UI 提供两个叠加的SurfaceView（1440×1080）

一个用于显示Camera图像，一个用于绘制物体轮廓

例如：

maincamera_surface_view： 用于显示Camera图像

outline_surface_view ： 用于绘制物体轮廓

```
<FrameLayout
             android:layout_width="1080px"
             android:layout_height="1440px">
    <SurfaceView
                 android:id="@+id/maincamera_surface_view"
                 android:layout_width="match_parent"
                 android:layout_height="match_parent"/>
    <SurfaceView
                 android:id="@+id/outline_surface_view"
                 android:layout_width="match_parent"
                 android:layout_height="match_parent"/>
</FrameLayout>
```

onCreate()的时候调用

```
/** * 构造tofcubing * @param activity */
private TofCubing tofCubing;
private MeasureState measureState;
private MeasurePostProces measurePostProces;
private void createTofCubingSDK(Activity activity){
    if(tofCubing == null){
        tofCubing = new TofCubing(activity,this, getMainSurfaceView());
    }
    tofCubing.softPauseCamera(TofCubing.CAMERA_PAUSE_NOTHING);
    measureState = MeasureState.getMesaureStateInstance(tofCubing);
}
```

onResume()

```
tofCubing.onResuemTofCubing();
outlineSurfaceView.setZOrderMediaOverlay(true);
outlineSurfaceView.getHolder().setFormat(PixelFormat.TRANSLUCENT);
measureState.setMode(MEASUREMODE.Plane);
```

onPause()

```
tofCubing.onPauseTofCubing();
```

onDestroy()

```
tofCubing.onDestoryTofCubing();
measureState = null;
tofCubing = null;
```

数据处理：

```
     public interface IProcessCallback {
    /**
     * @param
     * rgb RGB相机图像
     * bounding 包裹bounding box坐标点
     * measureMode 当前测量模式
     * length 包裹长
     * width 包裹宽
     * height 包裹高
     * dist 深度相机到测量包裹单面拟合平面的距离
     * angle 深度相机与测量包裹单面拟合平面法线的夹角
     * tofTemp  深度相机运行时的温度
     */
    void processCallback(byte[] rgb, final float[] bounding,
                         int measureMode, int length, int width, int height,
float dist, float angle, float tofTemp);

    /**
     * 后摄数据回调
     * @param rgb RGB相机图像
     */
    void onFaceBackCameraData(byte[] rgb);

    /**
     * -1, 禁止的测量角度
     *
     * @param errorCode
     * @param msg
     */
    void onError(int errorCode, String msg);
    /**
     * 当测量结果稳定后返回
     * @param length 包裹长
     * @param width  包裹宽
     * @param high   包裹高
     */
    void onAverageResultCallback(int length, int width, int high);

}
```

```
@Override
public void processCallback(byte[] rgbsync, final float[] bounding,
                int measureMode, int length, int width, int height, float dist,
float angle, float tofTemp)
{
    updatePointData(bounding);
    updateOutLine();
}
```

```
/**
* 更新物体坐标到cubePoints
* @param bounding
*/
private void updatePointData(float[] bounding){
    int j = 0;
    for(int i = 0; i< cubePoints.length; i++){
        if(cubePoints[i] == null){
            cubePoints[i] = new Point(0,0);
        }
        cubePoints[i].x = (int) bounding[j++];
        cubePoints[i].y = (int) bounding[j++];
    }
}
```

```
/** * 重新绘制物体坐标 */
private void updateOutLine(){
    if (tofCubing.getCameraMode() == TofCubing.CAMERA_PAUSE_NOTHING) {
        //2. 绘制物体坐标
        measurePostProces.drawOutLine(getOutLineSurfaceView().getHolder());
    }
}
```

```
 public void drawOutLine(SurfaceHolder holder){
     Canvas canvas = holder.lockCanvas();
     if(canvas == null) return;
     canvas.rotate(90);
     canvas.translate(0,-CameraApi.PREVIEW_HEIGHT);
     canvas.drawColor(Color.TRANSPARENT, PorterDuff.Mode.CLEAR);
     CanvasUtils.drawOutLine(canvas,cubePoints,14,18,false);
     holder.unlockCanvasAndPost(canvas);
     isMeasureOkView = false;
 }
```

所需权限:

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-feature
android:name="android.hardware.camera.autofocus"
android:required="false" />
```