结构：

chip\_details:

drectangle rect; // drectangle使用double而非long来存储rect

double angle;

unsigned long rows;

unsigned long cols;

函数：

inline chip\_details get\_face\_chip\_details (

const full\_object\_detection& det,

const unsigned long size = 200,

const double padding = 0.2)

功能：extract a copy of the face that has been rotated upright, centered, and scaled to a standard size

void extract\_image\_chips (const image\_type1& img,

const std::vector<chip\_details>& chip\_locations,

dlib::array<image\_type2>& chips,

const interpolation\_type& interp)

cuda/tensor.h

class alias\_tensor

{

private:

mutable alias\_tensor\_instance inst;

}

指向内存中现有的tensor，而不拥有自己的memory。

class tensor

{

protected:

long long m\_n;

long long m\_k;

long long m\_nr;

long long m\_nc;

long long m\_size; // always equal to m\_n\*m\_k\*m\_nr\*m\_nc

}

四维的tensor，（n,k,nr,nc）

image\_pyramid.h

class pyramid\_down{

void operator() (const in\_image\_type& original,

out\_image\_type& down) const;

down是original的（（N-1）/N）;

template <typename T>

vector<double,2> point\_down (const vector<T,2>& p) const;

p为原图中的点，返回downsample中对应的点p，与point\_up函数对应

template <typename T>

vector<double,2> point\_down (

const vector<T,2>& p,

unsigned int levels) const;

返回levels次point\_down的结果，即:

point\_down(p,2)=point\_down(point\_down(p));

template <typename T>

vector<double,2> point\_up (const vector<T,2>& p) const；

drectangle rect\_down (const drectangle& rect) const;

返回drectangle(point\_down(rect.tl\_corner()), point\_down(rect.br\_corner()))的结果

其他函数以此类推。

}

功能：It downsamples images at a ratio of N to N-1.

class pyramid\_down<2>等价于pyramid\_down\_2\_1

class pyramid\_down\_2\_1

{

private:

template <typename T, typename U>

struct both\_images\_rgb

{ typedef typename image\_traits<T>::pixel\_type T\_pix;

typedef typename image\_traits<U>::pixel\_type U\_pix;

const static bool value = pixel\_traits<T\_pix>::rgb && pixel\_traits<U\_pix>::rgb;};

struct rgbptype

{ uint16 red;

uint16 green;

uint16 blue;};

}

point\_transform.h

class point\_rotator

{

private:

double sin\_angle;

double cos\_angle;

const dlib::vector<T,2> operator() (

const dlib::vector<T,2>& p) const

{x=cos\*p.x-sin\*p.y;

y=sin\*p.x+cos\*p.y;}

}