#ifndef IMAGE\_OPENCV\_H

#define IMAGE\_OPENCV\_H

#include "image.h"

#include "matrix.h"

#ifdef \_\_cplusplus

extern "C" {

#endif

#ifdef OPENCV

// declaration

typedef void\* mat\_cv;

typedef void\* cap\_cv;

typedef void\* write\_cv;

//typedef struct mat\_cv mat\_cv;

//typedef struct cap\_cv cap\_cv;

//typedef struct write\_cv write\_cv;

// cv::Mat

mat\_cv \*load\_image\_mat\_cv(const char \*filename, int flag);

image load\_image\_cv(char \*filename, int channels);

image load\_image\_resize(char \*filename, int w, int h, int c, image \*im);

int get\_width\_mat(mat\_cv \*mat);

int get\_height\_mat(mat\_cv \*mat);

void release\_mat(mat\_cv \*\*mat);

// IplImage - to delete

//int get\_width\_cv(mat\_cv \*ipl);

//int get\_height\_cv(mat\_cv \*ipl);

//void release\_ipl(mat\_cv \*\*ipl);

// image-to-ipl, ipl-to-image, image\_to\_mat, mat\_to\_image

//mat\_cv \*image\_to\_ipl(image im); // to delete

//image ipl\_to\_image(mat\_cv\* src\_ptr); // to delete

// mat\_cv \*image\_to\_ipl(image im)

// image ipl\_to\_image(mat\_cv\* src\_ptr)

// cv::Mat ipl\_to\_mat(IplImage \*ipl)

// IplImage \*mat\_to\_ipl(cv::Mat mat)

// Mat image\_to\_mat(image img)

// image mat\_to\_image(cv::Mat mat)

image mat\_to\_image\_cv(mat\_cv \*mat);

// Window

void create\_window\_cv(char const\* window\_name, int full\_screen, int width, int height);

void destroy\_all\_windows\_cv();

int wait\_key\_cv(int delay);

int wait\_until\_press\_key\_cv();

void make\_window(char \*name, int w, int h, int fullscreen);

void show\_image\_cv(image p, const char \*name);

//void show\_image\_cv\_ipl(mat\_cv \*disp, const char \*name);

void show\_image\_mat(mat\_cv \*mat\_ptr, const char \*name);

// Video Writer

write\_cv \*create\_video\_writer(char \*out\_filename, char c1, char c2, char c3, char c4, int fps, int width, int height, int is\_color);

void write\_frame\_cv(write\_cv \*output\_video\_writer, mat\_cv \*mat);

void release\_video\_writer(write\_cv \*\*output\_video\_writer);

//void \*open\_video\_stream(const char \*f, int c, int w, int h, int fps);

//image get\_image\_from\_stream(void \*p);

//image load\_image\_cv(char \*filename, int channels);

//int show\_image\_cv(image im, const char\* name, int ms);

// Video Capture

cap\_cv\* get\_capture\_video\_stream(const char \*path);

cap\_cv\* get\_capture\_webcam(int index);

void release\_capture(cap\_cv\* cap);

mat\_cv\* get\_capture\_frame\_cv(cap\_cv \*cap);

int get\_stream\_fps\_cpp\_cv(cap\_cv \*cap);

double get\_capture\_property\_cv(cap\_cv \*cap, int property\_id);

double get\_capture\_frame\_count\_cv(cap\_cv \*cap);

int set\_capture\_property\_cv(cap\_cv \*cap, int property\_id, double value);

int set\_capture\_position\_frame\_cv(cap\_cv \*cap, int index);

// ... Video Capture

image get\_image\_from\_stream\_cpp(cap\_cv \*cap);

image get\_image\_from\_stream\_resize(cap\_cv \*cap, int w, int h, int c, mat\_cv\*\* in\_img, int dont\_close);

image get\_image\_from\_stream\_letterbox(cap\_cv \*cap, int w, int h, int c, mat\_cv\*\* in\_img, int dont\_close);

// Image Saving

void save\_cv\_png(mat\_cv \*img, const char \*name);

void save\_cv\_jpg(mat\_cv \*img, const char \*name);

// Draw Detection

void draw\_detections\_cv\_v3(mat\_cv\* show\_img, detection \*dets, int num, float thresh, char \*\*names, image \*\*alphabet, int classes, int ext\_output);

// Draw Loss & Accuracy chart

mat\_cv\* draw\_train\_chart(char \*windows\_name, float max\_img\_loss, int max\_batches, int number\_of\_lines, int img\_size, int dont\_show, char\* chart\_path);

void draw\_train\_loss(char \*windows\_name, mat\_cv\* img, int img\_size, float avg\_loss, float max\_img\_loss, int current\_batch, int max\_batches,

float precision, int draw\_precision, char \*accuracy\_name, int dont\_show, int mjpeg\_port, double time\_remaining);

// Data augmentation

image image\_data\_augmentation(mat\_cv\* mat, int w, int h,

int pleft, int ptop, int swidth, int sheight, int flip,

float dhue, float dsat, float dexp,

int gaussian\_noise, int blur, int num\_boxes, float \*truth);

// blend two images with (alpha and beta)

void blend\_images\_cv(image new\_img, float alpha, image old\_img, float beta);

// bilateralFilter bluring

image blur\_image(image src\_img, int ksize);

// draw objects for Adversarial attacks

void cv\_draw\_object(image sized, float \*truth\_cpu, int max\_boxes, int num\_truth, int \*it\_num\_set, float \*lr\_set, int \*boxonly, int classes, char \*\*names);

// Show Anchors

void show\_acnhors(int number\_of\_boxes, int num\_of\_clusters, float \*rel\_width\_height\_array, model anchors\_data, int width, int height);

void show\_opencv\_info();

#else // OPENCV

void show\_opencv\_info();

int wait\_key\_cv(int delay);

int wait\_until\_press\_key\_cv();

void destroy\_all\_windows\_cv();

#endif // OPENCV

#ifdef \_\_cplusplus

}

#endif

#endif // IMAGE\_OPENCV\_H