         point_aux[2] = CAR_COUT;
```

```
260
261                         // Aggiungo il punto al nuovo traker
262                         point_found:
263                         tracker_aux.push_back(point_aux);
264                     }
265
266                     tracker_3 = tracker_2;
267                     tracker_2 = tracker_1;
268                     tracker_1 = tracker_aux;
269
270                     // Ogni miniuto aggiorno il database (se necessario)
271                     if(FRAME_COUNT%(FPS_TARGET*60) == 0){
272                         if(tracker_1.size() >= traffico_intenso){
273                             if(scrolling != 2){
274                                 scrolling = 2;
275                                 database.insert_registrazione(id, scrolling);
276                             }
277                         }else if (tracker_1.size() >= traffico) {
278                             if(scrolling != 1){
279                                 scrolling = 1;
280                                 database.insert_registrazione(id, scrolling);
281                             }
282                         }else{
283                             if(scrolling != 0){
284                                 scrolling = 0;
285                                 database.insert_registrazione(id, scrolling);
286                             }
287                         }
288                     }
289
290                     t_end = getTickCount();
291
292                     FPS_CURR_FREQ = 1.0/(getTickFrequency()/(t_end-t_start));
293
294                     std::this_thread::sleep_for(std::chrono::milliseconds((FPS_TARGET_FREQ >
    FPS_CURR_FREQ) ? short(1000*(FPS_TARGET_FREQ-FPS_CURR_FREQ)) : 0));
295
296                     FPS = short(getTickFrequency()/(getTickCount() - t_start));
297
298                 }
299
300                 database.stop_telecamera(id);
301                 return;
302
303             }
304
318             Camera(unsigned int id, std::string source_file, int traffico, int traffico_intenso,
    std::string name = "", bool cuda = false, bool debug = false,  int fps = -1,bool is_colored = false){
319                 CUDA = cuda;
320                 DEBUG = debug;
321                 FPS_TARGET = fps;
322                 this->is_colored = is_colored;
323                 this->id = id;
324                 this->name = name;
325                 this->source_file = source_file;
326                 this->traffico = traffico;
327                 this->traffico_intenso = traffico_intenso;
328
329                 status = "stopped";
330             }
331
332             ~Camera(){}
333
334     };
335
336 }
337
338 #endif //CAMERA_THREADS_HPP
```

4.3 Riferimenti per il file Core/includes/database_utilities.hpp

```
#include <cstdlib>
#include <vector>
#include <exception>
#include <iostream>
#include <mysql++.h>
#include <mysql++/connection.h>
#include <mysql++/query.h>
#include <mysql++/result.h>
```

```
#include <ostream>
#include <string>
#include <yaml-cpp/node/node.h>
#include <yaml-cpp/yaml.h>
#include <yaml-cpp/node/parse.h>
```

Composti

- class Database::Database

4.3.1 Descrizione dettagliata

Autore

Andrea Valenzano

Data

2022

Copyright

AGPLv3: Affero General Public License version 3

Precondizione

Libreria per l'interazione con il database

4.4 database_utilities.hpp

Vai alla documentazione di questo file.

```
1
9 /* Copyright (C) 2022  Andrea Valenzano
10  *
11  * This program is free software: you can redistribute it and/or modify
12  * it under the terms of the GNU Affero General Public License as published by
13  * the Free Software Foundation, either version 3 of the License, or
14  * (at your option) any later version.
15  *
16  * This program is distributed in the hope that it will be useful,
17  * but WITHOUT ANY WARRANTY; without even the implied warranty of
18  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  * GNU General Public License for more details.
20  *
21  * You should have received a copy of the GNU General Public License
22  * along with this program.  If not, see <http://www.gnu.org/licenses/>.
23  */
24
25 #ifndef DATABASE_UTILITIES_H
26 #define DATABASE_UTILITIES_H
27
28 #include <cstdlib>
29 #include <vector>
30 #include <exception>
31 #include <iostream>
32 #include <mysql++.h>
33 #include <mysql++/connection.h>
34 #include <mysql++/query.h>
35 #include <mysql++/result.h>
36 #include <ostream>
37 #include <string>
```

```
38 #include <yaml-cpp/node/node.h>
39 #include <yaml-cpp/yaml.h>
40 #include <yaml-cpp/node/parse.h>
41
42
43 namespace Database {
44
49     class Database{
50
51         private:
52
53         std::string database;
54         std::string server;
55         std::string user;
56         std::string passwd;
57         unsigned int port;
58
59         public:
60
66         Database(std::string config_path){
67
68             //Inizializzo i dati leggendoli dal database
69             YAML::Node config_file = YAML::LoadFile(config_path);
70             try{
71                 database = config_file["database"].as<std::string>();
72                 server = config_file["server"].as<std::string>();
73                 user = config_file["user"].as<std::string>();
74                 passwd = config_file["passwd"].as<std::string>();
75                 port = config_file["port"].as<unsigned>();
76             }catch(YAML::Exception e){
77                 std::cout « e.what() « std::endl;
78                 exit(EXIT_FAILURE);
79             }
80
81         }
87         bool check_conneciton(){
88             mysqlpp::Connection connection(false);
89             return connection.connect(database.c_str(), server.c_str(), user.c_str(), passwd.c_str(),
    port);
90         }
91
99         bool insert_registrazione(unsigned int id, unsigned int state){
100             std::string scorrimento = (state == 0)?"scorrevole":(state == 1)?"traffico":(state ==
    2)?"traffico intenso":"Error";
101             if(scorrimento == "Error") return false;
102             mysqlpp::Connection connection(false);
103             if(!connection.connect(database.c_str(), server.c_str(), user.c_str(), passwd.c_str(),
    port)) return false;
104             mysqlpp::Query query = connection.query("INSERT INTO registrazione VALUES(CURRENT_TIMESTAMP,
    \"" + scorrimento + "\", " + std::to_string(id) + ")");
105             mysqlpp::StoreQueryResult result = query.store();
106             if(query.affected_rows() == 0) return false;
107             return true;
108         }
109
117         bool insert_telecamere(float lat, float lng){
118             if(lat > 90 || lat < 0 || lng > 90 || lng < 0) return false;
119             mysqlpp::Connection connection(false);
120             if(!connection.connect(database.c_str(), server.c_str(), user.c_str(), passwd.c_str(),
    port)) return false;
121             mysqlpp::Query query = connection.query("INSERT INTO telecamere VALUES(NULL, " +
    std::to_string(lat) + ", " + std::to_string(lng) + ", \"OFF\")");
122             mysqlpp::StoreQueryResult result = query.store();
123             if(query.affected_rows() == 0) return false;
124             return true;
125         }
126
135         bool insert_telecamere_by_id(int id, float lat, float lng){
136             if(id < 0 || lat > 90 || lat < 0 || lng > 90 || lng < 0) return false;
137             mysqlpp::Connection connection(false);
138             if(!connection.connect(database.c_str(), server.c_str(), user.c_str(), passwd.c_str(),
    port)) return false;
139             mysqlpp::Query query = connection.query("INSERT INTO telecamere VALUES(" +
    std::to_string(id) + ", " + std::to_string(lat) + ", " + std::to_string(lng) + ", \"OFF\")");
140             mysqlpp::StoreQueryResult result = query.store();
141             if(query.affected_rows() == 0) return false;
142             return true;
143         }
144
151         bool start_telecamera(int id){
152             mysqlpp::Connection connection(false);
153             if(!connection.connect(database.c_str(), server.c_str(), user.c_str(), passwd.c_str(),
    port)) return false;
154             mysqlpp::Query query = connection.query("UPDATE telecamere SET status = 'ON' WHERE id = " +
    std::to_string(id));
155             mysqlpp::StoreQueryResult result = query.store();
156             if(query.affected_rows() == 0) return false;
```

```
157             return true;
158         }
159 
166         bool stop_telecamera(int id){
167             mysqlpp::Connection connection(false);
168             if(!connection.connect(database.c_str(), server.c_str(), user.c_str(), passwd.c_str(),
    port)) return false;
169             mysqlpp::Query query = connection.query("UPDATE telecamere SET status = 'OFF' WHERE id = " +
    std::to_string(id));
170             mysqlpp::StoreQueryResult result = query.store();
171             if(query.affected_rows() == 0) return false;
172             return true;
173         }
174 
180         int get_cameras_count(){
181             mysqlpp::Connection connection(false);
182             if(!connection.connect(database.c_str(), server.c_str(), user.c_str(), passwd.c_str(),
    port)) return -1;
183             mysqlpp::Query query = connection.query("SELECT * FROM telecamere");
184             mysqlpp::StoreQueryResult result = query.store();
185             return query.affected_rows();
186         }
192         std::vector<int> get_cameras_ids(){
193             std::vector<int> buffer;
194             mysqlpp::Connection connection(false);
195             if(!connection.connect(database.c_str(), server.c_str(), user.c_str(), passwd.c_str(),
    port)) return buffer;
196             mysqlpp::Query query = connection.query("SELECT * FROM telecamere");
197             mysqlpp::StoreQueryResult result = query.store();
198             for(auto id : result){
199                 buffer.push_back((int)id["id"]);
200             }
201             return buffer;
202         }
203      };
204 
205 
206 }
207 
208 #endif
```

4.5 Riferimenti per il file Core/includes/hub_utilities.hpp

```
#include <iostream>
#include <stdio.h>
#include <readline/readline.h>
#include <readline/history.h>
#include <string>
```

Composti

- struct utilities::colors
- class utilities::Prompt

  Classe contenitore per tutti i metodi usati nell'interazione grafica con l'utente.

Funzioni

- bool utilities::is_number (const std::string &s)

  Funzione che verifica se una stringa e' numerica.

4.5.1 Descrizione dettagliata

Autore

Andrea Valenzano

Data

2022

Copyright

AGPLv3: Affero General Public License version 3

Precondizione

Libreria che implementa le Utilities per la gestione dell'interfaccia utente

4.5.2 Documentazione delle funzioni

4.5.2.1 is_number() `bool utilities::is_number ( const std::string & s ) [inline]`

Funzione che verifica se una stringa e' numerica.

Parametri

in	*s*	std::string da verificare

Restituisce

Vero se la stringa e' numerica, Falso altrimenti

4.6 hub_utilities.hpp

Vai alla documentazione di questo file.

```
1
9 /* Programma scritto durante lo svolgimento del progetto finale per il
10  * corso di Multimedia App and Internet of Things dell'Universita' degl Studi di Genova
11  *
12  * Copyright (C) 2022  Andrea Valenzano
13  *
14  * This program is free software: you can redistribute it and/or modify
15  * it under the terms of the GNU Affero General Public License as published by
16  * the Free Software Foundation, either version 3 of the License, or
17  * (at your option) any later version.
18  *
19  * This program is distributed in the hope that it will be useful,
20  * but WITHOUT ANY WARRANTY; without even the implied warranty of
21  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
22  * GNU General Public License for more details.
23  *
24  * You should have received a copy of the GNU General Public License
```

```
25  * along with this program.  If not, see <http://www.gnu.org/licenses/>.
26  */
27
28 #ifndef HUB_UTILITIES_HPP
29 #define HUB_UTILITIES_HPP
30
31 #include <iostream>
32 #include <stdio.h>
33 #include <readline/readline.h>
34 #include <readline/history.h>
35 #include <string>
36
37
38 namespace utilities {
39
44     struct colors{
45         std::string red = "\033[31m";
46         std::string blue = "\033[34m";
47         std::string green = "\033[32m";
48         std::string magenta = "\033[35m";
49         std::string cyan = "\033[36m";
50         std::string yellow = "\033[33m";
51         std::string defaul = "\033[0m";
52     };
53
60     inline bool is_number(const std::string& s)
61     {
62         std::string::const_iterator it = s.begin();
63         while (it != s.end() && std::isdigit(*it)) ++it;
64         return !s.empty() && it == s.end();
65     }
66
71     class Prompt{
72
73         private:
74
75             bool is_colored;
76             utilities::colors color;
77
78         public:
79
86             std::string get_prompt(){
87                 std::string prompt;
88                 if(is_colored){
89                     prompt = color.green + "" + color.cyan + " Server" + color.defaul + "Iot" +
    color.green + "++" + color.defaul + " " + color.cyan + "" + color.defaul + "" + color.green + " " +
    color.defaul;
90                 }else
91                     prompt  =  "[ServerIot++] >> ";
92
93                 std::string input(readline(prompt.c_str()));
94                 return input;
95             }
96
101             void prompt_help(){
102                 std::cout « std::endl « "ServerIot 'help':" « std::endl;
103                 std::cout « std::endl « "     - camera " « std::endl;
104                 std::cout « "          Command to handle cameras" « std::endl;
105                 std::cout « std::endl « "     - system " « std::endl;
106                 std::cout « "          Execute a command on default shell" « std::endl;
107                 std::cout « std::endl « "     - help " « std::endl;
108                 std::cout « "          Show commands list" « std::endl;
109                 std::cout « std::endl « "     - uptime " « std::endl;
110                 std::cout « "          Show server uptime" « std::endl;
111                 std::cout « std::endl « "     - version " « std::endl;
112                 std::cout « "          Show program version" « std::endl;
113                 std::cout « std::endl « "     - exit " « std::endl;
114                 std::cout « "          Quit program" « std::endl;
115                 std::cout « std::endl;
116             }
117
123             void command_error(std::string command){
124                 std::cout « std::endl « ((is_colored==true)?color.red:"") + "Error!" + color.defaul + "
    command \"" « command « "\" not found!" « std::endl;
125                 std::cout « "Type \"" + ((is_colored==true)?color.magenta:"") + "help" + color.defaul +
    "\" for hints" « std::endl « std::endl;
126             }
127
133             void camera_error(std::string command){
134                 std::cout « std::endl « ((is_colored==true)?color.red:"") + "Error!" + color.defaul + "
    option \"" « command « "\" for command \"camera\" not found!" « std::endl;
135                 std::cout « "Type \"" + ((is_colored==true)?color.magenta:"") + "camera help" +
    color.defaul + "\" for hints" « std::endl « std::endl;
136             }
137
143             void camera_status_error(std::string command){
144                 std::cout « std::endl « ((is_colored==true)?color.red:"") + "Error!" + color.defaul + "
```

```
    option \"" « command « "\" for command \"camera status\" not found!" « std::endl;
145                 std::cout « "Type \"" + ((is_colored==true)?color.magenta:"") + "camera status help" +
    color.defaul + "\" for hints" « std::endl « std::endl;
146             }
147
153             void camera_start_error(std::string command){
154                 std::cout « std::endl « ((is_colored==true)?color.red:"") + "Error!" + color.defaul + "
    option \"" « command « "\" for command \"camera start\" not found!" « std::endl;
155                 std::cout « "Type \"" + ((is_colored==true)?color.magenta:"") + "camera start help" +
    color.defaul + "\" for hints" « std::endl « std::endl;
156             }
157
163             void camera_stop_error(std::string command){
164                 std::cout « std::endl « ((is_colored==true)?color.red:"") + "Error!" + color.defaul + "
    option \"" « command « "\" for command \"camera stop\" not found!" « std::endl;
165                 std::cout « "Type \"" + ((is_colored==true)?color.magenta:"") + "camera stop help" +
    color.defaul + "\" for hints" « std::endl « std::endl;
166             }
167
173             void camera_help(unsigned int help_code){
174                 switch (help_code) {
175                     case 1:
176                         std::cout « std::endl « "camera status 'help':" « std::endl;
177                         std::cout « std::endl « "     - camera status {'camera number' / all}" «
    std::endl;
178                         std::cout « "          Show Cameras status and uptime" « std::endl;
179                         std::cout « std::endl;
180                     break;
181
182                     case 2:
183                         std::cout « std::endl « "camera start 'help':" « std::endl;
184                         std::cout « std::endl « "     - camera start {'camera number/numbers'}" «
    std::endl;
185                         std::cout « "          Start target camera(s)" « std::endl;
186                         std::cout « std::endl;
187                     break;
188
189                     case 3:
190                         std::cout « std::endl « "camera stop 'help':" « std::endl;
191                         std::cout « std::endl « "     - camera stop {'camera number/numbers'}" «
    std::endl;
192                         std::cout « "          Stope target camera(s)" « std::endl;
193                         std::cout « std::endl;
194                     break;
195
196                     case 0:
197                     default:
198                         std::cout « std::endl « "camera 'help':" « std::endl;
199                         std::cout « std::endl « "     - status {'camera number' / all}" « std::endl;
200                         std::cout « "          Show cameras status and uptime" « std::endl;
201                         std::cout « std::endl « "     - start {'camera number/numbers'}" « std::endl;
202                         std::cout « "          Start target camera(s)" « std::endl;
203                         std::cout « std::endl « "     - stop {'camera number/numbers'}" « std::endl;
204                         std::cout « "          Stop target camera(s)" « std::endl;
205                         std::cout « std::endl;
206                 }
207             }
208
215             Prompt(bool colors = false){
216                 is_colored = colors;
217             }
218
219             ~Prompt(){}
220
221     };
222
223 }
224
225 #endif //HUB_UTILITIES_HPP
```

4.7 Riferimenti per il file Core/includes/string_parser.hpp

```
#include <iostream>
#include <queue>
```

Composti

- class stringParser::Parser

  Classe che implementa il manipolatore di stringhe.

4.7.1 Descrizione dettagliata

Autore

Andrea Valenzano

Data

2022

Copyright

AGPLv3: Affero General Public License version 3

Precondizione

Libreria per la gestione delle stringhe in input al programma

4.8 string_parser.hpp

Vai alla documentazione di questo file.

```
1
9 /* Programma scritto durante lo svolgimento del progetto finale per il
10  * corso di Multimedia App and Internet of Things dell'Universita' degl Studi di Genova
11  *
12  * Copyright (C) 2022  Andrea Valenzano
13  *
14  * This program is free software: you can redistribute it and/or modify
15  * it under the terms of the GNU Affero General Public License as published by
16  * the Free Software Foundation, either version 3 of the License, or
17  * (at your option) any later version.
18  *
19  * This program is distributed in the hope that it will be useful,
20  * but WITHOUT ANY WARRANTY; without even the implied warranty of
21  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
22  * GNU General Public License for more details.
23  *
24  * You should have received a copy of the GNU General Public License
25  * along with this program.  If not, see <http://www.gnu.org/licenses/>.
26  */
27
28 #ifndef STRING_PARSER_HPP
29 #define STRING_PARSER_HPP
30
31 #include <iostream>
32 #include <queue>
33
34 namespace stringParser{
35
40     class Parser{
41
42         private:
43
44             std::queue<std::string> buffer;
46         public:
47
56             int parser_init(std::string input){
57
58                 std::string token;
59                 size_t pos = 0;
60                 std::string delimiter = " ";
61
62                 try{
63
64                     while(!buffer.empty()){
65                         buffer.pop();
66                     }
67
68                     while ((pos = input.find(delimiter)) != std::string::npos) {
69                         token = input.substr(0, pos);
70                         input.erase(0, pos + delimiter.length());
```

```
71                          if(token != ""){
72                              buffer.push(token);
73                          }
74                      }
75                      if(input != ""){
76                              buffer.push(input);
77                          }
78
79                  } catch( std::exception e){
80                      e.what();
81                      return -1;
82                  }
83
84                  return (int)buffer.size();
85              }
86
94              int print_buffer(){
95
96                  std::string aux;
97
98                  try{
99                      for(unsigned int i=0; i<buffer.size(); i++){
100
101                          aux = buffer.front();
102                          std::cout « aux « std::endl;
103                          buffer.push(aux);
104                          buffer.pop();
105
106                      }
107                  } catch (std::exception e){
108                      e.what();
109                      return -1;
110                  }
111                  return 0;
112              }
113
120              std::queue<std::string> get_buffer(){
121                  return buffer;
122              }
123
129              Parser(){}
130
137              ~Parser(){
138                  while(!buffer.empty()){
139                      buffer.pop();
140                  }
141              }
142      };
143 }
144
145 #endif //STRING_PARSER_HPP
```

4.9 Riferimenti per il file Core/source/main.cpp

```
#include "../includes/string_parser.hpp"
#include "../includes/hub_utilities.hpp"
#include "../includes/camera_threads.hpp"
#include "../includes/database_utilities.hpp"
#include <cctype>
#include <cstdlib>
#include <iostream>
#include <readline/history.h>
#include <vector>
#include <yaml-cpp/node/parse.h>
#include <yaml-cpp/yaml.h>
#include <time.h>
#include <stdlib.h>
```

Definizioni

- #define **VERSION** "0.1.0 alpha"

Funzioni

- int **main** (int argc, char $**$argv)

  Funzione main Funzione che inplementa il main loop del programma e l'interfaccia per l'iserimento dei comandi da parte del'utente.

Variabili

- std::vector< camera::Camera > **cameras**

4.9.1 Descrizione dettagliata

Autore

Andrea Valenzano

Data

2022

Copyright

AGPLv3: Affero General Public License version 3

Precondizione

Server per la gestione delle telecamere, Interfaccia per la gestione e l'analisi dell'input delle telecamenre di Traffic Traker.