《百度地图实战 实验手册 03》

Android 课程组

版 本 1.0

文档提供：Java 课程组 丁盟

目录

第3章　百度地图地图覆盖物

3.1 实验目的

目的一： 了解百度地图提供的覆盖物类型。

目的二： 掌握基本的覆盖物创建及使用流程。

目的三： 通过覆盖物实现一些基本应用。

目的四： 了解百度地图图层的概念。

3.2 准备工作

准备一： 创建项目。

使用 Android Studio 创建一个新的空项目，并按照之前百度地图 SDK Android 开发环境配置对当前新项目进行配置。

准备二： 申请秘钥。

对当前项目申请百度地图开发者秘钥，并在项目中对秘钥信息进行设置。

当前面准备工作完成之后，在模拟器或真机中运行项目能够显示如下效果即表示准备工作已完成。

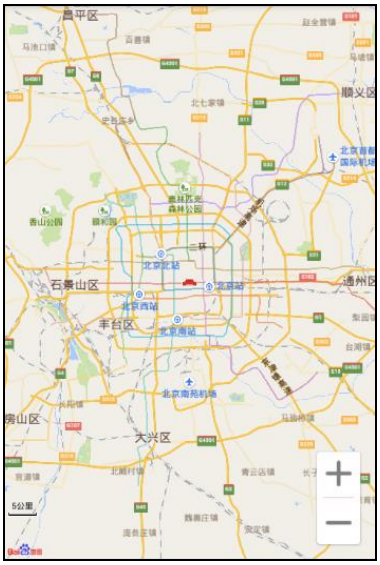

图 3.2.1

3.3 实验步骤

步骤一 在 MainActivity 中分别声明地图视图、地图控制器、地图 UI 控制器三个属性成员。

```
// 百度地图视图
private MapView mMapView = null;
// 百度地图控制器
private BaiduMap mBaiduMap = null;
// 百度地图UI控制器
private UiSettings mUiSettings = null;
```

步骤二 在 MainActivity 中创建 initBaiduMap()方法，用来完成百度地图初始化工作，并在 onCreate()中调用。

```
/* 初始化Baidu地图相关设置 */
private void initBaiduMap() {
    // 获取地图视图
    mMapView = (MapView) findViewById(R.id.bmapView);
    // 获取地图控制器
    mBaiduMap = mMapView.getMap();
    // 设置比例尺为500M
    MapStatusUpdate msu = MapStatusUpdateFactory.zoomTo(15.0f);
    mBaiduMap.setMapStatus(msu);
    // 获取地图UI控制器
    mUiSettings = mBaiduMap.getUiSettings();
    // 隐藏指南针
    mUiSettings.setCompassEnabled(false);
}
```

步骤三 在 res 目录下创建 menu 目录，并添加 main.xml 文件。

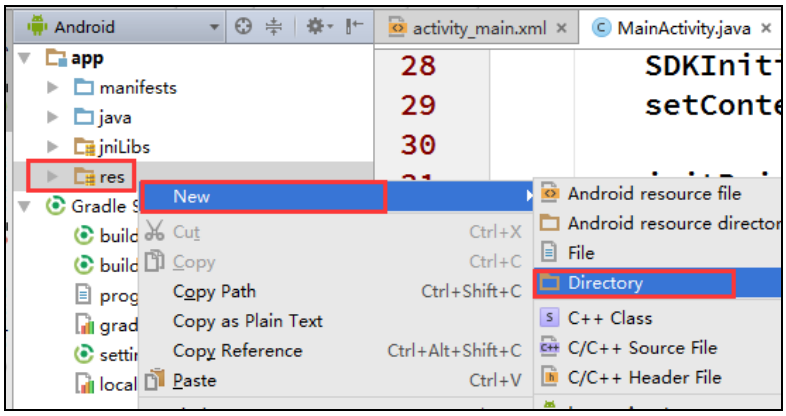

图 3.3.1

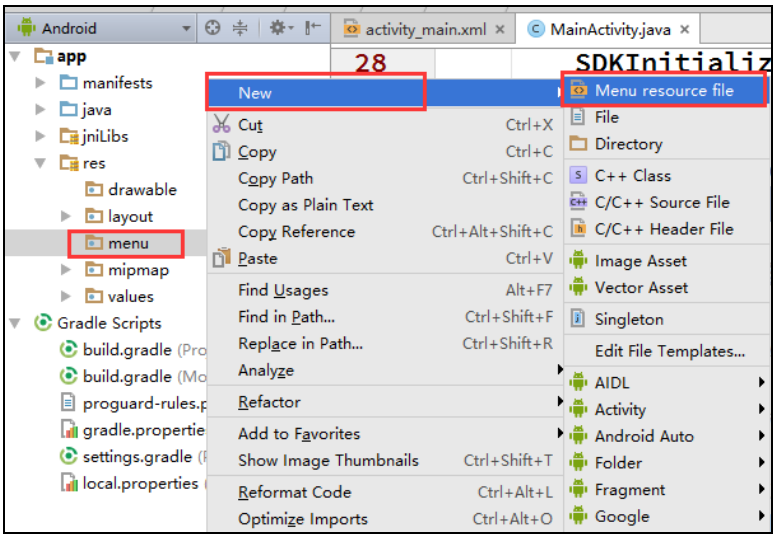

图 3.3.2

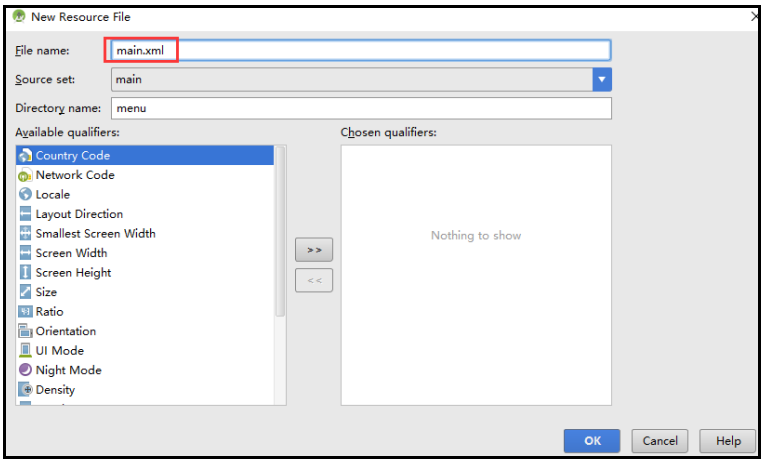

图 3.3.3

步骤四 当 main.xml 文件创建完成之后，在其中添加如下 item 项。(main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/android
"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/id_map_mark"
        android:title="标注覆盖物"
        app:showAsAction="never" />

    <item
        android:id="@+id/id_map_polygon"
```

```
        android:title="多边形覆盖物"
        app:showAsAction="never" />

    <item
        android:id="@+id/id_map_polyline"
        android:title="折线覆盖物"
        app:showAsAction="never" />

    <item
        android:id="@+id/id_map_text"
        android:title="文本覆盖物"
        app:showAsAction="never" />

    <item
        android:id="@+id/id_map_ground"
        android:title="地形图图层覆盖物"
        app:showAsAction="never" />

    <item
        android:id="@+id/id_map_infoWindow"
        android:title="弹出窗覆盖物"
        app:showAsAction="never" />

    <item
        android:id="@+id/id_map_clear"
        android:title="清空覆盖物"
        app:showAsAction="never" />

</menu>
```

步骤五 在 MainActivity 类中实现菜单创建接口。

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu, menu);
    return true;
}
```

步骤六 在 MainActivity 类中实现菜单点击相应接口。

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.id_map_mark:      // 标注覆盖物
            addMarkerOverlay();
            break;
```

```
        case R.id.id_map_polygon:   // 多边形覆盖物
            addPolygonOverlay();
            break;

        case R.id.id_map_polyline:  // 添加折线覆盖物
            addPolylineOverlay();
            break;

        case R.id.id_map_text:      // 文本覆盖物
            addTextOverLay();
            break;

        case R.id.id_map_ground:    // 地形图图层覆盖物
            addGroundOverLay();
            break;

        case R.id.id_map_infoWindow:// 弹出窗覆盖物
            addInfoWindow();
            break;

        case R.id.id_map_clear:     // 清空
            mBaiduMap.clear();
            break;
    }

    return super.onOptionsItemSelected(item);
}
```

步骤七 依次实现 addMarkerOverlay()、addPolygonOverlay()、addPolylineOverlay()、addTextOverLay()、addGroundOverLay()、addInfoWindow()函数，完成不同类型覆盖物的创建及显示。（代码详见MainActivity.java）

步骤八 创建 setMarkerListener()函数用来响应标注覆盖物操作，并在initBaiduMap()函数中完成调用。

```
private void setMarkerListener() {
    // 调用BaiduMap对象的setOnMarkerClickListener方法
    // 设置marker点击事件的监听
    mBaiduMap.setOnMarkerClickListener(new
BaiduMap.OnMarkerClickListener() {
```

```
        public boolean onMarkerClick(Marker marker) {
            // 点击处理
            Toast.makeText(MainActivity.this,
                    marker.getTitle(),
                    Toast.LENGTH_SHORT)
                    .show();
            return false;
        }
    });

    // 调用BaiduMap对象的setOnMarkerDragListener方法
    // 设置marker拖拽事件的监听
    mBaiduMap.setOnMarkerDragListener(new
BaiduMap.OnMarkerDragListener() {
        private boolean bFirst = true;
        public void onMarkerDrag(Marker marker) {
            // 拖拽中
            if(bFirst) {
                Toast.makeText(MainActivity.this,
                        "拖拽中",
                        Toast.LENGTH_SHORT)
                        .show();
                bFirst = false;
            }
        }
        public void onMarkerDragEnd(Marker marker) {
            // 拖拽结束
            Toast.makeText(MainActivity.this,
                    "拖拽结束",
                    Toast.LENGTH_SHORT)
                    .show();
            bFirst = true;
        }
        public void onMarkerDragStart(Marker marker) {
            // 开始拖拽
            Toast.makeText(MainActivity.this,
                    "开始拖拽",
                    Toast.LENGTH_SHORT)
                    .show();
        }
    });
}
```

步骤九 最终 MainActivity 类完整代码如下。(MainActivity.java)

```
public class MainActivity extends AppCompatActivity {
    // 百度地图视图
    private MapView mMapView = null;
    // 百度地图控制器
    private BaiduMap mBaiduMap = null;
    // 百度地图UI 控制器
    private UiSettings mUiSettings = null;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        SDKInitializer.initialize(getApplicationContext());
        setContentView(R.layout.activity_main);

        initBaiduMap();
    }

    // =============接口实现============================
    /**
     * 菜单创建
     * @param menu
     * @return
     */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    /**
     * 菜单点击响应
     * @param item
     * @return
     */
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.id_map_mark:      // 标注覆盖物
                addMarkerOverlay();
                break;

            case R.id.id_map_polygon:   // 多边形覆盖物
```

```
                addPolygonOverlay();
                break;

            case R.id.id_map_polyline:  // 添加折线覆盖物
                addPolylineOverlay();
                break;

            case R.id.id_map_text:      // 文本覆盖物
                addTextOverLay();
                break;

            case R.id.id_map_ground:    // 地形图图层覆盖物
                addGroundOverLay();
                break;

            case R.id.id_map_infoWindow:// 弹出窗覆盖物
                addInfoWindow();
                break;

            case R.id.id_map_clear:     // 清空
                mBaiduMap.clear();
                break;
        }

        return super.onOptionsItemSelected(item);
    }

    // =================自定义函数====================
    /**
     * 始化 Baidu 地图相关设置
     */
    private void initBaiduMap() {
        // 获取地图视图
        mMapView = (MapView) findViewById(R.id.bmapView);
        // 获取地图控制器
        mBaiduMap = mMapView.getMap();
        // 设置比例尺为 500M
        MapStatusUpdate msu =
MapStatusUpdateFactory.zoomTo(15.0f);
        mBaiduMap.setMapStatus(msu);
        // 获取地图 UI 控制器
        mUiSettings = mBaiduMap.getUiSettings();
        // 隐藏指南针
        mUiSettings.setCompassEnabled(false);
```

```
        // 设置标注覆盖物的监听
        setMarkerListener();
    }

    /**
     * 设置标注覆盖物监听
     */
    private void setMarkerListener() {
        // 调用 BaiduMap 对象的 setOnMarkerClickListener 方法
        // 设置 marker 点击事件的监听
        mBaiduMap.setOnMarkerClickListener(new
BaiduMap.OnMarkerClickListener() {
            public boolean onMarkerClick(Marker marker) {
                // 点击处理
                Toast.makeText(MainActivity.this,
                        marker.getTitle(),
                        Toast.LENGTH_SHORT)
                        .show();
                return false;
            }
        });

        // 调用 BaiduMap 对象的 setOnMarkerDragListener 方法
        // 设置 marker 拖拽事件的监听
        mBaiduMap.setOnMarkerDragListener(new
BaiduMap.OnMarkerDragListener() {
            private boolean bFirst = true;
            public void onMarkerDrag(Marker marker) {
                // 拖拽中
                if(bFirst) {
                    Toast.makeText(MainActivity.this,
                            "拖拽中",
                            Toast.LENGTH_SHORT)
                            .show();
                    bFirst = false;
                }
            }
            public void onMarkerDragEnd(Marker marker) {
                // 拖拽结束
                Toast.makeText(MainActivity.this,
                        "拖拽结束",
                        Toast.LENGTH_SHORT)
                        .show();
```

```
                bFirst = true;
            }
            public void onMarkerDragStart(Marker marker) {
                // 开始拖拽
                Toast.makeText(MainActivity.this,
                        "开始拖拽",
                        Toast.LENGTH_SHORT)
                        .show();
            }
        });
    }

    /**
     * 添加标注覆盖物
     */
    private void addMarkerOverlay() {
        // 定义Maker坐标点
        LatLng point = new LatLng(39.913285, 116.403923);
        // 构建Marker图标
        BitmapDescriptor bitmap = BitmapDescriptorFactory
                .fromResource(R.drawable.marker);
        // 构建MarkerOption，用于在地图上添加Marker
        OverlayOptions option = new MarkerOptions()
                .position(point)    // 设置marker的位置
                .draggable(true)    // 设置是否允许拖拽
                .title("国旗")      // 设置marker的title
                .icon(bitmap);      // 必须设置marker图标
        //在地图上添加Marker，并显示
        Marker marker = (Marker)
mBaiduMap.addOverlay(option);
    }

    /**
     * 添加多边形覆盖物
     */
    private void addPolygonOverlay() {
        // 定义多边形的五个顶点
        LatLng pt1 = new LatLng(39.919857,  116.399054);
        LatLng pt2 = new LatLng(39.92023,  116.408118);
        LatLng pt3 = new LatLng(39.929084,  116.407651);
        LatLng pt4 = new LatLng(39.928731,  116.398578);

        List<LatLng> pts = new ArrayList<LatLng>();
        pts.add(pt1);
```

```
        pts.add(pt2);
        pts.add(pt3);
        pts.add(pt4);

        // 构建用户绘制多边形的Option 对象
        OverlayOptions polygonOption = new PolygonOptions()
                .points(pts)    // 点信息
                .stroke(new Stroke(5, 0xAAFF0000))  // 边框
                .fillColor(0xAAFFFF00); // 填充颜色

        // 在地图上添加多边形Option，用于显示
        mBaiduMap.addOverlay(polygonOption);
    }

    /**
     * 添加折线覆盖物
     */
    private void addPolylineOverlay() {
        // 构造折线点坐标
        List<LatLng> points = new ArrayList<LatLng>();
        points.add(new LatLng(39.913776,116.398039));
        points.add(new LatLng(39.913887,116.402063));
        points.add(new LatLng(39.909325,116.402409));
        points.add(new LatLng(39.909463,116.404107));
        // 构建分段颜色索引数组
        List<Integer> colors = new ArrayList<>();
        colors.add(Integer.valueOf(Color.RED));
        colors.add(Integer.valueOf(Color.YELLOW));
        colors.add(Integer.valueOf(Color.GREEN));
        OverlayOptions polyline = new PolylineOptions()
                .width(15)              // 宽度
                .colorsValues(colors)   // 颜色信息
                .points(points);        // 点信息
        // 添加在地图中
        mBaiduMap.addOverlay(polyline);
    }

    /**
     * 添加文本覆盖物
     */
    private void addTextOverLay() {
        // 定义文字所显示的坐标点
        LatLng llText = new LatLng(39.912739, 116.404017);
        // 构建文字Option 对象，用于在地图上添加文字
```

```
        OverlayOptions textOption = new TextOptions()
                .bgColor(0xAAFFFF00)    // 背景颜色
                .fontSize(32)           // 字号
                .fontColor(0xFFFF00FF)  // 字体颜色
                .text("天安门广场")      // 文本
                .position(llText);      // 位置
        // 在地图上添加该文字对象并显示
        mBaiduMap.addOverlay(textOption);

    }

    /**
     * 添加地形图图层覆盖物
     */
    private void addGroundOverLay() {
        // 定义 Ground 显示的图片
        BitmapDescriptor bdGround = BitmapDescriptorFactory
                .fromResource(R.drawable.gugong);

        LatLng southwest = new LatLng(39.919404,
116.398035);//西南
        LatLng northeast = new LatLng(39.92964,
116.408527);//东北
        LatLngBounds bounds = new LatLngBounds.Builder()
                .include(southwest)
                .include(northeast)
                .build();//得到一个地理范围对象

        // 定义 Ground 覆盖物选项
        OverlayOptions ooGround = new GroundOverlayOptions()
                .positionFromBounds(bounds)
                .image(bdGround)
                .transparency(0.8f);

        // 在地图中添加 Ground 覆盖物
        mBaiduMap.addOverlay(ooGround);

    }

    /**
     * 添加弹出窗覆盖物
     */
    private void addInfoWindow() {
        // 创建 infowindow 展示的 view
```

```
        TextView location = new
TextView(getApplicationContext());
        location.setBackgroundResource(R.drawable.popup);
        location.setPadding(15, 5, 15, 20);
        location.setText("天安门东");
        location.setTextColor(Color.WHITE);
        location.setGravity(Gravity.CENTER);

        BitmapDescriptor bitmapDescriptor =
BitmapDescriptorFactory
                .fromView(location);

        // Infowindow 点击事件
        InfoWindow.OnInfoWindowClickListener
infoWindowClickListener
                = new InfoWindow.OnInfoWindowClickListener() {
            @Override
            public void onInfoWindowClick() {
                // 隐藏 InfoWindow
                mBaiduMap.hideInfoWindow();
            }
        };

        // 创建 Infowindow
        LatLng point = new LatLng(39.914088, 116.407844);
        InfoWindow infoWindow
                = new InfoWindow(bitmapDescriptor,
                point,
                0,
                infoWindowClickListener);
        // 显示 InfoWindow
        mBaiduMap.showInfoWindow(infoWindow);

    }
}
```

3.4 实验结论

当编码工作完成后在模拟器或真机中运行项目，效果如下：

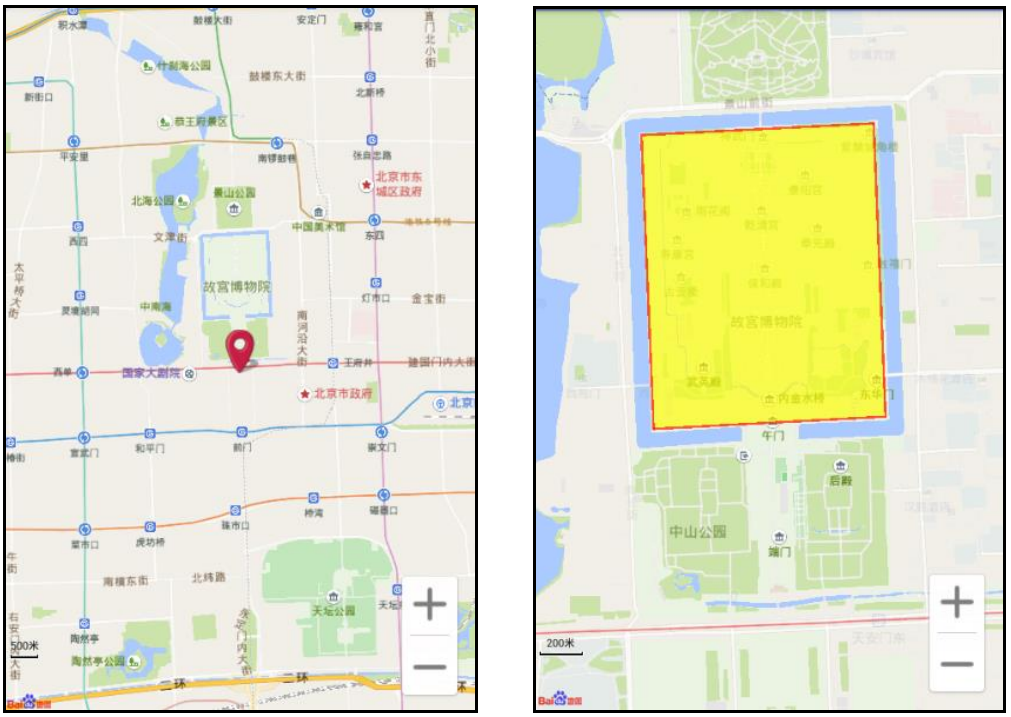
故宫博物院
中山公园
500米
200米

图 3.4.1

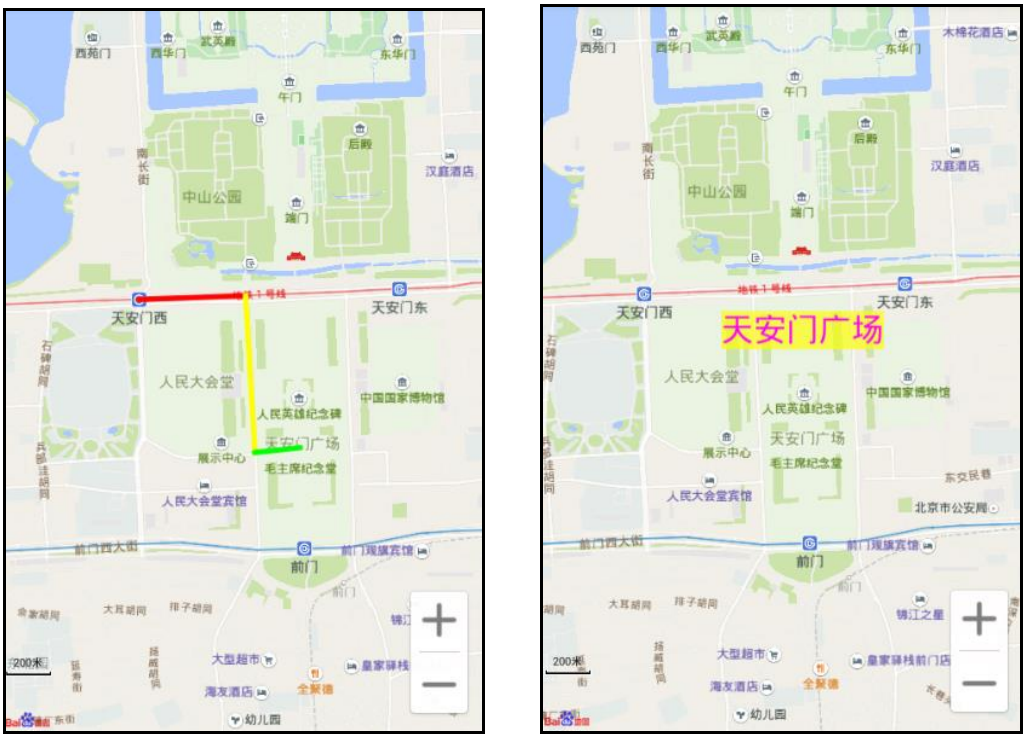
天安门西
天安门东
天安门广场
人民大会堂
中国国家博物馆
人民英雄纪念碑
毛主席纪念堂
前门
中山公园
200米

图 3.4.2


沙滩宾馆
红墙君豪
万春亭
北京工
故宫博物院
木棉花酒店
西苑门
午门
后殿
汉庭酒店
中山公园
端门
天安门西
天安门东
地铁1号线
200米
人民大会堂
王府世纪
武英殿
东华门
王府井大厦
东方文化大厦
南河沿大街
天安门东
北京饭店莱佛士
六方亭
长安大厦
人民大会堂
中国国家博物馆
人民英雄纪念碑
北京市政府
天安门广场
展示中心
毛主席纪念堂
东交民巷
人民大会堂宾馆
首都大酒店
前门
前门东大街
地铁2号线
锦江之星
大型超市
皇家驿栈前门店
海友酒店
全聚德

图 3.4.3