关于warpaffine的cuda实现，一开始测试，我以为warpaffine的实现resize bilinear lettor box没有对齐

关于warpaffine和resize总共两个问题

1.padding问题，查询了资料，该cuda的实现的变换矩阵采用了padding，因为使用了缩放和平移，

并且带越界处理问题

变换矩阵参考这个解释

就是使用带偏移和scale的矩阵，由于使用的是双线性插值https://zhuanlan.zhihu.com/p/89684929 [2]

需要的是dst img 到原图的srcimg的value映射关系，所以需要有dst 坐标

640 480 0 0--> 0 0 配合boarder

opencv resize linear

```
for (int dx = 0; dx < dsize.width; dx++)
            {
                fxx = (float)((dx+0.5)*inv_fx - 0.5);
                //用到了0.5 1 0 scale 1 1 情况下 1 0
                sx = cvFloor(fxx);
      //         int i = (int)value;
    //return i - (i > value);
                fxx -= sx;

                if (sx < 0)
                    fxx = 0, sx = 0;

                if (sx >= ssize.width-1)
                    fxx = 0, sx = ssize.width-1;

                xofs[dx] = sx;
                ialpha[dx*2 + 0] = saturate_cast<short>((1.f - fxx) *
INTER_RESIZE_COEF_SCALE);
                ialpha[dx*2 + 1] = saturate_cast<short>(fxx         *
INTER_RESIZE_COEF_SCALE);
            }

            for (int dy = 0; dy < dsize.height; dy++)
            {
                fyy = (float)((dy+0.5)*inv_fy - 0.5);
                sy = cvFloor(fyy);
                fyy -= sy;

                yofs[dy] = sy;
                ibeta[dy*2 + 0] = saturate_cast<short>((1.f - fyy) *
INTER_RESIZE_COEF_SCALE);
                ibeta[dy*2 + 1] = saturate_cast<short>(fyy         *
INTER_RESIZE_COEF_SCALE);
            }
```

warpaffine在源码中变换

warpaffine matrix

```
s2d.value[0] = scale;
s2d.value[1] = 0;
s2d.value[2] = -scale * src_width * 0.5 + dst_width * 0.5;
s2d.value[3] = 0;
s2d.value[4] = scale;
s2d.value[5] = -scale * src_height * 0.5 + dst_height * 0.5;
////做了逆变换到逆矩阵
 cv::invertAffineTransform(m2x3_s2d, m2x3_d2s);
```


$$scale = min(\frac{Dst.width}{Origin.width}, \frac{Dst.height}{Origin.height})$$

$$M = \begin{Bmatrix} scale & 0 & -\frac{scale \times Origin.width}{2} + \frac{Dst.width}{2} \\ 0 & scale & -\frac{scale \times Origin.height}{2} + \frac{Dst.height}{2} \end{Bmatrix}$$

#

$$\begin{Bmatrix} x' \\ y' \end{Bmatrix} = \begin{Bmatrix} scale & 0 & -\frac{scale \times Origin.width}{2} + \frac{Dst.wi}{2} \\ 0 & scale & -\frac{scale \times Origin.height}{2} + \frac{Dst.he}{2} \end{Bmatrix}$$

由于变换比较简单，那么逆变换则可以很容易的写出来

#

$$k = scale$$

$$b1 = -\frac{scale \times Origin.width}{2} + \frac{Dst.width}{2}$$

$$b2 = -\frac{scale \times Origin.height}{2} + \frac{Dst.height}{2}$$

$$x' = kx + b1$$

$$y' = ky + b2$$

$$x = \frac{x'-b1}{k} = x' \times \frac{1}{k} + (-\frac{b1}{k})$$

$$y = \frac{y'-b2}{k} = y' \times \frac{1}{k} + (-\frac{b2}{k})$$

$$M^{-1} = \begin{Bmatrix} \frac{1}{k} & 0 & -\frac{b1}{k} \\ 0 & \frac{1}{k} & -\frac{b2}{k} \end{Bmatrix}$$

```
int position = blockDim.x * blockIdx.x + threadIdx.x;
//这里获得了运算grid中块的idx坐标
int dx = position % dst_width;
int dy = position / dst_width;
//获得dst 目标计算图中坐标dx dy
```

测试640 480 则 0 0 -->0 0

```
//0.5f已经使用offset
//m_x1:scale_x m_x2:scale_y 缩放比例
//m_z1:偏移
//0.5f:是为计算中心 resize使用的是0.5f 而传统的warpaffine用的0.0f所以有误差
//参考[2]
//这里有个0.5的偏移，主要是因为warpaffine做resize和中心和opencvresize差距
//https://zhuanlan.zhihu.com/p/99626808 该文章
//https://zhuanlan.zhihu.com/p/89684929 该文章
// 0 0 scale 1的情况下 0.5f
float src_x = m_x1 * dx + m_y1 * dy + m_z1 + 0.5f;
float src_y = m_x2 * dx + m_y2 * dy + m_z2 + 0.5f;
```

cuda 实现中采用边界处理的方式

```
if (src_x <= -1 || src_x >= src_width || src_y <= -1 || src_y >= src_height)
{
// out of range
c0 = const_value_st;
c1 = const_value_st;
c2 = const_value_st;
}
//利用逆变换矩阵,对于dx dy 0 0点会映射到边界外直接赋予 const_value_st 114 同理
```

获得四个角点坐标执行公式即参考

```
int y_low = floorf(src_y);
int x_low = floorf(src_x);
int y_high = y_low + 1;
int x_high = x_low + 1;
//获得坐标
v1 = src + y_low * src_line_size + x_low * 3;
v2 = src + y_low * src_line_size + x_high * 3;
v3 = src + y_high * src_line_size + x_low * 3;
v4 = src + y_high * src_line_size + x_high * 3;
```

$$
\begin{aligned}
f(x,y) &\approx \frac{y_2-y}{y_2-y_1}f(x,y_1)+\frac{y-y_1}{y_2-y_1}f(x,y_2) \\
&\approx \frac{x_2-x}{x_2-x_1}\frac{y_2-y}{y_2-y_1}f(x_1,y_1)+\frac{x-x_1}{x_2-x_1}\frac{y_2-y}{y_2-y_1}f(x_2,y_1) \\
&+ \frac{x_2-x}{x_2-x_1}\frac{y-y_1}{y_2-y_1}f(x_1,y_2)+\frac{x-x_1}{x_2-x_1}\frac{y-y_1}{y_2-y_1}f(x_2,y_2)
\end{aligned}
$$

```
float ly = src_y - y_low;
float lx = src_x - x_low;

float hy = 1 - ly;
float hx = 1 - lx;

float w1 = hy * hx, w2 = hy * lx, w3 = ly * hx, w4 = ly * lx;
计算了点的value
c0 = w1 * v1[0] + w2 * v2[0] + w3 * v3[0] + w4 * v4[0];
```

w1 = hy * hx, w2 = hy * lx, w3 = ly * hx, w4 = ly * lx;

其中w1为 v1[0]该点的权重值，v1为像素值

后续部分是rgb通道互换还有NHWC转NCHW

```
//完成通道转换
float t = c2;
c2 = c0;
c0 = t;

//normalization
c0 = c0 / 255.0f;
c1 = c1 / 255.0f;
c2 = c2 / 255.0f;

//rgbrgbrgb to rrrgggbbb
int area = dst_width * dst_height;
float *pdst_c0 = dst + dy * dst_width + dx;
float *pdst_c1 = pdst_c0 + area;
float *pdst_c2 = pdst_c1 + area;
//赋值
*pdst_c0 = c0;
*pdst_c1 = c1;
*pdst_c2 = c2;
```

2.bilinear的实现形式，的确在warpaffine的实现中使用的是opencv bilinear但是在使用过程中会出现边界问题，参考如下文章

https://zhuanlan.zhihu.com/p/89684929

https://zhuanlan.zhihu.com/p/99626808