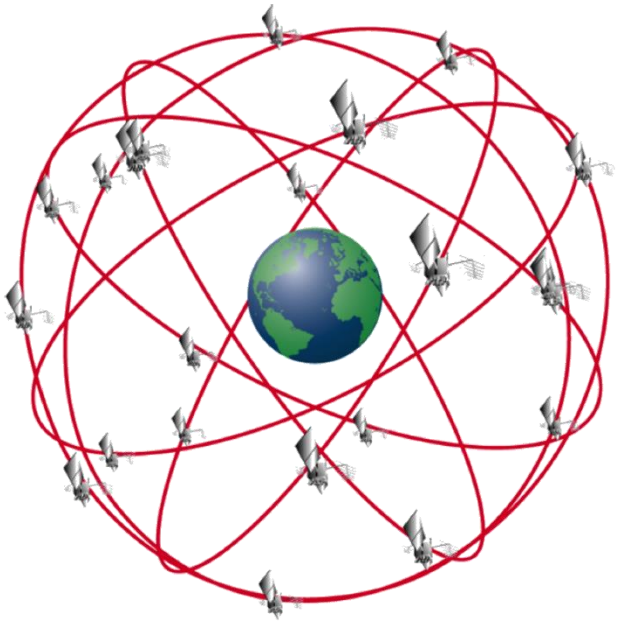
RTKLIB学习小组



rtkpos函数梳理

赵奕澎

2023.08.05

主要内容

- rtkpos函数作用及变量解析
- 函数流程
- 单点定位函数pntpos

rtkpos函数作用及变量解析


```
while ((nobs=inputobs(obs,rtk->sol.stat,popt))>=0) {

    /* exclude satellites */
    for (i=n=0;i<nobs;i++) {
        if ((satsys(obs[i].sat,NULL)&popt->navsys)&&
            popt->exsats[obs[i].sat-1]!=1) obs[n++]=obs[i];
    }
    if (n<=0) continue;

    /* carrier-phase bias correction */
    if (!strstr(popt->pppopt,"-ENA_FCB")) {
        corr_phase_bias_ssr(obs,n,&navs);
    }
    if (!rtkpos(rtk,obs,n,&navs)) {
        if (rtk->sol.eventime.time != 0) {
            if (mode == 0) {
                outinvalidtm(fptm, sopt, rtk->sol.eventime);
            } else if (!revs) {
                invalidtm[nitm++] = rtk->sol.eventime;
            }
        }
        continue;
    }
```

在procpos函数中，会在inputobs后(即观测值输入后)调用rtkpos函数主要作用就是**根据观测数据和导航信息，计算接收机的位置、速度和钟差。**

rtkpos函数作用及变量解析

变量1： **rtk_t *rtk rtk解算要使用到的结构体**

```
typedef struct {            /* RTK control/result type */
    sol_t  sol;             /* 存放结果的结构体 */
    double rb[6];           /* 基站的位置和速度 (m|m/s) */
    int nx,na;              /* 参数的个数，na为除模糊度外参数数、nx为加上模糊度参数数 */
    double tt;              /* 当前历元和先前历元的时间差 (s) */
    double *x, *P;          /* 浮点解和对应的协方差 */
    double *xa,*Pa;         /* 固定解和对应的协方差 */
    int nfix;               /* 连续固定解的个数 */
    int excsat;             /* index of next satellite to be excluded for partial ambiguity resolution */
                            /* 就是不使用的卫星对应的索引 */
    int nb_ar;              /* number of ambiguities used for AR last epoch */
                            /* 模糊度固定的时候用到的卫星个数 */
    double com_bias;        /* phase bias common between all sats (used to be distributed to all sats */
    /* com_bias: Phase bias common between all satellites.
    It represents a common bias that is shared among all satellites
    and can be used for carrier phase-based positioning. */
    char holdamb;           /* set if fix-and-hold has occurred at least once */
    /* holdamb: Flag indicating if fix-and-hold has occurred at least once,
    which refers to the usage of fixed/frozen ambiguities to aid in ambiguity resolution. */
    ambc_t ambc[MAXSAT];    /* 模糊度控制结构体数组 */
    ssat_t ssat[MAXSAT];    /* 卫星状态控制结构体数组 */
    int neb;                /* 错误信息的缓冲区长度 */
    char errbuf[MAXERRMSG]; /* 错误信息的缓冲区 */
    prcopt_t opt;           /* 进行处理的选项 */
    int initial_mode;       /* 最初的定位模式 */
} rtk_t;
```

rtkpos函数作用及变量解析

变量2： **obsd_t *obs 单历元的观测值**

```
typedef struct {                        /* 观测值结构体 */
    gtime_t time;                       /* 接收机的采样时间 (GPST) */
    uint8_t sat,rcv;                    /* 卫星/接收机的个数 */
    uint16_t SNR[NFREQ+NEXOBS];         /* 信号的强度(信噪比) (0.001 dBHz) */
    uint8_t  LLI[NFREQ+NEXOBS];         /* LLI标志(用于周跳检测) */
    uint8_t code[NFREQ+NEXOBS];         /* code indicator (CODE_???)  码的标志 */
    double L[NFREQ+NEXOBS];             /* 载波观测值 (cycle) */
    double P[NFREQ+NEXOBS];             /* 伪距观测值 (m) */
    float  D[NFREQ+NEXOBS];             /* 多普勒观测值 (Hz) */

    int timevalid;                      /* time is valid (Valid GNSS fix) for time mark */
    gtime_t eventime;                   /* time of event (GPST) */
    uint8_t qualL[NFREQ+NEXOBS];        /* quality of carrier phase measurement */
    uint8_t qualP[NFREQ+NEXOBS];        /* quality of pseudorange measurement */
    uint8_t freq;                       /* GLONASS frequency channel (0-13) */
} obsd_t;
```

变量3： **int n 观测值的个数**

rtkpos函数作用及变量解析

变量4： **nav_t *nav 导航电文结构体**

```
typedef struct {        /* navigation data type */
    int n,nmax;         /* number of broadcast ephemeris */
    int ng,ngmax;       /* number of glonass ephemeris */
    int ns,nsmax;       /* number of sbas ephemeris */
    int ne,nemax;       /* number of precise ephemeris */
    int nc,ncmax;       /* number of precise clock */
    int na,namax;       /* number of almanac data */
    int nt,ntmax;       /* number of tec grid data */
    eph_t *eph;         /* GPS/QZS/GAL/BDS/IRN ephemeris */
    geph_t *geph;       /* GLONASS ephemeris */
    seph_t *seph;       /* SBAS ephemeris */
    peph_t *peph;       /* precise ephemeris */
    pclk_t *pclk;       /* precise clock */
    alm_t *alm;         /* almanac data */
    tec_t *tec;         /* tec grid data */
    erp_t  erp;         /* earth rotation parameters */
    double utc_gps[8];  /* GPS delta-UTC parameters {A0,A1,Tot,WNt,dt_LS,WN_LSF,DN,dt_LSF} */
    double utc_glo[8];  /* GLONASS UTC time parameters {tau_C,tau_GPS} */
    double utc_gal[8];  /* Galileo UTC parameters */
    double utc_qzs[8];  /* QZS UTC parameters */
    double utc_cmp[8];  /* BeiDou UTC parameters */
    double utc_irn[9];  /* IRNSS UTC parameters {A0,A1,Tot,...,dt_LSF,A2} */
    double utc_sbs[4];  /* SBAS UTC parameters */
    double ion_gps[8];  /* GPS iono model parameters {a0,a1,a2,a3,b0,b1,b2,b3} */
    double ion_gal[4];  /* Galileo iono model parameters {ai0,ai1,ai2,0} */
    double ion_qzs[8];  /* QZSS iono model parameters {a0,a1,a2,a3,b0,b1,b2,b3} */
    double ion_cmp[8];  /* BeiDou iono model parameters {a0,a1,a2,a3,b0,b1,b2,b3} */
    double ion_irn[8];  /* IRNSS iono model parameters {a0,a1,a2,a3,b0,b1,b2,b3} */
    int glo_fcn[32];    /* GLONASS FCN + 8 */
    double cbias[MAXSAT][3]; /* satellite DCB (0:P1-P2,1:P1-C1,2:P2-C2) (m) */
    double rbias[MAXRCV][2][3]; /* receiver DCB (0:P1-P2,1:P1-C1,2:P2-C2) (m) */
    pcv_t pcvs[MAXSAT]; /* satellite antenna pcv */
    sbssat_t sbssat;    /* SBAS satellite corrections */
    sbsion_t sbsion[MAXBAND+1]; /* SBAS ionosphere corrections */
    dgps_t dgps[MAXSAT]; /* DGPS corrections */
    ssr_t ssr[MAXSAT];  /* SSR corrections */
} nav_t;
```

rtkpos函数流程

1. 设置rtk内基准站坐标rtk->rb，速度设为0.0，基准站坐标execses()函数内调用antpos()函数根据选项获取：

- postype=POSOPT_SINGLE：调用avepos()利用基准站的观测文件计算其SPP定位结果平均值作为基准站的坐
- postype=POSOPT_FILE：调用getstapos()从pos文件读取基准站坐标 。
- postype=POSOPT_RINEX ：从rinex头文件中获取测站经过相位中心改正的位置数据。头文件中的测站数据经过读取后已存到stas中。

```
/* set base station position */
if (opt->refpos<=POSOPT_RINEX&&opt->mode!=PMODE_SINGLE&&
    opt->mode!=PMODE_MOVEB) {
    /* 基准站坐标在execses()函数中计算 */
    /* 速度都先设为0 */
    for (i=0;i<6;i++) rtk->rb[i]=i<3?opt->rb[i]:0.0;
}
```

rtkpos函数流程

2. 统计基准站(nu)和观测站(nr)观测值的个数

```
/* count rover/base station observations */
/* 分别统计基准站观测值的个数nu和流动站观测值的个数nr */
for (nu=0;nu   <n&&obs[nu   ].rcv==1;nu++) ;
for (nr=0;nu+nr<n&&obs[nu+nr].rcv==2;nr++) ;
```

3. 获取得到上一历元的时间

```
/* 这里还未更新rtk->sol.time，所以是上一历元的时间 */
time=rtk->sol.time; /* previous epoch */
```

rtkpos函数流程

4. 调用pntpos函数进行SPP解算

```
/* rover position by single point positioning */
/* 这里会先计算SPP结果，作为kalman滤波的近似坐标 */
if (!pntpos(obs,nu,nav,&rtk->opt,&rtk->sol,NULL,rtk->ssat,msg)) {
    /* SPP解算失败，这个时候会进到这里 */
    errmsg(rtk,"point pos error (%s)\n",msg);

    if (!rtk->opt.dynamics) {
        outsolstat(rtk,nav);
        return 0;
    }
}
```

5. 计算当前历元与上一历元的时间差

```
if (time.time!=0) rtk->tt=timediff(rtk->sol.time,time);
```

rtkpos函数流程

6. 对时间差进行判断，大于5分钟就重新进入初始状态

```
/* return to static start if long delay without rover data */
/* 当时间差过大的时候，可以认为解不可靠了，这个时候就相当于重新初始化 */
 if (fabs(rtk->tt)>300&&rtk->initial_mode==PMODE_STATIC_START) {
    rtk->opt.mode=PMODE_STATIC_START;
    for (i=0;i<3;i++) initx(rtk,rtk->sol.rr[i],VAR_POS,i);
    if (rtk->opt.dynamics) {
        for (i=3;i<6;i++) initx(rtk,1E-6,VAR_VEL,i);
        for (i=6;i<9;i++) initx(rtk,1E-6,VAR_ACC,i);
    }
    trace(3,"No data for > 5 min: switch back to static mode:\n");
}
```

7. SPP模式就可以直接输出刚才计算得到的结果

```
/* single point positioning */
if (opt->mode==PMODE_SINGLE) {
    outsolstat(rtk,nav);
    return 1;
}
```

rtkpos函数流程

8. 不是单点定位的模式，就不输出单点定位的解

```
/* suppress output of single solution */
/* 非单点定位模式，就不输出单点定位的解 */
if (!opt->outsingle) {
    rtk->sol.stat=SOLQ_NONE;
}
```

9. PPP模式就调用pppos函数进行PPP的解算

```
/* precise point positioning */
/* PPP模式，调用pppos进行PPP解算 */
if (opt->mode>=PMODE_PPP_KINEMA) {
    pppos(rtk,obs,nu,nav);
    outsolstat(rtk,nav);
    return 1;
}
```

rtkpos函数流程

10. 对差分龄期(age of differential)进行判断

```
else {
    rtk->sol.age=(float)timediff(obs[0].time,obs[nu].time); // 移动站相对基准站的时间差

    /* 时间差过大的情况下直接返回 */
    if (fabs(rtk->sol.age)>opt->maxtdiff) {
        errmsg(rtk,"age of differential error (age=%.1f)\n",rtk->sol.age);
        outsolstat(rtk,nav);
        return 1;
    }
}
```

关于**差分龄期**：差分龄期即基站和移动站之间的时间差。在实时定位中，可能由于时间延迟、网络故障等原因未能接收到当前时刻的基站信息，在后处理中，基站数据可能中间有丢失，或者基站数据不能完全覆盖移动站。基站数据和移动站数据有时间差，并不意味不能进行定位。通常接收机的定位信息中都会包含age信息，并且也可以设置接收机的最大age。age越大，往往意味着定位精度的下降。

http://t.csdn.cn/MFrus

rtkpos函数流程

11. 调用relpos函数进行RTK解算
    调用outsolstat函数输出结果

```
/* relative potitioning */
/* 核心函数 */
relpos(rtk,obs,nu,nr,nav);
outsolstat(rtk,nav);
```

单点定位求解函数pntpos

通俗的说，SPP是所有解算模式的基础，因此会在rtkpos函数里直接调用，函数对应的变量如下：

obsd_t *obs	观测数据
int n	卫星个数
nav_t *nav	导航（星历）数据
prcopt_t *opt	处理策略选项
sol_t *sol	求解
double *azel	仰角/卫星仰角
ssat_t *ssat	卫星状态
char *msg	错误信息

单点定位求解函数pntpos

函数定义了一些之后解算会用到的矩阵

```
/**
 * rs      位置速度
 * dts     钟差(卫星+接收机)
 * var     协方差
 * azel_   方位角+仰角
 * resp    残差
*/
rs=mat(6,n); dts=mat(2,n); var=mat(1,n); azel_=zeros(2,n); resp=mat(1,n);
```

对于大气延迟，在不是SPP模式的情况下，进行设置

```
/* 如果处理选项不是SPP，就设置电离层和对流层分别为Klobuchar和Saastmoinen */
if (opt_.mode!=PMODE_SINGLE) { /* for precise positioning */
    opt_.ionoopt=IONOOPT_BRDC;
    opt_.tropopt=TROPOPT_SAAS;
}
```

单点定位求解函数pntpos

计算卫星位置、速度

```
/* satellite positons, velocities and clocks */
/* 计算卫星位置、速度和钟差 */
satposs(sol->time,obs,n,nav,opt_.sateph,rs,dts,var,svh);
```

利用伪距观测值估计位置

```
/* estimate receiver position with pseudorange */
/* 利用伪距观测值估计接收机的位置 */
stat=estpos(obs,n,rs,dts,var,svh,nav,&opt_,ssat,sol,azel_,vsat,resp,msg);
```

卫星完好性检测

```
/* RAIM FDE */
if (!stat&&n>=6&&opt->posopt[4]) {
    stat=raim_fde(obs,n,rs,dts,var,svh,nav,&opt_,ssat,sol,azel_,vsat,resp,msg);
}
```

单点定位求解函数pntpos

利用多普勒观测值估计速度

```
/* estimate receiver velocity with Doppler */
if (stat) {
    /* 利用多普勒观测值进行速度的估计 */
    estvel(obs,n,rs,dts,nav,&opt_,sol,azel_,vsat);
}
```

结果变量赋值

```
if (azel) {
    for (i=0;i<n*2;i++) azel[i]=azel_[i];
}
if (ssat) {
    for (i=0;i<MAXSAT;i++) {
        ssat[i].vs=0;
        ssat[i].azel[0]=ssat[i].azel[1]=0.0;
        ssat[i].resp[0]=ssat[i].resc[0]=0.0;
    }
    for (i=0;i<n;i++) {
        ssat[obs[i].sat-1].azel[0]=azel_[  i*2];
        ssat[obs[i].sat-1].azel[1]=azel_[1+i*2];
        if (!vsat[i]) continue;
        ssat[obs[i].sat-1].vs=1;
        ssat[obs[i].sat-1].resp[0]=resp[i];
    }
}
```

单点定位—残差计算rescode

pntpos函数的关键步骤在estpos函数里
而estpos函数里通过调用**rescode函数**进行了残差的计算

先进行坐标转换，从ecef坐标系转换至大地坐标系下(WGS84-BLH)

```
/* 坐标转换 */
ecef2pos(rr,pos);
```

拒绝重复观测，并剔除不使用的卫星

```
/* reject duplicated observation data */
/* 拒绝重复观测 */
if (i<n-1&&i<MAXOBS-1&&sat==obs[i+1].sat) {
    trace(2,"duplicated obs data %s sat=%d\n",time_str(time,3),sat);
    i++;
    continue;
}
/* excluded satellite? */
/* 剔除不使用的卫星 */
if (satexclude(sat,vare[i],svh[i],opt)) continue;
```

单点定位—残差计算rescode

卫地距以及卫星高度角、方位角计算

```
/* geometric distance and elevation mask*/
/* 卫地距和方位角、高度角计算 */
if ((r=geodist(rs+i*6,rr,e))<=0.0) continue;
if (satazel(pos,e,azel+i*2)<opt->elmin) continue;
```

然后判断信号是否可用

```
/* test SNR mask */
/* 根据接收机高度角和信号频率来检测该信号是否可用 */
if (!snrmask(obs+i,azel+i*2,opt)) continue;
```

单点定位—残差计算rescode

计算电离层延迟

```
/* ionospheric correction */
/* 电离层延迟 */
if (!ionocorr(time,nav,sat,pos,azel+i*2,opt->ionoopt,&dion,&vion)) {
    continue;
}
/* 电离层延迟和信号频率相关，所以改到对应的频率上 */
if ((freq=sat2freq(sat,obs[i].code[0],nav))==0.0) continue;
dion*=SQR(FREQL1/freq);
vion*=SQR(FREQL1/freq);
```

计算对流层延迟

```
/* tropospheric correction */
/* 对流层延迟 */
if (!tropcorr(time,nav,pos,azel+i*2,opt->tropopt,&dtrp,&vtrp))
continue;
```

单点定位—残差计算rescode

码偏差改正

```
/* psendorange with code bias correction */
/* 码偏差改正 */
if ((P=prange(obs+i,nav,opt,&vmeas))==0.0) continue;
```

计算对流层延迟

```
/* pseudorange residual */
/* 残差计算 */
v[nv]=P-(r+dtr-CLIGHT*dts[i*2]+dion+dtrp);
```

单点定位—残差计算rescode

组装设计矩阵H

The measurement equation and its partial derivative matrix for the single point positioning are formed as:

$$\boldsymbol{h}(\boldsymbol{x})=\begin{pmatrix}\rho_r^1+cdt_r-cdT^1+I_r^1+T_r^1\\ \rho_r^2+cdt_r-cdT^2+I_r^2+T_r^2\\ \rho_r^3+cdt_r-cdT^3+I_r^3+T_r^{s3}\\ \vdots\\ \rho_r^m+cdt_r-cdT^m+I_r^m+T_r^m\end{pmatrix}\quad \boldsymbol{H}=\begin{pmatrix}-\boldsymbol{e}_r^{1T} & 1\\ -\boldsymbol{e}_r^{2T} & 1\\ -\boldsymbol{e}_r^{3T} & 1\\ \vdots & \vdots\\ -\boldsymbol{e}_r^{mT} & 1\end{pmatrix} \tag{F.6.21}$$

where the geometric range ρ_r^s and LOS vector $\boldsymbol{e}_r^s$ are given by E.3 (4) and E.3 (5) with the satellite and receiver positions. The satellite positions $\boldsymbol{r}^s$ and the clock biases dT^s are also derived from the GNSS satellite ephemerides and clocks described in E.4 according to the processing option "Satellite Ephemeris/Clock".

单点定位—残差计算rescode

组装设计矩阵H

```
/* design matrix */
/* 组装设计矩阵H单位向量的反,前3行为中计算得到的视线向,第4行为1,其它行为0 */
for (j=0;j<NX;j++) {
    H[j+nv*NX]=j<3?-e[j]:(j==3?1.0:0.0);
}
/* time system offset and receiver bias correction */
/* 处理不同系统（GPS、GLO、GAL、CMP）之间的时间偏差，修改矩阵H */
if      (sys==SYS_GLO) {v[nv]-=x[4]; H[4+nv*NX]=1.0; mask[1]=1;}
else if (sys==SYS_GAL) {v[nv]-=x[5]; H[5+nv*NX]=1.0; mask[2]=1;}
else if (sys==SYS_CMP) {v[nv]-=x[6]; H[6+nv*NX]=1.0; mask[3]=1;}
else if (sys==SYS_IRN) {v[nv]-=x[7]; H[7+nv*NX]=1.0; mask[4]=1;}
```

单点定位—残差计算rescode

调用varerr函数，计算此时的导航系统误差，累加到URE上

```
/* variance of pseudorange error */
/* 伪距残差的协方差计算 */
var[nv++]=varerr(opt,azel[1+i*2],snr_rover,sys)+vare[i]+vmeas+vion+vtrp;
```

最后就是返回值了

```
/* constraint to avoid rank-deficient */
/* 防止秩亏，把H和var补满 */
for (i=0;i<NX-3;i++) {
    if (mask[i]) continue;
    v[nv]=0.0;
    for (j=0;j<NX;j++) H[j+nv*NX]=j==i+3?1.0:0.0;
    var[nv++]=0.01;
}
/**
 * 返回值v和resp的主要区别在于长度不一致，v是需要参与定位方程组的解算的，维度为 nv*1
 * 而resp仅表示所有观测卫星的伪距残余，维度为 n*1，对于没有参与定位的卫星，该值为 0
 */
return nv;
```

单点定位—结果检验valsol

结果检验第一种是进行卡方检验

```
/* Chi-square validation of residuals */
/* 卡方检验 */
vv=dot(v,v,nv);
/* chisqr (alpha=0.001时的卡方检验表) */
if (nv>nx&&vv>chisqr[nv-nx-1]) {
    sprintf(msg,"Warning: large chi-square error nv=%d vv=%.1f cs=%.1f",nv,vv,chisqr[nv-nx-1]);
    /* return 0; */ /* threshold too strict for all use cases, report error but continue on */
}
```

第二种是GDOP值检验

GDOP 全称为 Geometric Dilution of Precision，几何精度因子。它是衡量定位精度的一个很重要的系数，它代表卫星测距误差造成的接收机与空间卫星间的距离矢量放大因子，是理论选星算法的一个重要参数。

```
/* DOP值计算 */
dops(ns,azels,opt->elmin,dop);
if (dop[0]<=0.0||dop[0]>opt->maxgdop) {
    sprintf(msg,"gdop error nv=%d gdop=%.1f",nv,dop[0]);
    return 0;
}
```

单点定位—完好性检测raim_fde

完好性检测在raim_fde函数里进行

接收机自主完好性监测（RAIM: Receiver Autonomous Integrity Monitoring）是根据用户接收机的冗余观测值监测用户定位结果的完好性，其目的是在导航过程中检测出发生故障的卫星，并保障导航定位精度。

完好性检测包括**伪距残差判断法**、**伪距比较法**、**校验向量法**以及**最大解分离法**等，在rtklib中，使用的是第一种方法。

单点定位—速度估计estvel

速度估计的原理和位置估计的原理相似，主要的计算在estvel函数中
具体的残差计算原理可以参考手册(E6.27-30)

```
for (i=0;i<MAXITR;i++) {

    /* range rate residuals (m/s) */
    /* 速度残差的计算 */
    if ((nv=resdop(obs,n,rs,dts,nav,sol->rr,x,azel,vsat,err,v,H))<4) {
        break;
    }
    /* least square estimation */
    /* 最小二乘解算 */
    if (lsq(H,v,4,nv,dx,Q)) break;

    for (j=0;j<4;j++) x[j]+=dx[j];

    if (norm(dx,4)<1E-6) {
        matcpy(sol->rr+3,x,3,1);
        sol->qv[0]=(float)Q[0];  /* xx */
        sol->qv[1]=(float)Q[5];  /* yy */
        sol->qv[2]=(float)Q[10]; /* zz */
        sol->qv[3]=(float)Q[1];  /* xy */
        sol->qv[4]=(float)Q[6];  /* yz */
        sol->qv[5]=(float)Q[2];  /* zx */
        break;
    }
}
```

小结

主要内容

- ✓ rtkpos函数

重点

- ✓ rtkpos函数的整体流程
- ✓ pntpos函数进行单点定位