关于横幅点击之后如何获取到图片的解决方法

这个问题我认为可以分成两个部分：

- 点击事件的获取问题，当用户点击的时候，知道它点击的是哪一个横幅？
- 点击之后图片获取问题，怎么获取，获取到怎么传到后台来？

下面我就对这两个问题说一下我的想法。

点击事件的获取问题

方案1（不推荐）

让用户实现的点击事件，长按事件回调接口继承我们的，这样我们就可以在上层知道何时点击了这个view

优点： 用户可以灵活控制那些点击时间被统计埋点。
缺点： 用户使用起来比较麻烦，需要改动的代码比较多，对上层的逻辑不信任，怕影响效果，侵入性高

方案2（推荐）

使用`application.registerActivityLifecycleCallbacks`方法监听整个app的activity生命周期。

```
application.registerActivityLifecycleCallbacks(new Application.ActivityLifecycleCallbacks() {
  ...
    @Override
    public void onActivityResumed(Activity activity) {
        activity.getWindow().getDecorView().getRootView()
    }
    ...
});
```

然后在onActivityResumed()方法中，我们拿到它的最外层布局，这时候我们给它的子view用代码在外面包上一个我们自定义的布局，这个自定义布局很简单，我们重写它的onTouchEvent()，当用户触摸时，我们可以找到最终这个事件被谁消费了，进而就知道点击的是谁了，代码大概这样：

```
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        switch (event.getAction()){
            case MotionEvent.ACTION_UP:
                float x = event.getRawX();
                float y = event.getRawY();
                //1.通过x，y坐标找到点击了那个view
                View view=getTouchTarget(rootview,x,y);

                //这里咱们SDK分发处理这个事件，可以执行从View抓图，坐标，大小，上传，缓存等等操作
                sdkDispatchEvent(view);
                break;
        }
        //原来的事件分发逻辑该怎么走还怎么走。
        return super.onTouchEvent(event);
    }
//通过x，y坐标找到点击了那个view
//根据坐标返回触摸到的View
private View getTouchTarget(View rootView, int x, int y) {
  View targetView = null;
  ArrayList<View> touchableViews = rootView.getTouchables();
```

```
    for (View touchableView : touchableViews){
      if (isTouchPointInView(touchableView, x, y)){
        targetView = touchableView;
        break;
      }
    }
    return targetView;
  }

  //(x,y)是否在view的区域内
  private boolean isTouchPointInView(View view, int x, int y) {
    if (view == null){
      return false;
    }
    int[] position = new int[2];
    view.getLocationOnScreen(position);
    int left = position[0];
    int top = position[1];
    int right = left + view.getMeasuredWidth();
    int bottom = top + view.getMeasuredHeight();
    if (x >= left && x <= right && y >= top && y <= bottom){
      return true;
    }
    return false;
  }
```

抓取view上的位图：

```
view.setDrawingCacheEnabled(true);
view.buildDrawingCache();
Bitmap bitmap = view.getDrawingCache();
```

以上是第二种方案的简化的部分代码，实际使用的时候可能还有一些细节需要处理。
以下分析下这个方案的优缺点。

优点： 几乎对所有的view触摸事件我们都可以抓取到了，而且用户无感知，方法统一，不管是单击，长按，滑动，双击等，我们都可以知道，甚至按下的时间毫秒数都可以知道。侵入性非常小，只需要在Appcation中注册一下。

缺点： 缺点就在于根据坐标找到触摸到的View上，如果这里的View层级比较深，而且view特别多的话，可能会有一些性能上的问题，不过这个问题我们可以通过一些缓存的逻辑来规避优化，另外对于view的遍历，这里应该是比findviewById还要快的，因为我们只是拿到rootView.getTouchables()来遍历。我在我手机上试了一下200多个view也只是再几毫秒之内就完成了。

图片抓取到，上传的问题

我们当然是可以直接上传这张图片的，点击抓到就上传，不过因为我们仅仅是统计，上传整张图片对我们来说代价还是比较大，此方法pass掉

优化一下这个方案：

在上传之前，把这个图片统一缩放到一个固定的尺寸，然后计算他的hash值，然后查询这个hash值对应的图片是不是在后端存在了，如果存在就统计值+1，不存在就传上去。伪代码：

```
Bitmap bitmap = getViewBitMap();
Bitmap newBitmap= zoom(bitmap);
String hashCode = get bitmap hashCode;

if(server this hashcode is empty){
    count++;
}else{
    upload this bitmap;
```

```
}
```

对于hashcode计算的问题，可以使用常规的方法，我认为也可以分别使用计算图片垂直方向和水平方向上，最顶，最左边，最低，最右边，最中间一行像素值来计算，这样即使是图片缩放了，其像素序列肯定也是存在于原图序列中的，我们可以很简单的计算一个缩放后的图与原图是否对应，因此甚至还可以从原图，主动穷举出所有的hash值。因此我们可以根据图片是否裁剪，来使用不同的hash计算方法

用图片最外边像素和中间像素序列计算出来的hashcode

```
if（hash 存在于 原图主动穷举出来的可能的hashcodes ？）{
    count++;
}else{
    用普通方法计算hashcode值
    if（hash 存在于 数据库后来记录的hashcodes ？）{
        count++;
    }else{
        上传图片
    }
}
```

这里再说下后台的处理，因为一张原图在不同的机型上可能展示的尺寸可能不一样，因此这里即使是缩放尺寸相同之后，也有可能出现hashcode计算不一致的情况，因此这里后台的存储结构，图片跟hashcode应该是一对多的，后端在入库前端传上来的图片的时候，可以使用Gist特征提取算法，将一组图归为一张图对应多个hashcode。一方面减少存储空间，一方面还可以归纳统计数据。

对于这个问题目前就想到了这个方案，对于其他的标签绑定，View.setTag(k,v);，提供一组自定义UI库给开发者使用什么的，总感觉代码侵入性太高，就不说了。

结束

感谢您的阅读，技术尚浅，以上文字中可能还会有一些疏漏的地方，这个还是需要结合项目更具体的一些因素来考量，欢迎您的批评指正。

另外对于这个问题本身的需求上，我的理解可能还存在一些偏差，如果以上有哪些没有理解到点上，请您指出。

祝好！