太阳影子定位

摘　要

本文分析物体在太阳下影子变化的情况，以《地面气象观测规范》中的气象观测公式为基础建立几何模型，以此求得已知杆长的影长变化，然后采用遗传算法，洪水填充，投影校正等算法来搜索确定物体所在位置以及拍摄时间。

建立模型一时，利用日期求得积日来计算真太阳时与地方平均太阳时之差与北京时间（以下简称北京时）之间的关系，再加上经度修正时求得数据采集地真太阳时与北京时的关系，从而可知太阳时角与北京时的函数关系。并求出赤纬，再联合纬度可求得太阳高度角的函数，而得到太阳高度角的函数后可用三角函数处理杆的高度求得影子长度随时间变化的曲线，从曲线中可知杆的影子于11:58取得最小值，此时预测影长为3.6516米。

模型二与模型三的建立依赖于模型一中所建立的各参数间的依赖关系。模型二主要对于固定高度（高度未知）的杆在具体日期的定位问题，即在模型一的基础上转换为经纬度由影子长度和杆长确定的函数关系。由于满足函数关系的可行解很多导致matlab无法求解。但在影长一定时，杆长和纬度有实相关性，于是我们对于杆长分类并依靠遗传算法处理约束条件进行搜索，以实现对不同纬度的分区间搜索，通过固定杆长，搜索经纬度，并比较理论影子长度和实际影子长度之间的误差，对可能解进行方差分析从而得到近似最优解。模型二中得到的可能的地点有云南省，海南岛等，其中有四个在海上，排除后还有四个陆上地点，模型二中得到可能结果如表一中所示。

模型三在模型二的基础上进行灵敏度分析确定主要影响参数，将模型转化为经纬度由影子长度，杆长和具体日期确定的函数关系。同样由于满足函数关系的可行解较多，而日期与经度有着一定的相关性，于是我们对于杆长和日期分类，利用遗传算法进行搜索并进行误差分析，取与实际影长数据的方差较小点作为近似最优解，以确保对不同经纬度的分区间搜索。模型三中得到的可能的地点有青海省，新疆维吾尔自治区等，模型三中得到可能结果如表二，表三中所示。

模型四将视频转化为多帧照片并降噪处理进行洪水填充得到图像和数据。利用基于相机的校正方法确定投影校正矩阵，从而得到杆的影子实际长度及其变化情况，将之转化为模型二进行求解。若未知拍摄日期则可转化成模型三进行求解。模型四中得到的可能的地点有一个，定位于41.0410 N，112.6320E，其位置位于内蒙古自治区。

本文在计算后决定不考虑大气折射对于太阳光角度的影响，以气象观测公式为基础建立了上述模型，使得该模型可在较小误差内能够根据物体的影子进行物体拍摄地点的经纬度与拍摄日期的确定。

关键词：气象观测模型 遗传算法 洪水填充 基于相机标定的投影校正 OpenCV(OPEN SOURCE COMPUTER VISION)

1 问题重述

确定视频的拍摄地点和拍摄日期是视频数据分析的重要方面，太阳影子定位技术就是通过分析视频中物体的太阳影子变化，确定视频拍摄的地点和日期的一种方法。

1.建立影子长度变化的数学模型，分析影子长度关于各个参数的变化规律，并应用你们建立的模型画出2015年10月22日北京时间9:00-15:00之间天安门广场（北纬39度54分26秒,东经116度23分29秒) 3米高的直杆的太阳影子长度的变化曲线。

2.根据某固定直杆在水平地面上的太阳影子顶点坐标数据，建立数学模型确定直杆所处的地点。将模型应用于附件1的影子顶点坐标数据，给出若干个可能的地点。

3.根据某固定直杆在水平地面上的太阳影子顶点坐标数据，建立数学模型确定直杆所处的地点和日期。将你们的模型分别应用于附件2和附件3的影子顶点坐标数据，给出若干个可能的地点与日期。

4.通过太阳下的影子变化的视频，已估计直杆的高度为2米。建立确定视频拍摄地点的数学模型，并应用你们的模型给出若干个可能的拍摄地点。

如果拍摄日期未知，应用模型根据视频确定出拍摄地点与日期。

2 问题分析

本文所研究的问题一主要是基础计算和气象观测问题，首先经度和拍摄时间可以计算积日订正日。而时差由积日，积日订正和年份对日期的影响决定。综合时差，时区时和经度订正确定真太阳时，而真太阳时可以求解出太阳时角。

直杆的顶点和影子顶点的连线与地平面夹角为太阳高度角，太阳高度角又与纬度，赤纬角和太阳时角满足函数关系，便可联立求得影子长度与拍摄时间的函数关系。

问题二问题与问题一大同小异。在问题一的模型的基础上，在已知日期的情形之下，确定出影子长度，经度，纬度，杆长这四个量满足的关系，然后带入影子长度数据求解方程。但在实际进行的过程中发现由于满足模型方程的解的个数太多而导致难以确定，不宜进行直接求解，所以用对全范围的点的搜索来代替直接求解影子长度。在搜索的过程里发现对函数关系进行约束将大大简化搜索过程以及使解变得有实际意义，于是选择使用遗传算法对约束问题进行规划。

在问题三中，发现日期也是未知的，而通过灵敏度分析年份对于赤纬角和时差甚至整个模型的系数很小，使得误差在允许的范围内因而忽略不计。

在忽略年份影响的情况下，问题三和问题二如出一辙。在问题二的几何模型中确定的影子长度，经纬度，杆长这四个量满足的模型，将之前给定的日期变为一个变量，然后再次进行搜索。在这次搜索的过程中意识到应该确定的地点的误差导致其他位置的精度比理想地点的精度高，而模型只搜索得出误差最小的地点而舍弃了理想地点。于是我们通过杆长的变化对纬度进行分区间，通过日期的变化对经度进行分区间，从而实现不同经纬度区间内的搜索，以免漏过理想地点。并借此想法对问题二进行改进。

而对于问题四，如何处理视频，将视频中影子长度提取出来是将问题转化为问题二或三从而解决该问题的关键所在。可以进行降噪后使用洪水填充，然后影子模块求得数据，再利用基于相机定位的投影校正及其相关方法，以其投影关系建立模型获取影长数据，将问题四转化为已解决的问题二或三。

3 模型的假设

- 假设一：地球围绕太阳的运动轨迹为椭圆，每年以365.2422天计算
- 假设二：假设太阳光始终是直线，忽略其散射，折射情况（大气，雾霾等影响忽略）
- 假设三：地球是标准球体，海拔忽略不计
- 假设四：假设所有地面水平
- 假设五：忽略所有测量误差，忽略物体厚度对于影子长度的影响
- 假设六：忽略夏令时

4 符号与公式的约定和说明

物理量符号	物理量含义	单位
E_q	真太阳时与地方平均太阳时的差值	小时
N	按天数顺序排列的积日	天数
ΔN	积日订正值	天数
N_0	年度修正	天数
L	观测地点与格林尼治经度产生的时间差订正值	小时
TT	真太阳时	时间格式
T_M	地方平均太阳时	时间格式
C_T	地方标准时	时间格式
L_C	经度订正(4 min/度)	小时
D_E	太阳赤纬	弧度
h	太阳高度角	弧度
t	太阳时角	弧度
$(n-1)_s$	标准状态下影子折射率	
σ	真空中的波数	μm^{-1}
$(n-1)_{tp}$	表示标准干燥空气在温度t(以℃表示), 压力P(以torr为单位表示)时的折射率	
$(n-1)_{tpf}-(n-1)_{tp}$	表示含有f水蒸气分压力的潮湿空气和 总压力相同的干燥空气的折射率之差	
H	理论的太阳高度角	弧度
A	太阳方位角	弧度
Φ	所在地的纬度	弧度
λ	所在地的经度	弧度
l_{gmax} bibitem1	杆长上限	米
l_{gmin}	杆长下限	米
t_{max}	日期上限	时间格式
t_{min}	日期下限	时间格式

Table 1: 参数说明表

5 模型的建立和求解

5.1 模型一的建立

5.1.1 模型的准备

对于时间计算的基础 有两个重要的概念:当地时间(太阳时间)和标准时间(时钟时间)。当地时间是用来衡量太阳相对于当地的位置。这是当地人们日常作息时间的规律标准。标准时间是某个时区标准子午线的太阳时间。为了获得当地太阳时间,在标准时间已知的情况下必须进行三种校准值计算。第一个校准值的产生是由于地球公转轨迹不是一个圆而是一个椭圆,并且它的旋转轴与其旋转平面不是垂直的。另外太阳与地球不是统一旋转的。这个重要的校准值最大可以产生早于或者晚于四分之一小时的误差,因此被称作地方平均太阳时（地平时）。第二个校准值的产生是当地经度需要与当地标准时区的中央子午线偏差。对于一个给定时区,如果物体位于该时区的中央子午线,那么当地时间与标准时间是一致的并只需要进行平衡时间的校准。如果物体位于中央子午线的西部(东部),那么物体的真太阳时将晚于(早于)当地标准时间。这种校准值可以通过对时间线进行某一角度的旋转

来获得。其中旋转角度是当地经度与中央子午线的差值。第三个校准值是夏令时时间。某些国家或地区在夏季会将标准时间提前一个小时。其取值为0或1小时（本文忽略）。[6]

对时间的变化使用数值模拟法，其实质就是先从某年的天文年历中将所需计算参数的逐日值录出作为应变量，以积日为自变量，然后借助傅立叶回归法，求出各项的相应系数。在本文中参考文献[7]

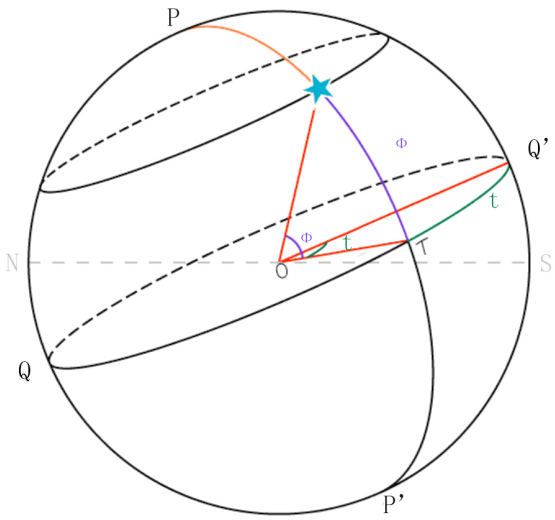

Figure 1: 时间角

本文采用新的国际单位、新的空气密度数据、IT-S90 国际温标、水蒸气折射率新数据、二氧化碳含量的增加，使用修正后的空气折射率埃德林公式，得出修改后的空气折射率计算公式为：

$$(n-1)_s \cdot 10^8 = 8324.54 + \frac{2406147}{130-\sigma^2} + \frac{15998}{38.9-\sigma^2} \tag{1}$$

$$(n-1)_{tp} = \frac{P(n-1)_s}{96095.43}\frac{[1+10^{-8}(0.601-0.00972t_{90})P]}{1+0.00361t_{90}} \tag{2}$$

$$n_{tpf} - n_{tp} = -f(3.7345 - 0.0341\sigma^2)10^{-10} \tag{3}$$

上式中，P 的单位为Pa。将数据代入上式，可计算出修正后含有水汽压的空气折射率。大气中修正后的折射率为 $\alpha = 1 + n_{tpf}$。 此外，在实际研究中发现，在早晚太阳光入射角较大时，无论是未修正的太阳高度角计算方法，还是考虑空气折射率的太阳高度角计算方法其误差均较大。所以，本文在计算早晚太阳高度角时通过乘加系数进行了修正，使数据更接近实测值。

由图2所示的太阳影子及大气折射示意图可以看出，当太阳光进入大气层时会发生折射，理论高度角$H = 90^o - R = 90^0 - i$。 根据斯涅尔定律可以得出修正大气折射率后的太阳高度角h，其计算公式为：

$$\alpha = \frac{\sin i}{\sin r} = \frac{\sin(90^o - H)}{\sin r} \tag{4}$$

$$\sin A = \frac{\sin(90^o - H)}{\alpha} \tag{5}$$

$$r = \arcsin[\frac{\sin(90^o - H)}{\alpha}] \tag{6}$$

因此，可得出考虑大气折射影响的太阳高度角h 为：

$$h = 90^o - \arcsin[\frac{\sin(90^o - H)}{\alpha}] \tag{7}$$

计算得 $n_{tpf} < 0.0005$，结合上述公式说明折射率对太阳高度角影响甚小，可以忽略折射率等的影响，因此本文在下文中需要考虑模型假设二。

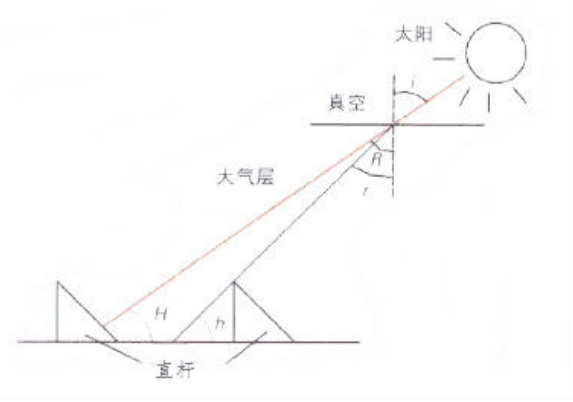

Figure 2: 折射图

5.1.2 模型的建立与计算

通过模型准备中的分析，我们已经知道了各个时间量之间的关系为： 时差E_q：

$$E_q = 0.0028 - 1.9857\sin Q + 9.9059\sin 2Q - 7.0924\cos Q - 0.6882\cos 2Q \tag{8}$$

Q：

$$Q = \frac{2\pi 57.3(N + \Delta N - N_0)}{365.2422} \tag{9}$$

式中N为按天数顺序排列的积日。1月1日是0；2日是1 . . . 依此类推。 积日订正日$\triangle N$：

$$\Delta N = \frac{W - L}{24} \tag{10}$$

时差订正值：

$$\pm L = \frac{\left(D + \frac{M}{60}\right)}{15} \tag{11}$$

其中D为观测点精度的度值，M为分值，换算成与格林尼治时间差L。东经为负，西经取正。 W为拍摄时间：

$$W = S + \frac{M}{60} \tag{12}$$

年份对积日的修正：

$$N_0 = 79.6764 + 0.2422(Y - 1985) - INT(0.25(Y - 1985)) \tag{13}$$

真太阳时TT：

$$TT = T_m + E_q = C_T + L_C + E_q \tag{14}$$

而空间角度之间的关系有：

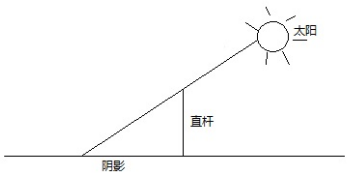

Figure 3:

太阳高度角满足的关系式h：

$$\tan h = \frac{L_g}{L_y} \tag{15}$$

而角度与经纬度的关系由：

$$\sin h = \sin \Phi \sin D_E + \cos \Phi \cos D_E \cos T_0 \tag{16}$$

赤纬角D_E ：

$$\begin{aligned} D_E = 0.3723 + 23.2576 \sin Q + 0.1149 \sin 2Q - 0.1712 \sin 3Q - 0.7580 \cos Q \\ +0.3656 \cos 2Q + 0.0201 \cos 3Q \end{aligned} \tag{17}$$

太阳时角t：

$$t = (TT - 12) \cdot 15^o \tag{18}$$

由题中给出 经度为东经116度23分29秒 纬度为北纬39度54分26 拍摄日期为2015年10月22日

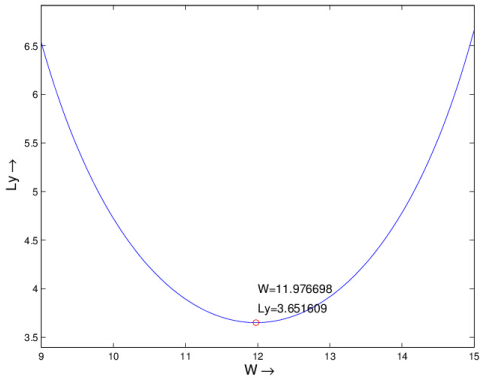

Figure 4: 影长变化图

由上述方程可求得影子长度与时间的关系，如4所示： 图4表明影子在当地真太阳时的12:00在北京时间11:58取得最小值，预测该影长为3.6516 米。

5.2 模型二的建立

5.2.1 模型的分析

本模型与模型一一脉相承，在几何模型方面几乎没有需要修改的部分；在此就不赘述了。

5.2.2 模型的建立与计算

模型二的解决采用遗传算法，从代表问题可能潜在的解集的一个种群开始。初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代演化产生出越来越好的近似解，在每一代，根据问题域中个体的适应度大小选择个体，并借助遗传算子进行组合交叉和变异，产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码，可以作为问题近似最优解。

在求解过程之中我们选择使用MATLAB自带遗传算法函数GA。

用遗传算法寻找函数的最优解,其语法规则为：

x = ga(fitnessfcn,nvars)
x = ga(fitnessfcn,nvars,A,b)
x = ga(fitnessfcn,nvars,A,b,Aeq,beq)
x = ga(fitnessfcn,nvars,A,b,Aeq,beq,LB,UB)

其中fitnessfc为函数的句柄或者为匿名函数 nvars,表示自变量个个数（例如自变量为向量X,nvars代表X中的元素个数） A，b就是表达式$A \cdot X <= b$; Aeq:表示线性等式约束矩阵，若是没有等式约束就写为[ ]; Beq:表示线性等式约束的个数Beq=length(nvars);

对此我们需要对模型中的参数进行以下约束 $-\pi < \lambda < \pi, 0 < \Phi < \frac{\pi}{2}, 0 < L_g < 4$ 有约束矩阵:

$$\mathbf{A}x \leq b \quad \mathbf{A} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \quad x = \begin{pmatrix} \phi \\ \lambda \\ L_g \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad b = \begin{pmatrix} \frac{\pi}{2} \\ \frac{\pi}{2} \\ \pi \\ \pi \\ l_{gmin} \\ l_{gmax} \end{pmatrix} \tag{19}$$

如此只能求得全局一个近似最优解，为了能够根据不同纬度进行分区域求解，对杆长进行区分，并进行搜索。

求解情况如下表所示： 对表中的部分点进行误差分析：

位置信息 / 杆长 L_g	纬度(单位是角度)	经度(单位是角度)	大致位置	方差
0.944363	28.1288	-130.6684	太平洋西侧日本下方	3.2029
1.949987	18.1105	-109.9501	海南岛南侧近三亚	0.0938
2.265514	21.6811	-104.9038	越南境内近河内	0.02745
3.381471	25.4876	-91.5369	印度梅加拉亚邦近西隆	0.4114
0.133458	78.6361	170.65255	北极圈外围	3.11641
1.863093	16.8230	−111.3985	海南岛以南西沙群岛	0.08625
2.656331	24.1613	−99.3641	云南省临沧市永德县 小勐统镇鸭塘村	0.0000
3.151356	25.3723	−93.6858	印度那加兰南侧	0.07322

Table 2: 可能的结果图(其中经度西经为正，北纬为正)

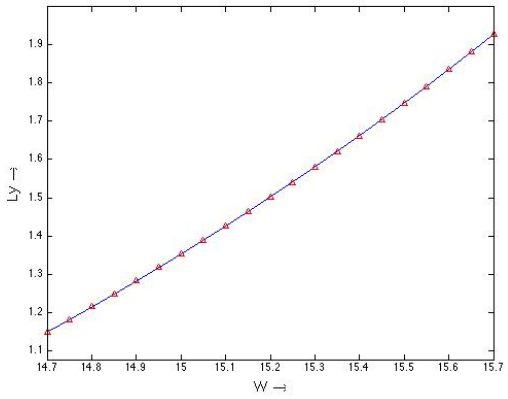

Figure 5: 附录一误差分析

由图中可见，所求点的影长预测与给出点的影长误差很小。故所求点应该与给出点距离不大。

5.3 模型三的建立

5.3.1 模型的分析

在模型二的基础上将一个已知量变为变量，仍然采用遗传算法对约束问题进行搜索求近似最优解。

5.3.2 模型的建立与计算

由于增加了一个变量，因而约束矩阵也相应地变为

$$\mathbf{A}x \leq b \quad \mathbf{A} = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad x = \begin{pmatrix} \phi \\ \lambda \\ L_g \\ N \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad b = \begin{pmatrix} \frac{\pi}{2} \\ \frac{\pi}{2} \\ \pi \\ \pi \\ l_{gmin} \\ l_{gmax} \\ t_{max} \\ t_{min} \end{pmatrix} \tag{20}$$

附录二的求解情况如下表所示：

杆长 L_g \ 日期	第一季度0 ~ 91	第二季度91 ~ 182	第三季度182 ~ 274	第四季度274 ~ 365
0 ~ 1	非洲以南的海上	中东地区里海北侧	印度洋北侧	非洲南侧
1 ~ 2	斯里兰卡 以西的海上	巴基斯坦北侧的乡下	印度洋北侧	非洲东南侧
2 ~ 3	印度洋中心	西藏自治区日喀则地区 聂拉木县	新疆维吾尔自治区 阿克苏地区沙雅县	印度洋中心
3 ~ 4	印度洋中心	印度东侧曼尼普尔 近锡尔杰尔	青海省海西蒙古族 藏族自治州格尔木市	印度洋中心

Table 3: 结果图，其中 绿色 代表方差[0,5] 浅绿 代表方差[5,10] 白色 代表方差[10,15] 浅红 代表方差[15,20] 红 代表方差[20,+∞]

由于解中南半球的点大都落在海洋中，模型将范围缩小至仅在北半球进行搜索。求得的解如下表所示：

杆长 L_g \ 日期	第一季度0 ~ 91	第二季度91 ~ 182	第三季度182 ~ 274	第四季度274 ~ 365
0 ~ 1	孟买西侧 阿拉伯海域	舍甫琴柯堡 西侧里海海域	马哈奇卡拉 东侧里海海域	阿曼东侧 阿拉伯海海域
1 ~ 2	马尔代夫西北 方向阿拉伯海域	阿富汗 中部地区	塔吉克斯坦 西南角	印度索姆 纳特东侧
2 ~ 3	斯里兰卡东南 方向印度洋海域	西藏自治区 日喀则地区吉隆县	新疆维吾尔自治区 阿克苏地区沙雅县	印度金奈东侧 阿拉伯海海域
3 ~ 4	印度尼西亚棉兰	西藏自治区 那曲地区那曲县	青海省海西 蒙古族藏族自治州	孟加拉湾安达 曼群岛西侧

Table 4: 结果图，其中 绿色 代表方差[0,5] 浅绿 代表方差[5,10] 白色 代表方差[10,15] 浅红 代表方差[15,20] 红 代表方差[20,+∞]

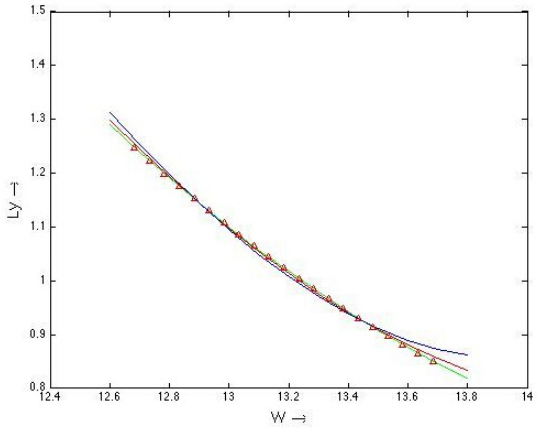

Figure 6: 附录二误差分析

对表中的点进行误差分析

红色代表位于西藏的点，绿色代表位于新疆的点，蓝色代表位于青海的点。

由图中可见，所求点的影长预测与给出点的影长误差很小。故所求点应该与给出点距离不大，且日期也是合理的。

相应的附录三的求解情况如下表所示：

杆长 L_g \ 日期	第一季度0 ~ 91	第二季度91 ~ 182	第三季度182 ~ 274	第四季度274 ~ 365
0 ~ 1	南极洲海岸边	印度洋南侧	澳大利亚西侧	澳大利亚西侧
1 ~ 3	南极洲海岸边	澳大利亚与大洋洲	蒙古曼达勒敖包	俄罗斯西伯利亚
3 ~ 4	广西壮族自治区	俄罗斯外贝加尔	澳大利亚西南侧	俄罗斯蒙古的国境线
4 ~ 6	澳大利亚与大洋洲	澳大利亚西南侧	俄罗斯萨哈共和国	非洲南侧
6 ~ 7	印度洋南侧	澳大利亚西侧	澳大利亚西侧	印度洋中心

Table 5: 结果图，其中 绿色 代表方差[0,5] 浅绿 代表方差[5,10] 白色 代表方差[10,15] 浅红 代表方差[15,20] 红 代表方差[20,+∞]

由于解中南半球的点大都落在海洋中，模型将范围缩小至仅在北半球进行搜索。求得的解如下表所示：

杆长 L_g \ 日期	第一季度0 ~ 91	第二季度91 ~ 182	第三季度182 ~ 274	第四季度274 ~ 365
0 ~ 1	蒙古巴彦德勒格尔	越南陆地最南端	蒙古共和国南侧	俄罗斯西伯利亚平原
1 ~ 3	湖南省益阳市桃江县	北冰洋中心	内蒙古自治区 阿拉善盟阿拉善左旗	蒙古共和国与 俄罗斯国境线北侧
3 ~ 4	湖南省永州市零陵区	俄罗斯西伯利亚 平原中心	北冰洋中心	蒙古共和国与 俄罗斯国境线北侧
4 ~ 6	柬埔寨东北侧	蒙古共和国	俄罗斯西伯利亚 平原北岸	蒙古共和国 乌兰巴托
6 ~ 7	越南陆地最南端	蒙古共和国南侧	俄罗斯西伯利亚 平原中部	蒙古自治区巴 内彦淖尔市乌拉特 中旗呼四线

Table 6: 结果图，其中 绿色 代表方差[0,5] 浅绿 代表方差[5,10] 白色 代表方差[10,15] 浅红 代表方差[15,20] 红 代表方差[20,+∞]

对表中的点进行误差分析

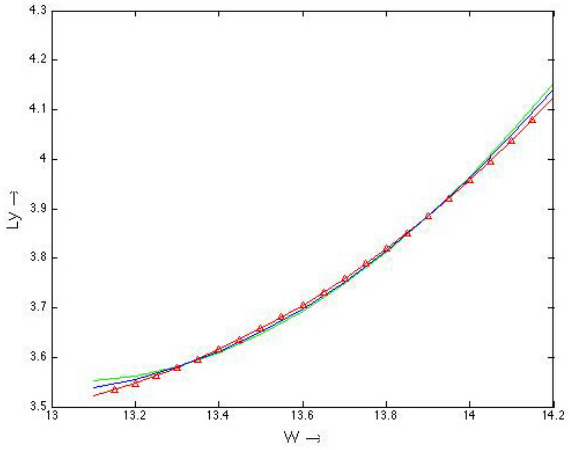

Figure 7: 附录三误差分析

红色代表位于湖南，绿色表示位于内蒙古，蓝色表示位于云南。

由图中可见，所求点的影长预测与给出点的影长误差很小。故所求点应该与给出点距离不大，且日期也是合理的。

5.4 模型四的建立

5.4.1 模型的基础

n维空间的透视和校正可以看成一种坐标的变化,对于坐标的处理可以看成乘以一个满秩矩阵。在这种映射下,无穷远的点被映射到任意的点。因此,n维透视空间的透视变换可以借由齐次坐标的线性变换来表示。

透视空间可以用较方便的方式来表示现实空间。类似的,通过映射世界坐标到一个2D空间表示的照片,通常会被延伸于二维透视空间的。而在真实世界中,照片平面并不包含位于无穷远的点,也就是照片中无穷远的直线和世界坐标中无穷远的平面。正是由于这个原因,意识到无穷远处的直线和平面在某些情况下是必要而特殊的。这种理解初见很颠覆，但是却十分适用。

照片平面中的任意一个点定义了一条空间中与无穷远平面相交于一个点的直线。

其中地平线估计方法需要平行阴影线。因此这样的阴影线可以提供与投影物体垂直的一个消隐点，又称为灭点。灭点的建立可以帮助我们求解部分数据。

5.4.2 模型的分析

根据题目描述，知道第四问给出了时间，日期，直杆长度，以及一段长度为40分钟40秒视角固定面透视视频，需要求解的变量为直杆所在地经度以及直杆所在地纬度。

由本文之前建立的第一个模型，可以得出一个影长关于时间，日期，直杆长度，直杆所在地经度，直杆所在地纬度的映像函数。 将其应用在第二个模型，既基于遗传算法的方程求近似解以及结果分治系统，可以得到一个输入时间-影长向量，日期，可按特许要求分类求出直杆所在地经度、直杆所在地经度、以及杆长的近似解的随机优化系统。

那么本题重点既在于如何从一段长度为40分钟40秒视角固定面透视视频中，精确、高效、稳定的提取出直杆影长在地面上的坐标，以及影子长度轨迹。

本文第四个模型自动化根据拍摄有影子的视频短片定位拍摄地点模型分为六大模块:

- 一. 自动化视频译码固定时间间隔图像保存模块
- 二. 人工干涉图像分析相关参数配置模块
- 三. 自动化图像像素编码转换以及基于阈值的灰度两极分化模块
- 四. 自动化基于洪水填充的色块分析以及智能影子色块信息查询输出模块
- 五. 自动化基于消隐点的面透视反向坐标系解析模块
- 六. 自动化数据收集以及与模型二的耦合及应用模块

其中模型四模块一使用开源应用ffmpeg，是一个完整的跨平台音视频录制、编码转换、数据流控制解决方案(A complete, cross-platform solution to record, convert and stream audio and video.) 。模型四模块三使用开源python语言库PIL(Python Imaging Library)，主要功能是为python解释器添加上图像处理功能，同时提供多样化的图片格式支持，高效的内部对象表示以及相对强大的图像处理功能(The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.) 。

除此以上开源应用以及开源python语言库以及OpenCV中开源代码以外，所有代码均为本文作者编写，并附有详细的注释，如需参考请见附录。

关于基于遗传算法的方程求近似解以及结果分治系统的讨论在此不再赘述，参考模型一及模型二的模型分析。

第四个模型结果的正确性主要有以下保证：

- 1. 模型四模块一可以将视频按时间顺序译码成带时间标记的二进制数据流，可以自如的跳过任意帧数以及随时对数据流进行速写。即在进行视频截图时不会产生新的误差，可保证在截图时间点的精确性。
- 2. 视频拍摄的图像千差万别，并且可能存在多个物体有影子被拍摄，此处我们在模型四模块二加入人工干涉，选择正确的影子分析区域，保证后面模块得到数据的整洁性和完整性。
- 3. 模型四模块三以及模块四的敏感度阈值可以在模块二中进行调整，可适应多种多样的数据源。
- 4. 模型四模块五对视频录制时产生的面透视进行还原，修正纵深方向上的误差。

5.4.3 模型的建立与计算

算法设计：

- 模型四模块一：
  输入：视频文件
  输出：截取图像
  步骤：程序入口－ffmpeg解码－程序出口

- 模型四模块二：
  输入：配置文件
  输出：无
  步骤：程序入口－设置环境变量－应用配置－程序出口

- 模型四模块三：
  输入：截取图像
  输出：黑白两色图像
  步骤：程序入口－修改图像编码－循环降噪－扩大灰度差－程序出口

- 模型四模块四：
  输入：黑白两色图像
  输出：目标影子色块信息
  步骤：程序入口－基于广度优先搜索的洪水填充法区分出色块－获取各个色块特征数值－分析特征－程序出口

- 模型四模块五：
  输入：点阵图像，目标影子色块信息
  输出：影长及时间
  步骤：程序入口－OpenCV提取棋点－计算映射矩阵－矫正－程序出口

- 模型四模块六：
  输入：影长及时间
  输出：直杆所在地经度，直杆所在地纬度
  步骤：程序入口－数据收集－模型二经纬度计算－程序出口

算法实现： 如分析所述，视频格式为H264编码压制而成的avi，模块一设置好跳过时间即可正确运行。然而截好的图片太过于粗略，如图所示，地面上有多块阴影区域，需要进行人工干涉排除不需要的部分。

Figure 8: 原始截图

识别影子的轮廓，首先要简化问题，将裁剪图片化成灰度图像。

Figure 9: 裁剪后的截图

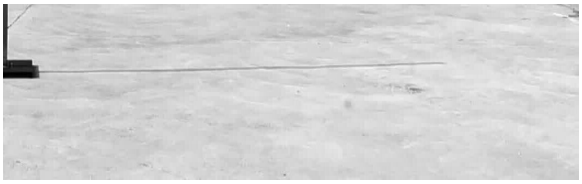

Figure 10: 灰度图像

显然，斑驳的地面无法满足后续模型的查找，需要扩大差异。于是通过降噪以及基于阈值的差分灰度算法，可以得到易于处理的黑白图像。

Figure 11: 黑白图像

区分色块简单高效的算法就是洪水填充，经过节省内存空间的广度优先搜索可以获得各个色块的信息。大概可以获得10^3 个色块，这里挑选最大的四个色块进行展示。

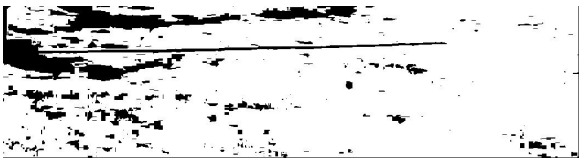

Figure 12: 色块一

Figure 13: 色块二

直杆阴影色块形状特殊，通过检查色块最左侧和最右侧轮廓以及色块横行长度可以识别出。通过自动化脚本，可以跑出一个时间－影长图像。

之后确定校正矩阵是一个关键。在投影校正技术中，基于相机的校正方法已经成为投影校正的基础，其基木原理就是通过相机来建立投影仪与相机之间的变换关系，再以这种变换关系来实现投影校正。而这种变换关系则通过几何模型变换来获得，这也是此处方法的技术基础，本部分将着重介绍相机投影模型，以及几何模型之间的变换关系。 相机几何模型表达了三维空间与二维图像之间

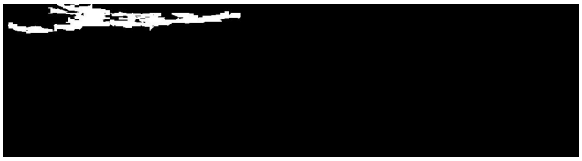

Figure 14: 色块三

Figure 15: 色块四

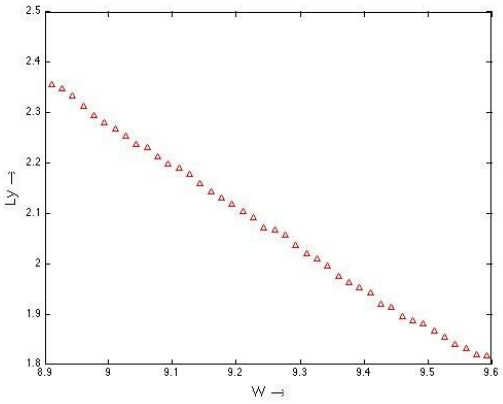

Figure 16: 模型四原始时间-影长图像

的映射变换关系，只要通过分析相机图像上的信息就可获得对三维空间的信息评测，接下来分析三维物体在相机几何模型中的几何表示与坐标转换方法。 对三维场景定性或定量分析都离不开参考坐标系[参考文献：一种连续，用于交通事故] 。描述环境中任何物体位置的坐标系为世界坐标系W 。世界坐标系$W(o_w, x_w, y_w, z_w)$, 相机坐标系$C(o_c, x_c, y_c, z_c)$, 图像坐标系$F(o_i, x, y)$, 象素坐标系I , 如图1所示。象素坐标系I 中, 每一象素坐标(u , v)分别是该象素在图像中的列数和行数。在这些坐标系中, 除了象素坐标是象素阵列的整数下标外, 其他坐标都是沿坐标轴的实数下标。

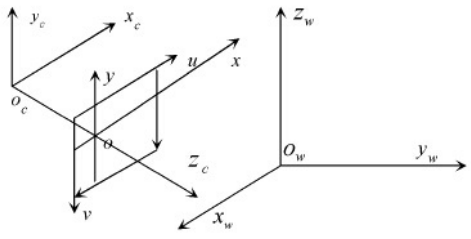

Figure 17: 坐标图

$$\lambda \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} \frac{1}{d_x} & 0 & u_0 \\ 0 & \frac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \tag{21}$$

$$\mathbf{M_1} = \begin{pmatrix} \frac{f}{d_x} & 0 & u_0 \\ 0 & \frac{f}{d_y} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{22}$$

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{23}$$

$$\mathbf{R}_\alpha = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \tag{24}$$

$$\mathbf{R}_\beta = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \tag{25}$$

$$\mathbf{R}_\gamma = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{26}$$

$$\mathbf{M_2} = \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{pmatrix} \tag{27}$$

$$\lambda \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = \mathbf{M_1 M_2} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} = \mathbf{M} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{28}$$

在建立了以上坐标系之后，使用基于OpenCV 的标定算法 确定相机参数的过程称为相机标定。基于图像的路面两点间距离的测量, 在理论上最终归结为求解世界平面到图像平面的对应关系矩阵M。采用平面棋盘模板基于OpenCV 的图像测距系统实现的关键就是得到矩阵M。如果已知n 组空间点坐标和图像点坐标数据, 则可以得到2n 个关于M 矩阵元素m ij 的线性方程。当$2n >> 11$ 时, 用最小二乘拟合的方法可以求出相机矩阵M 的11 个参数的值。点数n 至少是6 个, 最好是25 个或者更多。根据平面模板可以得到一组空间点坐标, 通过OpenCV 角点提取函数可以精确获得相应标定点图像坐标。考虑到图像角点提取失败会影响到标定结果以及采用最小二乘法进行标定需要多幅图像, 设计了如下标定算法[5-7] : (1)系统读取指定数量的图像, 并设置初始参数; (2)通过图像预处理, 对图像数据进行筛选。cvFind-ChessBoardCornerGuesses()读入图像数据, 如果图像上所有角点都被发现, 函数返回非零值, 并且将角点按一定顺序排列, 否则, 函数返回零。如果返回值为1 表示提取角点成功, 若为0 , 则表示该幅图像角点提取失败, 要舍弃该图像; (3)如果经过图像预处理后剩余图像数目大于某个预先设定值, 即满足标定需要, 则继续步骤(4), 否则返回步骤(1); (4)为标定点分配相应的内存空间, 由函数cvFind-ChessBoardCornerGuesses ()及函数cvFindCornerSubPix()得到子象素精度的角点图像坐标; (5)标定相机, 获得相机内外参数。将一组子象素精度的角点图像坐标和相应点在世界坐标系下的坐标值代入相机标定函数cvCalibrateCamera(),即得到标定结果。cvFindChessBoardCornerGuesses()的函数原型为: int cvFindChessBoardCornerGuesses(const CvArr * image , Cv Arr * thresh, Cv MemStor age * storage , Cv Size board size, Cv Point2D32f * corners , int * cornercount=NULL)

5.4.4 模型的建立与计算

经过上面求解过程中我们求得了唯一解，其坐标为：（41.0410 N，112.6320 E）
大致位于：

Figure 18: 地图

对该点影长模拟并进行误差分析，如下图所示：

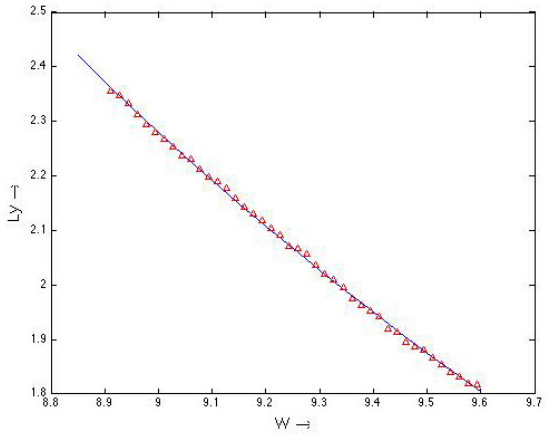

Figure 19: 附件4误差分析图

由图中可见，所求点的影长预测与给出点的影长误差很小。故所求点应该与给出点距离不大。

6 模型的评价与改进方向

6.1 模型的评价

6.1.1 模型一的评价

模型一中描述了赤纬角，太阳高度角，时差，经度修正时，太阳时等量和变量经度，纬度，日期，时区时之间的函数关系，使用了国家认可的修正公式，保证了模型的精确性。并且整个模型为后述模型打下了坚实的基础。

6.1.2 模型二的评价

在模型二中，解决方法采用遗传算法，充分发挥了遗传算法的与问题领域无关切快速随机的搜索能力，而且搜索从群体出发，具有潜在的并行性，可以进行多个个体的同时比较，稳定性较高。搜索使用评价函数启发，过程简单，并且使用概率机制进行迭代，具有随机性。体现出了遗传算法具有的良好的全局搜索能力，可以快速地将解空间中的全体解搜索出，而不会陷入局部最优解的快速下降陷阱；并且利用它的内在并行性，可以方便地进行分布式计算，加快求解速度。

以经纬度和杆长为基础，分区间搜索，以免漏掉局部近似最优解。同时对于结果本文进行了误差分析，以确保其准确性。

6.1.3 模型三的评价

模型三中，首先利用相关性分析确定了年份对于赤纬和时差的影响很小，因而忽略不计，将模型一中的变量变为杆长，经度，纬度和日期。同样利用遗传算法的长处，综合以杆长和时间划分的小区间，在区间内搜索可能点求解局部近似最优解，在保证解得完整性的同时，也确保了其准确性。

6.1.4 模型四的评价

模型四最开始应用自动化视频译码固定时间间隔图像保存模块将视频处理为40余帧照片，再利用人工干涉图像分析相关参数配置模和自动化图像像素编码转换以及基于阈值的灰度两极分化模块将照片转化为像素点。自动化基于洪水填充的色块分析以及智能影子色块信息查询输出模块把照片中的物体像素和影子像素自动提取出来。再利用自动化基于相机的校正方法的投影校正透视反向坐标系解析模块求得较为精确的数据，然后使用自动化数据收集以及与模型二的耦合及应用模块将模型四转化为前面模型进行求解。过程中误差小，自动化程度高，可推广应用至大部分阴影清晰的视频。

6.2 模型的改进和推广

6.2.1 模型的改进

本文模型中忽略了温度，压强对空气的影响而导致的各种折射，从而会导致结果有部分偏离。如果在有更大更多更全的数据支持的情况下，可以进行折射率的考虑从而对模型进行进一步的修正。

模型中的模型算法是在对时间的变化使用数值模拟法上推导的计算公式的基础上建立的，进一步补充时间数据也会提高模型的精度。

本文在搜索的过程中发现南半球的地点大都位于海上，因而将搜索的范围确定在北半球。

模型的求解方式是通过分区域后进行基于遗传算法的搜索，可以将区域进一步地缩小从而获得更多的可能近似解进行比对。遗传算法也可以考虑进一步的优化。并且求解方式可以考虑基于文献[1] 的约束搜索，作出可能的数条局部近似最优解的曲线后，利用图形法求得其交点即为近似最优解。

在模型四的投影校正中，可以采用准确的投影矩阵，考虑single view metrology见文献[2]再利用Validation of uncertainty analysis可以获得更准确的投影校正后的影子的长度

6.2.2 模型的推广

由于本文中所有的代码都是在通用的模型之上建立的，再加上模型中的高度自动化代码，我们可以方便的套入阴影清晰的不同的视频，直接轻松地获得所在地的经纬度，也可以实现和模型三的耦合而不是得到数据后再次带入模型三中，使得不知道日期的视频也可以得到定位和估计日期。

在投影校正的方面有很多种方法，如由矩形确定摄像机内参数与位置的线性方法，双线性映射矫正模型等，都可以纳入考虑范围，进行模型的推广。

6.2.3 总结

本文通过对于影长和经纬度，时间的分析建立了几何模型，并分区域利用遗传算法进行搜索求得局部近似最优解，而且在视频处理中考虑基于相机的校正矩阵的修正，可转化获取所需数据，并利用其求解。从而实现了太阳影子定位技术，实现了确定视频的拍摄地点和拍摄日期的目的。

模型利用《地面气象观测规范》中的气象观测公式，matlab自带的遗传算法ga函数，降噪和洪水填充处理视频图像以及OpenCV的标定算法，完成了整个模型的建立和搜索得解。并且验证了

《地面气象观测规范》中公式的准确性，实现了能够自动处理视频，进行修正并求解的高度自动的函数。

本文在搜索中集中目光于北半球，并别选择忽视折射率的影响，在投影矩阵的确定上也采取了模糊处理，使得整体的解存在一定的误差，可以利用更加精确的校正方式来解决图像以获得更高的精度。

7 参考文献

[1] Frode Eika Sandnes.Determining the Geographical Location of Image Scenes based on Object Shadow Lengths.Springer Science+Business Media, LLC .2010.

[2] Criminisi A, Reid I, Zisserman A. Single view metrology[J]. International Journal of Computer Vision,2000, 40(2): 123-148.2011.

[3] Hartley R, Zisserman A. Multiple view geometry in computer vision[M]. Cambridge university press, 2003.

[4] 倪育才. 空气折射率艾德林公式的修改[J]. 计量技术,: 22－27,1998,3.

[5] 韩燕, 强希文. 大气折射率高度分布模式及其应用[J]. 红外与 激光工程,267－271,2009, 2.

[6] 武琳,操晓春.基于太阳阴影轨迹的经纬度估计技术研究.天津大学.2010.

[7] 中国气象局. 地面气象观测规范[M]. 北京: 气象出版社，268-269. 2004.

[8] 何绪堂,用于交通事故处理的图像测距研究与实现,《现代电子技术》第24 期总第263 期,2007

[9] 潘玮璟,一种在线的连续投影校正方法,浙江大学,计算机应用技术,2011

附录一 通用代码

求影长代码

```
function Ly = shadow_length_calculator(Lg, th, D, Y, N, W)
        % core (reference our paper plz)
    L = D/15; % Long Correction
    dN = (W + L)/24; % Delta of time
    N0 = 79.6764 + 0.2422 * (Y - 1985) - floor(0.25 * (Y - 1985));
    Ct = W; % Standard Time
    Lc = ((-120)-D*180/pi)*4/60; % Longtitude Correction
    Tm = Ct + Lc; % Avg. real sun time

    Q = (2 * pi * 57.3 * (N + dN - N0) / 365.2422)/180*pi;
    Eq = (0.0028 - 1.9857 * sin(Q) + 9.9059 * sin(2 * Q) ...
      - 7.0924 * cos(Q) - 0.6882 * cos(2 * Q))/60; % Time equation

    TT = Tm + Eq; % Real sun time
    De = (0.3723 + 23.2567 * sin(Q) + 0.1149 * sin(2 * Q) ...
      - 0.1712 * sin(3 * Q) - 0.7580 * cos(Q) + 0.3656 * cos(2 * Q) ...
      + 0.0201 * cos(3 * Q))/180*pi; % Solar Declination

    t = ((TT - 12) * 15)/180*pi; % Solar Angle

    h = asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t)); % Solar height angle
    Ly = Lg / tan(h); % Length of shadow
End
```

求杆长代码

```
function Lg = stick_length_calculator(Ly, th, D, Y, N, W)
        % core (reference our paper plz)
```

```
    L = D/15; % Long Correction
    dN = (W + L)/24; % Delta of time
    N0 = 79.6764 + 0.2422 * (Y - 1985) - floor(0.25 * (Y - 1985));
    Ct = W; % Standard Time
    Lc = ((-120)-D*180/pi)*4/60; % Longtitude Correction
    Tm = Ct + Lc; % Avg. real sun time

    Q = (2 * pi * 57.3 * (N + dN - N0) / 365.2422)/180*pi;
    Eq = (0.0028 - 1.9857 * sin(Q) + 9.9059 * sin(2 * Q) ...
        - 7.0924 * cos(Q) - 0.6882 * cos(2 * Q))/60; % Time equation

    TT = Tm + Eq; % Real sun time
    De = (0.3723 + 23.2567 * sin(Q) + 0.1149 * sin(2 * Q) ...
        - 0.1712 * sin(3 * Q) - 0.7580 * cos(Q) + 0.3656 * cos(2 * Q) ...
        + 0.0201 * cos(3 * Q))/180*pi; % Solar Declination

    t = ((TT - 12) * 15)/180*pi; % Solar Angle

    h = asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t)); % Solar height angle
    Lg = Ly * tan(h); % Length of stick
end
```

附录二 问题一代码

问题一代码一

```
clear;
syms W;
% Question 1 solution (direct computation)

% W = 9;
```

```
th = (39 + 54/60 + 26/3600)/180*pi; % Lat
D = -(116 + 23/60 + 29/3600)/180*pi; % Long
Y = 2015; % Year
N = 294; % DayOfYear
Lg = 3;

Ly = shadow_length_calculator(Lg, th, D, Y, N, W);
h = ezplot(Ly, 9:15);
Ly_fun = @(x) double(subs(Ly, x));
x_min = fminunc(Ly_fun, 9, 15);
hold on
plot(x_min, Ly_fun(x_min), 'ro');
text(12, 4, sprintf('\\fontsize{12}W=%f', x_min));
text(12, 3.8, sprintf('\\fontsize{12}Ly=%f', Ly_fun(x_min)));
xlabel('\fontsize{14} W \rightarrow');
ylabel('\fontsize{14} Ly \rightarrow');
title('')
```

问题一代码二

```
clear;
syms W;
% Question 1 attempt (angle unit mismatch) (spend: 4 hours)

%W = 12;
th = (39 + 54/60 + 26/3600)/180*pi; % Lat
D = -(116 + 23/60 + 29/3600)/180*pi; % Long
Y = 2015; % Year
N = 294; % DayOfYear
Lg = 3; % Length of the stick

De = -0.4092797 * cos(2*pi/365*(N+10)); % Solar Declination
```

```
TT = (W-8) - 12/pi * D; % Real sun time
t = ((TT - 12) * 15)/180*pi; % Solar Angle

% atan(Lg/Ly) == asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t)); % Solar height angle

h = asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t));

Ly = Lg / tan(h); % Length of shadow

disp(Ly);
ezplot(Ly, 9:15);
```

附录三 问题二代码

问题二代码一

```
clear;
% Question 2 attempt (direct solve equation) (answers too random and too inaccurate)

% [ float(row.split(',')[0].split(':')[0])+float(row.split(',')[0].split(':')[1])/60 for row in
open('data/appendix_1.csv').read().split('\r\n')[3:] ]
time = [14.7, 14.75, 14.8, 14.85, 14.9, 14.95, 15.0, 15.05, 15.1, 15.15, 15.2, 15.25, 15.3, 15.35,
15.4, 15.45, 15.5, 15.55, 15.6, 15.65, 15.7];
% [ float(row.split(',')[1]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
x = [1.0365, 1.0699, 1.1038, 1.1383, 1.1732, 1.2087, 1.2448, 1.2815, 1.3189, 1.3568, 1.3955,
1.4349, 1.4751, 1.516, 1.5577, 1.6003, 1.6438, 1.6882, 1.7337, 1.7801, 1.8277];
% [ float(row.split(',')[2]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
y = [0.4973, 0.5029, 0.5085, 0.5142, 0.5198, 0.5255, 0.5311, 0.5368, 0.5426, 0.5483, 0.5541,
0.5598, 0.5657, 0.5715, 0.5774, 0.5833, 0.5892, 0.5952, 0.6013, 0.6074, 0.6135];
% [ math.sqrt(x[i]**2 + y[i]**2) for i in xrange(0,21) ]
```

```
LengthOfShadow = [1.149625826084296, 1.1821989764840775, 1.215296955480429,
1.2490510517989248, 1.2831953397670988, 1.3179931486923595, 1.3533640493230192,
1.389387091490345, 1.4261528564638504, 1.4633998530818568, 1.5014816215991456,
1.5402318169678226, 1.5798533159758852, 1.6201445151590645, 1.6612706131151542,
1.7032906328633408, 1.7462059099659466, 1.790050915476987, 1.8350142724240595,
1.8808750011630226, 1.927918447445327];

Y = 2015; % Year
N = 107; % DayOfYear
syms th D Lg;

chosen = [9, 15, 20];

for i = chosen
    W = time(i);
    Ly = LengthOfShadow(i);
    L = D/15; % Longtitude Correction
    dN = (W + L)/24; % Delta of time
    N0 = 79.6764 + 0.2422 * (Y - 1985) - floor(0.25 * (Y - 1985));
    Ct = W; % Standard Time
    Lc = ((-120)-D*180/pi)*4/60; % Longtitude Correction
    Tm = Ct + Lc; % Avg. real sun time

    Q = (2 * pi * 57.3 * (N + dN - N0) / 365.2422)/180*pi;
    Eq = (0.0028 - 1.9857 * sin(Q) + 9.9059 * sin(2 * Q) ...
        - 7.0924 * cos(Q) - 0.6882 * cos(2 * Q))/60; % Time equation

    TT = Tm + Eq; % Real sun time
    De = (0.3723 + 23.2567 * sin(Q) + 0.1149 * sin(2 * Q) ...
        - 0.1712 * sin(3 * Q) - 0.7580 * cos(Q) + 0.3656 * cos(2 * Q) ...
        + 0.0201 * cos(3 * Q))/180*pi; % Solar Declination
```

```
    t = ((TT - 12) * 15)/180*pi; % Solar Angle

    % atan(Lg/Ly) == asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t)); % Solar height angle

    if i == chosen(1)
        eq1 = atan(Lg/Ly) == abs(asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t)));
    elseif i == chosen(2)
        eq2 = atan(Lg/Ly) == abs(asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t)));
    elseif i == chosen(3)
        eq3 = atan(Lg/Ly) == abs(asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t)));
    end
end

sol = solve(eq1, eq2, eq3);

D = subs(sol.D);
th = subs(sol.th);
Lg = subs(sol.Lg);
disp(double(D*180/pi));
disp(double(th*180/pi));
disp(Lg);

L = D/15; % Longtitude Correction
dN = (W + L)/24; % Delta of time
N0 = 79.6764 + 0.2422 * (Y - 1985) - floor(0.25 * (Y - 1985));
Ct = W; % Standard Time
Lc = ((-120)-D*180/pi)*4/60; % Longtitude Correction
Tm = Ct + Lc; % Avg. real sun time

Q = (2 * pi * 57.3 * (N + dN - N0) / 365.2422)/180*pi;
```

```
Eq = (0.0028 - 1.9857 * sin(Q) + 9.9059 * sin(2 * Q) ...
    - 7.0924 * cos(Q) - 0.6882 * cos(2 * Q))/60; % Time equation

TT = Tm + Eq; % Real sun time
De = (0.3723 + 23.2567 * sin(Q) + 0.1149 * sin(2 * Q) ...
    - 0.1712 * sin(3 * Q) - 0.7580 * cos(Q) + 0.3656 * cos(2 * Q) ...
    + 0.0201 * cos(3 * Q))/180*pi; % Solar Declination

t = ((TT - 12) * 15)/180*pi; % Solar Angle

h = asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t)); % Solar height angle
Ly = Lg / tan(h);
```

问题二代码二

```
clear;
% Question 2 attempt (polyfit the curve before solve equation) (found no improvement)

% [ float(row.split(',')[0].split(':')[0])+float(row.split(',')[0].split(':')[1])/60 for row in
open('data/appendix_1.csv').read().split('\r\n')[3:] ]
time = [12, 12.1, 12.2];
% [ float(row.split(',')[1]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
% x = [1.0365, 1.0699, 1.1038, 1.1383, 1.1732, 1.2087, 1.2448, 1.2815, 1.3189, 1.3568, 1.3955,
1.4349, 1.4751, 1.516, 1.5577, 1.6003, 1.6438, 1.6882, 1.7337, 1.7801, 1.8277];
% [ float(row.split(',')[2]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
% y = [0.4973, 0.5029, 0.5085, 0.5142, 0.5198, 0.5255, 0.5311, 0.5368, 0.5426, 0.5483, 0.5541,
0.5598, 0.5657, 0.5715, 0.5774, 0.5833, 0.5892, 0.5952, 0.6013, 0.6074, 0.6135];
% [ math.sqrt(x[i]**2 + y[i]**2) for i in xrange(0,21) ]
% LengthOfShadow = [];
% for i = 1:20
%     LengthOfShadow(end + 1) = sqrt(x(i)*x(i) + y(i)*y(i)); %#ok<SAGROW>
% end
```

```
LengthOfShadow = [3.6517, 3.6554, 3.6554];

Y = 2015; % Year
N = 294; % DayOfYear
syms Lg D;

% coefs = polyfit(time, LengthOfShadow, 2);
% parabola = @(x) coefs(1) * x * x + coefs(2) * x + coefs(3);

th = (39 + 54/60 + 26/3600)/180*pi; % Lat
% D = -(116 + 23/60 + 29/3600)/180*pi; % Long
chosen = [2, 3];

W = time(chosen(1));
Ly = LengthOfShadow(chosen(1));
De = -0.4092797 * cos(2 * pi / 365 * (N + 10)); % Solar Declination
TT = (W - 8) - 12/pi * D; % Real sun time
t = ((TT - 12) * 15)/180*pi; % Solar Angle
h = asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t));
cond1 = h >= 0;
cond2 = h <= pi/2;
eq1 = Lg/Ly == tan(h);

W = time(chosen(2));
Ly = LengthOfShadow(chosen(2));
De = -0.4092797 * cos(2 * pi / 365 * (N + 10)); % Solar Declination
TT = (W - 8) - 12/pi * D; % Real sun time
t = ((TT - 12) * 15)/180*pi; % Solar Angle
h = asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t));
cond3 = h >= 0;
cond4 = h <= pi/2;
```

```
eq2 = Lg/Ly == tan(h);

sol = solve(eq1, eq2, cond1, cond2, cond3, cond4);

% D = subs(sol.D);
% th = subs(sol.th);
% Lg = subs(sol.D);
% disp(double(D));
% disp(double(th));
disp(sol.D);
```

问题二代码三

```
clear;
% Question 2 attempt (manual simplify the equation) (found no improvement)

% [ float(row.split(',')[0].split(':')[0])+float(row.split(',')[0].split(':')[1])/60 for row in
open('data/appendix_1.csv').read().split('\r\n')[3:] ]
time = [14.7, 14.75, 14.8, 14.85, 14.9, 14.95, 15.0, 15.05, 15.1, 15.15, 15.2, 15.25, 15.3, 15.35,
15.4, 15.45, 15.5, 15.55, 15.6, 15.65, 15.7];
% [ float(row.split(',')[1]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
x = [1.0365, 1.0699, 1.1038, 1.1383, 1.1732, 1.2087, 1.2448, 1.2815, 1.3189, 1.3568, 1.3955,
1.4349, 1.4751, 1.516, 1.5577, 1.6003, 1.6438, 1.6882, 1.7337, 1.7801, 1.8277];
% [ float(row.split(',')[2]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
y = [0.4973, 0.5029, 0.5085, 0.5142, 0.5198, 0.5255, 0.5311, 0.5368, 0.5426, 0.5483, 0.5541,
0.5598, 0.5657, 0.5715, 0.5774, 0.5833, 0.5892, 0.5952, 0.6013, 0.6074, 0.6135];
% [ math.sqrt(x[i]**2 + y[i]**2) for i in xrange(0,21) ]
LengthOfShadow = [1.149625826084296, 1.1821989764840775, 1.215296955480429,
1.2490510517989248, 1.2831953397670988, 1.3179931486923595, 1.3533640493230192,
1.389387091490345, 1.4261528564638504, 1.4633998530818568, 1.5014816215991456,
1.5402318169678226, 1.5798533159758852, 1.6201445151590645, 1.6612706131151542,
```

```
1.7032906328633408, 1.7462059099659466, 1.790050915476987, 1.8350142724240595,
1.8808750011630226, 1.927918447445327];

Y = 2015; % Year
N = 107; % DayOfYear
syms th D Lg;

chosen = [9, 15, 20];

for i = chosen
    W = time(i);
    Ly = LengthOfShadow(i);
    De = -0.4092797 * cos(2*pi/365*(N+10)); % Solar Declination
    TT = (W-8) - 12/pi * D; % Real sun time
    t = ((TT - 12) * 15)/180*pi; % Solar Angle

    % atan(Lg/Ly) == asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t)); % Solar height angle

    if i == chosen(1)
        eq1 = atan(Lg/Ly) == asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t));
    elseif i == chosen(2)
        eq2 = atan(Lg/Ly) == asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t));
    elseif i == chosen(3)
        eq3 = atan(Lg/Ly) == asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t));
    end
end

sol = solve(eq1, eq2, eq3);

D = subs(sol.D);
th = subs(sol.th);
```

```
Lg = subs(sol.Lg);
disp(double(D*180/pi));
disp(double(th*180/pi));
disp(Lg);

L = D/15; % Longtitude Correction
dN = (W + L)/24; % Delta of time
N0 = 79.6764 + 0.2422 * (Y - 1985) - floor(0.25 * (Y - 1985));
Ct = W; % Standard Time
Lc = ((-120)-D*180/pi)*4/60; % Longtitude Correction
Tm = Ct + Lc; % Avg. real sun time

Q = (2 * pi * 57.3 * (N + dN - N0) / 365.2422)/180*pi;
Eq = (0.0028 - 1.9857 * sin(Q) + 9.9059 * sin(2 * Q) ...
    - 7.0924 * cos(Q) - 0.6882 * cos(2 * Q))/60; % Time equation

TT = Tm + Eq; % Real sun time
De = (0.3723 + 23.2567 * sin(Q) + 0.1149 * sin(2 * Q) ...
    - 0.1712 * sin(3 * Q) - 0.7580 * cos(Q) + 0.3656 * cos(2 * Q) ...
    + 0.0201 * cos(3 * Q))/180*pi; % Solar Declination

t = ((TT - 12) * 15)/180*pi; % Solar Angle

h = asin(sin(th)*sin(De) + cos(th)*cos(De)*cos(t)); % Solar height angle
Ly = Lg / tan(h);
```

问题二代码四

```
clear;
% Question 2 attempt (longitude and latitude and stick length enumeration) (solution
covers too much area)
```

```
% [ float(row.split(',')[0].split(':')[0])+float(row.split(',')[0].split(':')[1])/60 for row in
open('data/appendix_1.csv').read().split('\r\n')[3:] ]
time = [14.7, 14.75, 14.8, 14.85, 14.9, 14.95, 15.0, 15.05, 15.1, 15.15, 15.2, 15.25, 15.3, 15.35,
15.4, 15.45, 15.5, 15.55, 15.6, 15.65, 15.7];
% [ float(row.split(',')[1]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
x = [1.0365, 1.0699, 1.1038, 1.1383, 1.1732, 1.2087, 1.2448, 1.2815, 1.3189, 1.3568, 1.3955,
1.4349, 1.4751, 1.516, 1.5577, 1.6003, 1.6438, 1.6882, 1.7337, 1.7801, 1.8277];
% [ float(row.split(',')[2]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
y = [0.4973, 0.5029, 0.5085, 0.5142, 0.5198, 0.5255, 0.5311, 0.5368, 0.5426, 0.5483, 0.5541,
0.5598, 0.5657, 0.5715, 0.5774, 0.5833, 0.5892, 0.5952, 0.6013, 0.6074, 0.6135];
% [ math.sqrt(x[i]**2 + y[i]**2) for i in xrange(0,21) ]
LengthOfShadow = [1.149625826084296, 1.1821989764840775, 1.215296955480429,
1.2490510517989248, 1.2831953397670988, 1.3179931486923595, 1.3533640493230192,
1.389387091490345, 1.4261528564638504, 1.4633998530818568, 1.5014816215991456,
1.5402318169678226, 1.5798533159758852, 1.6201445151590645, 1.6612706131151542,
1.7032906328633408, 1.7462059099659466, 1.790050915476987, 1.8350142724240595,
1.8808750011630226, 1.927918447445327];

Y = 2015;
N = 107;

syms Lg

esp = 1e-1;
th_arr = [];
D_arr = [];
for D = -pi:0.05:pi
    for th = -pi/2:0.05:pi/2
        for Lg = 0.5:0.1:3
            dif = 0;
```

```
        for i = 1:20
            Ly = shadow_length_calculator(Lg, th, D, Y, N, time(i));
            dif = dif + abs(Ly-LengthOfShadow(i));
            if dif > esp
                break;
            end
        end
        if dif < esp
            th_arr(end + 1) = th * 180/pi;
            D_arr(end + 1) = D * 180/pi;
            disp('----------')
            disp(th*180/pi)
            disp(D*180/pi)
            disp(Lg)
        end
    end
  end
end

scatter(D_arr, th_arr)
```

问题二代码五

```
clear;
% Question 2 attempt (abandon shadow length, calc stick length instead) (solution covers too much area) (in weird shape)

% [ float(row.split(',')[0].split(':')[0])+float(row.split(',')[0].split(':')[1])/60 for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
```

```
time = [14.7, 14.75, 14.8, 14.85, 14.9, 14.95, 15.0, 15.05, 15.1, 15.15, 15.2, 15.25, 15.3, 15.35,
15.4, 15.45, 15.5, 15.55, 15.6, 15.65, 15.7];
% [ float(row.split(',')[1]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
x = [1.0365, 1.0699, 1.1038, 1.1383, 1.1732, 1.2087, 1.2448, 1.2815, 1.3189, 1.3568, 1.3955,
1.4349, 1.4751, 1.516, 1.5577, 1.6003, 1.6438, 1.6882, 1.7337, 1.7801, 1.8277];
% [ float(row.split(',')[2]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
y = [0.4973, 0.5029, 0.5085, 0.5142, 0.5198, 0.5255, 0.5311, 0.5368, 0.5426, 0.5483, 0.5541,
0.5598, 0.5657, 0.5715, 0.5774, 0.5833, 0.5892, 0.5952, 0.6013, 0.6074, 0.6135];
% [ math.sqrt(x[i]**2 + y[i]**2) for i in xrange(0,21) ]
LengthOfShadow = [1.149625826084296, 1.1821989764840775, 1.215296955480429,
1.2490510517989248, 1.2831953397670988, 1.3179931486923595, 1.3533640493230192,
1.389387091490345, 1.4261528564638504, 1.4633998530818568, 1.5014816215991456,
1.5402318169678226, 1.5798533159758852, 1.6201445151590645, 1.6612706131151542,
1.7032906328633408, 1.7462059099659466, 1.790050915476987, 1.8350142724240595,
1.8808750011630226, 1.927918447445327];

Y = 2015;
N = 107;

syms th D

esp = 1e-2;
th_arr = [];
D_arr = [];

for D = -pi:0.05:pi
    for th = -pi/2:0.05:pi/2
        Lg_arr = [];
        for i = 1:20
            Lg_arr(end + 1) = stick_length_calculator(LengthOfShadow(i),...
                th, D, Y, N, time(i));
```

```
        end
        if mean(Lg_arr)>0.78 && mean(Lg_arr)<1.92 && std(Lg_arr) < esp
            th_arr(end + 1) = th * 180/pi;
            D_arr(end + 1) = D * 180/pi;
            disp('----------')
            disp(th*180/pi)
            disp(D*180/pi)
            disp(mean(Lg_arr))
            disp(std(Lg_arr))
        end
    end
end

scatter(D_arr, th_arr)
[mean(th_arr), mean(D_arr)]
```

问题二代码六

```
clear;
% Question 2 attempt (polyfit the shadow length curve to find the longitude first, then
brutal force the latitude) (found Guangxi as an solution)

% [ float(row.split(',')[0].split(':')[0])+float(row.split(',')[0].split(':')[1])/60 for row in
open('data/appendix_1.csv').read().split('\r\n')[3:] ]
time = [14.7, 14.75, 14.8, 14.85, 14.9, 14.95, 15.0, 15.05, 15.1, 15.15, 15.2, 15.25, 15.3, 15.35,
15.4, 15.45, 15.5, 15.55, 15.6, 15.65, 15.7];
% [ float(row.split(',')[1]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
x = [1.0365, 1.0699, 1.1038, 1.1383, 1.1732, 1.2087, 1.2448, 1.2815, 1.3189, 1.3568, 1.3955,
1.4349, 1.4751, 1.516, 1.5577, 1.6003, 1.6438, 1.6882, 1.7337, 1.7801, 1.8277];
% [ float(row.split(',')[2]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
```

```
y = [0.4973, 0.5029, 0.5085, 0.5142, 0.5198, 0.5255, 0.5311, 0.5368, 0.5426, 0.5483, 0.5541,
0.5598, 0.5657, 0.5715, 0.5774, 0.5833, 0.5892, 0.5952, 0.6013, 0.6074, 0.6135];
% [ math.sqrt(x[i]**2 + y[i]**2) for i in xrange(0,21) ]
LengthOfShadow = [1.149625826084296, 1.1821989764840775, 1.215296955480429,
1.2490510517989248, 1.2831953397670988, 1.3179931486923595, 1.3533640493230192,
1.389387091490345, 1.4261528564638504, 1.4633998530818568, 1.5014816215991456,
1.5402318169678226, 1.5798533159758852, 1.6201445151590645, 1.6612706131151542,
1.7032906328633408, 1.7462059099659466, 1.790050915476987, 1.8350142724240595,
1.8808750011630226, 1.927918447445327];

syms x_min;
coefs = polyfit(time, LengthOfShadow, 2);
parabola =coefs(1) * x_min * x_min + coefs(2) * x_min + coefs(3);

% ezplot(parabola, 12:16)
dy = diff(parabola);
x_min = solve(dy);
W = double(x_min);

syms D
Y = 2015;
N = 107;
TT = 12;
L = D/15; % Long Correction
dN = (W + L)/24; % Delta of time
N0 = 79.6764 + 0.2422 * (Y - 1985) - floor(0.25 * (Y - 1985));
Ct = W; % Standard Time
Lc = ((-120)-D*180/pi)*4/60; % Longtitude Correction
Tm = Ct + Lc; % Avg. real sun time
Q = (2 * pi * 57.3 * (N + dN - N0) / 365.2422)/180*pi;
Eq = (0.0028 - 1.9857 * sin(Q) + 9.9059 * sin(2 * Q) ...
```

```
    - 7.0924 * cos(Q) - 0.6882 * cos(2 * Q))/60; % Time equation
% ezplot(Tm+Eq-12, [-105/180*pi, -120/180*pi])
sol = solve(12 == Tm + Eq, D);
D = double(subs(sol))*180/pi;


esp = 1e2;
th_arr = [];
for th = -90/180*pi:0.01:90/180*pi
    for Lg = 0.7:1.9
        dif = 0;
        for i = 1:20
            Ly = shadow_length_calculator(Lg, th, D, Y, N, time(i));
            dif = dif + abs(Ly - LengthOfShadow(i));
        end
        if dif < esp
            th_arr(end + 1) = th * 180/pi;
            disp('----------')
            disp(th*180/pi)
            disp(Lg)
        end
    end
end
th_arr
plot(x,y)
% hist(th_arr)
```

问题二代码七

```
clear;
% Question 2 attempt (got the longitude, polyfit the [x, y]) (found polyfit realy inaccurative)
```

```
% [ float(row.split(',')[0].split(':')[0])+float(row.split(',')[0].split(':')[1])/60 for row in
open('data/appendix_1.csv').read().split('\r\n')[3:] ]
time = [14.7, 14.75, 14.8, 14.85, 14.9, 14.95, 15.0, 15.05, 15.1, 15.15, 15.2, 15.25, 15.3, 15.35,
15.4, 15.45, 15.5, 15.55, 15.6, 15.65, 15.7];
% [ float(row.split(',')[1]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
x = [1.0365, 1.0699, 1.1038, 1.1383, 1.1732, 1.2087, 1.2448, 1.2815, 1.3189, 1.3568, 1.3955,
1.4349, 1.4751, 1.516, 1.5577, 1.6003, 1.6438, 1.6882, 1.7337, 1.7801, 1.8277];
% [ float(row.split(',')[2]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
y = [0.4973, 0.5029, 0.5085, 0.5142, 0.5198, 0.5255, 0.5311, 0.5368, 0.5426, 0.5483, 0.5541,
0.5598, 0.5657, 0.5715, 0.5774, 0.5833, 0.5892, 0.5952, 0.6013, 0.6074, 0.6135];
% [ math.sqrt(x[i]**2 + y[i]**2) for i in xrange(0,21) ]
LengthOfShadow = [1.149625826084296, 1.1821989764840775, 1.215296955480429,
1.2490510517989248, 1.2831953397670988, 1.3179931486923595, 1.3533640493230192,
1.389387091490345, 1.4261528564638504, 1.4633998530818568, 1.5014816215991456,
1.5402318169678226, 1.5798533159758852, 1.6201445151590645, 1.6612706131151542,
1.7032906328633408, 1.7462059099659466, 1.790050915476987, 1.8350142724240595,
1.8808750011630226, 1.927918447445327];

syms x_min;
coefs = polyfit(time, LengthOfShadow, 2);
parabola =coefs(1) * x_min * x_min + coefs(2) * x_min + coefs(3);

% ezplot(parabola, 12:16)
dy = diff(parabola);
x_min = solve(dy);
W = double(x_min);

syms D
Y = 2015;
N = 107;
TT = 12;
```

```
L = D/15; % Long Correction
dN = (W + L)/24; % Delta of time
N0 = 79.6764 + 0.2422 * (Y - 1985) - floor(0.25 * (Y - 1985));
Ct = W; % Standard Time
Lc = ((-120)-D*180/pi)*4/60; % Longtitude Correction
Tm = Ct + Lc; % Avg. real sun time
Q = (2 * pi * 57.3 * (N + dN - N0) / 365.2422)/180*pi;
Eq = (0.0028 - 1.9857 * sin(Q) + 9.9059 * sin(2 * Q) ...
    - 7.0924 * cos(Q) - 0.6882 * cos(2 * Q))/60; % Time equation
% ezplot(Tm+Eq-12, [-105/180*pi, -120/180*pi])
sol = solve(12 == Tm + Eq, D);
D = double(subs(sol))*180/pi;
D


syms x_min
coefs = polyfit(x, y, 2);
parabola =coefs(1) * x_min * x_min + coefs(2) * x_min + coefs(3);


% plot(x, y)
ezplot(parabola, -5:5)
dy = diff(parabola);
x_min = solve(dy);
W = double(x_min);
```

问题二代码八遗传算法（正解）

```
clear, clc;
% Question 2 solution (use ga to calc solution, and divide into several part by the length of stick) (which greatly influence the longitude)

A = [[-1,0,0,0,0,0],
[1,0,0,0,0,0],
```

```
[0,-1,0,0,0,0],
[0,1,0,0,0,0],
[0,0,-1,0,0,0],
[0,0,1,0,0,0]];

out=fopen('t2.txt','w');

for Lg_min = 0:1:3
    dif = 1e8;
    lat = 0.;
    long = 0.;
    Lg = 0.;
    for i = 1:4
        b = [pi/2, pi/2, pi, pi, -Lg_min, Lg_min+1];
        x = ga(@t2_ga_fun, 6, A, b);
        if t2_ga_fun(x) < dif
            dif = t2_ga_fun(x);
            lat = x(1);
            long = x(2);
            Lg = x(3);
        end
        disp([lat, long, Lg, Lg_min, dif]);
    end
    disp([lat, long, Lg, 'CHOSEN FOR ', Lg_min]);
    fprintf(out, '%f\r%f\r%f\n', lat, long, Lg);
end
```

问题二遗传算法优化函数

```
function dif = t2_ga_fun(X)
        % Question 2 tool (ga optimization function)
```

```
    time = [14.7, 14.75, 14.8, 14.85, 14.9, 14.95, 15.0, 15.05, 15.1, 15.15, 15.2, 15.25, 15.3, 15.35,
15.4, 15.45, 15.5, 15.55, 15.6, 15.65, 15.7];
    LengthOfShadow = [1.149625826084296, 1.1821989764840775, 1.215296955480429,
1.2490510517989248, 1.2831953397670988, 1.3179931486923595, 1.3533640493230192,
1.389387091490345, 1.4261528564638504, 1.4633998530818568, 1.5014816215991456,
1.5402318169678226, 1.5798533159758852, 1.6201445151590645, 1.6612706131151542,
1.7032906328633408, 1.7462059099659466, 1.790050915476987, 1.8350142724240595,
1.8808750011630226, 1.927918447445327];

    lat = X(1);
    long = X(2);
    Lg = X(3);

    Y = 2015;
    N = 107;

    dif = 0;
    for i = 1:20
        Ly = shadow_length_calculator(Lg, lat, long, ...
            Y, N, time(i));
        dif = dif + (Ly - LengthOfShadow(i)) * (Ly - LengthOfShadow(i));
    end
end
```

附录四 问题三代码

问题三拟合代码

```
clear, clc;

time = [12.683333333333334, 12.733333333333333, 12.783333333333333,
12.833333333333334, 12.883333333333333, 12.933333333333334, 12.983333333333333,
13.033333333333333, 13.083333333333334, 13.133333333333333, 13.183333333333334,
13.233333333333333, 13.283333333333333, 13.333333333333334, 13.383333333333333,
13.433333333333334, 13.483333333333333, 13.533333333333333, 13.583333333333334,
13.633333333333333, 13.683333333333334];
LengthOfShadow = [1.2472562046347977, 1.2227945902726263, 1.198921486169966,
1.1754289642509241, 1.1524395732531925, 1.1299174704375536, 1.1078354796629324,
1.0862542059757467, 1.0650810720316084, 1.0444462647738273, 1.024264126092484,
1.0046403137441777, 0.9854909081265032, 0.9667904943678335, 0.9485847352767173,
0.9309278812024055, 0.9137517496563277, 0.897109051342143, 0.8809737623788804,
0.8654922587753169, 0.8505044679482877];

syms x_min;
coefs = polyfit(time, LengthOfShadow, 2);
parabola =coefs(1) * x_min * x_min + coefs(2) * x_min + coefs(3);

ezplot(parabola, 12:16)
dy = diff(parabola);
x_min = solve(dy);
W = double(x_min);

syms D
Y = 2015;
```

```
N = 107;
TT = 12;
L = D/15; % Long Correction
dN = (W + L)/24; % Delta of time
N0 = 79.6764 + 0.2422 * (Y - 1985) - floor(0.25 * (Y - 1985));
Ct = W; % Standard Time
Lc = ((-120)-D*180/pi)*4/60; % Longtitude Correction
Tm = Ct + Lc; % Avg. real sun time
Q = (2 * pi * 57.3 * (N + dN - N0) / 365.2422)/180*pi;
Eq = (0.0028 - 1.9857 * sin(Q) + 9.9059 * sin(2 * Q) ...
    - 7.0924 * cos(Q) - 0.6882 * cos(2 * Q))/60; % Time equation
% ezplot(Tm+Eq-12, [-105/180*pi, -120/180*pi])
sol = solve(12 == Tm + Eq, D);
D = double(subs(sol))*180/pi;
D
% -71.7715
```

问题三第一问遗传算法（正解）

```
clear, clc;

A = [[-1,0,0,0,0,0,0,0],
[1,0,0,0,0,0,0,0],
[0,-1,0,0,0,0,0,0],
[0,1,0,0,0,0,0,0],
[0,0,-1,0,0,0,0,0],
[0,0,1,0,0,0,0,0],
[0,0,0,-1,0,0,0,0],
[0,0,0,1,0,0,0,0]];

% Data for 2015
```

```
season_begin_date = [0, 91, 182, 274, 365]

out=fopen('t3_1.txt','w');

for season = 1:4
        for Lg_min = 0:1:3
           dif = 1e8;
           lat = 0.;
           long = 0.;
           Lg = 0.;
           N = 0.;
           for i = 1:4
              b = [pi/2, pi/2, pi, pi, -Lg_min, Lg_min+1, -season_begin_date(season),
season_begin_date(season+1)];
              x = ga(@t3_1_ga_fun, 8, A, b);
              if t3_1_ga_fun(x) < dif
                 dif = t3_1_ga_fun(x);
                 lat = x(1);
                 long = x(2);
                 Lg = x(3);
                 N = x(4);
              end
              disp([lat, long, Lg, N, season, Lg_min, dif]);
           end
           disp([lat, long, Lg, N, 'CHOSEN FOR ', Lg_min]);
           fprintf(out, '%f\r%f\r%f\r%f\r%d\r%d\n', lat, long, Lg, N, season, Lg_min);
        end
end
```

问题三第一问遗传算法优化函数

```
function dif = t3_1_ga_fun(X)
        % Question 3 tool (ga optimization function)
   time = [12.683333333333334, 12.733333333333333, 12.783333333333333,
12.833333333333334, 12.883333333333333, 12.933333333333334, 12.983333333333333,
13.033333333333333, 13.083333333333334, 13.133333333333333, 13.183333333333334,
13.233333333333333, 13.283333333333333, 13.333333333333334, 13.383333333333333,
13.433333333333334, 13.483333333333333, 13.533333333333333, 13.583333333333334,
13.633333333333333, 13.683333333333334];
   LengthOfShadow = [1.2472562046347977, 1.2227945902726263, 1.198921486169966,
1.1754289642509241, 1.1524395732531925, 1.1299174704375536, 1.1078354796629324,
1.0862542059757467, 1.0650810720316084, 1.0444462647738273, 1.024264126092484,
1.0046403137441777, 0.9854909081265032, 0.9667904943678335, 0.9485847352767173,
0.9309278812024055, 0.9137517496563277, 0.897109051342143, 0.8809737623788804,
0.8654922587753169, 0.8505044679482877];
   lat = X(1);
   long = X(2);
   Lg = X(3);
   N = X(4);

   Y = 2015;

   dif = 0;
   for i = 1:20
      Ly = shadow_length_calculator(Lg, lat, long, ...
         Y, N, time(i));
      dif = dif + (Ly - LengthOfShadow(i)) * (Ly - LengthOfShadow(i));
   end
end
```

问题三第二问遗传算法（正解）

```
clear, clc;

A = [[-1,0,0,0,0,0,0,0],
[1,0,0,0,0,0,0,0],
[0,-1,0,0,0,0,0,0],
[0,1,0,0,0,0,0,0],
[0,0,-1,0,0,0,0,0],
[0,0,1,0,0,0,0,0],
[0,0,0,-1,0,0,0,0],
[0,0,0,1,0,0,0,0]];

% Data for 2015
season_begin_date = [0, 91, 182, 274, 365]

out=fopen('t3_2.txt','w');

for season = 1:4
        for Lg_min = 0:1.5:6
            dif = 1e8;
            lat = 0.;
            long = 0.;
            Lg = 0.;
            N = 0.;
            for i = 1:4
                b = [pi/2, pi/2, pi, pi, -Lg_min, Lg_min+1.5, -season_begin_date(season),
season_begin_date(season+1)];
                x = ga(@t3_2_ga_fun, 8, A, b);
                if t3_2_ga_fun(x) < dif
```

```
                dif = t3_2_ga_fun(x);
                lat = x(1);
                long = x(2);
                Lg = x(3);
                N = x(4);
            end
            disp([lat, long, Lg, N, season, Lg_min, dif]);
        end
        disp([lat, long, Lg, N, 'CHOSEN FOR ', Lg_min]);
        fprintf(out, '%f\r%f\r%f\r%f\r%d\r%d\n', lat, long, Lg, N, season, Lg_min);
    end
end
```

问题三第二问遗传算法优化函数

```
function dif = t3_2_ga_fun(X)
    % Question 3 tool (ga optimization function)
  time = [13.15, 13.2, 13.25, 13.3, 13.35, 13.4, 13.45, 13.5, 13.55, 13.6, 13.65, 13.7, 13.75, 13.8,
13.85, 13.9, 13.95, 14.0, 14.05, 14.1, 14.15];
  LengthOfShadow = [3.5331421836659787, 3.546768028783388, 3.561797643044871,
3.5781007154634423, 3.5957507825209465, 3.614934280177165, 3.635425983292742,
3.657218272129789, 3.68054111510794, 3.7051678356047515, 3.731278025020382,
3.7589179107823036, 3.7880878883679556, 3.818701014743102, 3.8508096187685,
3.884585219814337, 3.9199118280900147, 3.9568759924971113, 3.9955347902377225,
4.0357508347270405, 4.077863059250519];
  lat = X(1);
  long = X(2);
  Lg = X(3);
  N = X(4);
```

```
    Y = 2015;

    dif = 0;
    for i = 1:20
        Ly = shadow_length_calculator(Lg, lat, long, ...
            Y, N, time(i));
        dif = dif + (Ly - LengthOfShadow(i)) * (Ly - LengthOfShadow(i));
    end
end
```

附录五 问题四遗传算法代码

问题三遗传算法

```
clear, clc;
% Question 4 solution (use ga to calc solution)

A = [[-1,0,0,0],
[1,0,0,0],
[0,-1,0,0],
[0,1,0,0]];

out=fopen('t4.txt','w');

dif = 1e8;
lat = 0.;
long = 0.;
for i = 1:8
    b = [0, pi/2, pi, pi];
    x = ga(@t4_ga_fun, 4, A, b);
    if t4_ga_fun(x) < dif
```

```
        dif = t4_ga_fun(x);
        lat = x(1);
        long = x(2);
    end
    disp([lat, long, t4_ga_fun(x)]);
end
% disp([lat, long, Lg, 'CHOSEN FOR ', Lg_min]);
disp('-------------------------------------');
fprintf(out, '%f\r%f\r%f\n', lat, long);
```

问题四遗传算法优化函数

```
function dif = t4_ga_fun(X)
            % Question 4 tool (ga optimization function)
    % [ float(row.split(' ')[3]) for row in
open('data/solution/t4_video_raw_data.txt').read().split('\n')[4:-1] ]
    time = [8.91, 8.92666666667, 8.94333333333, 8.96, 8.97666666667, 8.99333333333, 9.01,
9.02666666667, 9.04333333333, 9.06, 9.07666666667, 9.09333333333, 9.11, 9.12666666667,
9.14333333333, 9.16, 9.17666666667, 9.19333333333, 9.21, 9.22666666667, 9.24333333333,
9.26, 9.27666666667, 9.29333333333, 9.31, 9.32666666667, 9.34333333333, 9.36,
9.37666666667, 9.39333333333, 9.41, 9.42666666667, 9.44333333333, 9.46, 9.47666666667,
9.49333333333, 9.51, 9.52666666667, 9.54333333333, 9.56, 9.57666666667, 9.59333333333];
    % [ float(row.split(' ')[4]) for row in
open('data/solution/t4_video_raw_data.txt').read().split('\n')[4:-1] ]
    LengthOfShadow = [2.35666758645, 2.34787871273, 2.33323062896, 2.31272339535,
2.29507849824, 2.28036614856, 2.26864718569, 2.25393756337, 2.23642018095,
2.23044140093, 2.21286219948, 2.19810742691, 2.18931744377, 2.17759748164,
2.15996986486, 2.14238960492, 2.13062511753, 2.11890473493, 2.10421346649, 2.0924928864,
2.07194459878, 2.06608421765, 2.05726025776, 2.03674861963, 2.01913746431,
2.01034664841, 1.99569529627, 1.97515733415, 1.96343611501, 1.95171490058,
1.94290189397, 1.91945921709, 1.91358059991, 1.89598484972, 1.88719370686, 1.8813329453,
```

```
1.86668104261, 1.85495952173, 1.84029828981, 1.83150710622, 1.81978552847,
1.81686458697];
%     adjust_alpha = 0.0085:0.000193:0.0166;
%     for i = 1:42
%         LengthOfShadow(i) = (1 + adjust_alpha(i)) * LengthOfShadow(i);
%     end

    lat = X(1);
    long = X(2);
    Lg = 2;

    Y = 2015;
    N = 193;

    dif = 0;
    for i = 1:42
        Ly = shadow_length_calculator(Lg, lat, long, ...
            Y, N, time(i));
        dif = dif + (Ly - LengthOfShadow(i)) * (Ly - LengthOfShadow(i));
    end
end
```

附录六 问题四视频处理及图像处理代码

问题四视频处理代码

```
#! /usr/bin/env python

import Image
```

```
import subprocess
import argparse
import os

parser = argparse.ArgumentParser()
parser.add_argument('-c', '--capture', action="store_true")
args = parser.parse_args()

video_addr = 'problem\ set/A/Appendix_4.avi'
screen_shot_addr = 'data/screen_shots'
screen_shot_fmt = 'img%03d.jpg'
gray_addr = 'data/gray'
gray_fmt = 'gray_img%03d.png'
interval = '600'

if args.capture:

        print '******Capturing screen shots, plz be patient...******'
        subprocess.call(['rm', '-rf', screen_shot_addr])
        subprocess.call(['mkdir', screen_shot_addr])
        subprocess.call("ffmpeg -loglevel 24 " \
                + "-i " + video_addr \
                + " -f image2 " \
                + "-vf fps=fps=1/" + interval + " " \
                + screen_shot_addr + "/" + screen_shot_fmt, shell=True)

print '******Converting RGB image into GRAY, plz be patient...******'
subprocess.call(['rm', '-rf', gray_addr])
subprocess.call(['mkdir', gray_addr])
for img in os.listdir(screen_shot_addr):
        # img = Image.open(screen_shot_addr + '/' + i).convert('LA')
```

```
        # img.save(gray_addr + '/' + 'gray_' + i.split('.')[0] + '.png')
        subprocess.call(['ffmpeg', '-loglevel', '16', \
                '-i', screen_shot_addr + '/' + img, \
                '-pix_fmt', 'gray', \
                gray_addr + '/' + 'gray_' + img.split('.')[0] + '.tiff'])
        print 'Convert ' + img + " success!"
```

问题四图像处理代码

```
#! /usr/bin/env python
import os
import math
from PIL import Image, ImageDraw, ImageChops

magic_point = (86.5, 0)
magic_length = 886.5 - 204
magic_beginning_time = 8. + 54./60 + 36./3600

def dist(a, b):
        return math.sqrt((a[0]-b[0])**2 + (a[1]-b[1])**2)

def clearSpots(image, G, out, cases):
        class Block:
                no = 0
                left_point = (0, image.size[0])
                right_point = (0, -1)
                def __init__(self, no):
                        self.no = no
                @property
                def length(self):
                        return self.right_point[1] - self.left_point[1]
```

```
            def __str__(self):
                return {'left_point': self.left_point, 'right_point': self.right_point,
'length': self.length}

        def label(x, y, cnt):
            block[x][y] = cnt
            if border[cnt].left_point[1] > y:
                border[cnt].left_point = (x, y)
            if border[cnt].right_point[1] < y:
                border[cnt].right_point = (x, y)

        def bfs(data, x, y, G, cnt):
            # print str(x) + ', ' + str(y)
            ini = data[y, x]
            label(x, y, cnt)
            queue = [(x, y)]
            while len(queue):
                x, y = queue.pop(0)

                if x>0 and block[x-1][y] == 0 and abs(ini - data[y, x-1]) < G:
                    label(x-1, y, cnt)
                    queue.append((x-1, y))
                if y>0 and block[x][y-1] == 0 and abs(ini - data[y-1, x]) < G:
                    label(x, y-1, cnt)
                    queue.append((x, y-1))
                if x<image.size[1]-1 and block[x+1][y] == 0 and abs(ini - data[y, x+1])
< G:
                    label(x+1, y, cnt)
                    queue.append((x+1, y))
```

```
                    if y<image.size[0]-1 and block[x][y+1] == 0 and abs(ini - data[y+1, x])
< G:
                            label(x, y+1, cnt)
                            queue.append((x, y+1))

    def output(cnt):
        new = Image.new('L', image.size)
        draw = ImageDraw.Draw(new)
        for i in xrange(0, image.size[1]):
            for j in xrange(0, image.size[0]):
                if block[i][j] == border[cnt].no:
                    draw.point((j, i), 255)
        return new
    block = [0]*image.size[1]
    for i in xrange(image.size[1]):
        block[i] = [0]*image.size[0]

    border = [Block(0)]

    data = image.load()
    cnt = 0
    for i in xrange(0, image.size[1]):
        for j in xrange(0, image.size[0]):
            if not block[i][j]:
                cnt += 1
                border.append(Block(cnt))
                bfs(data, i, j, G, cnt)
                # print 'New block No. ' + str(cnt)

    border = sorted(border, key=lambda x: x.length, reverse=True)
    for i in border:
```

```
                if i.length > 400:
                        print(i.__str__())

        out.write(str(border[2].right_point[1]) + ' ' \
                + str(border[2].right_point[0]) + ' ' \
                + str(dist(border[2].right_point, magic_point)) + ' ' \
                + str(magic_beginning_time + cases/60) + ' ' \
                + str(dist(border[2].right_point, magic_point)/magic_length*2) + ' ' \
                + '\n')
        print(cnt)
        return output(2)



def cut(image, xoffset, yoffset):
        new = Image.new('L', (image.size[0]-xoffset-5, image.size[1]-yoffset-5))
        new_data = new.load()
        image_data = image.load()
        for i in xrange(xoffset, image.size[0]-5):
                for j in xrange(yoffset, image.size[1]-5):
                        new_data[i-xoffset, j-yoffset] = image_data[i, j]
        return new


def bColor(image, G):
        data = image.load()
        new = Image.new('L', image.size)
        draw = ImageDraw.Draw(new)

        for x in xrange(0, image.size[0] - 1):
                for y in xrange(0, image.size[1] - 1):
                        if data[x, y] > G:
                                draw.point((x, y), 255)
```

```
                    else:
                        draw.point((x, y), 0)
    return new

def main():
    out = open('t4_data.txt', 'w')

    print 'Calculating plz be patient...'
    img_addr = 'data/screen_shots'
    cases = 0.
    for img in os.listdir(img_addr):
        image = bColor(cut(Image.open(img_addr + '/' + img).convert("L"), 870,
800), 210)
        image.save('data/bw/' + img)
        clearSpots(image, 20, out, cases).save('data/result/' + img)
        cases += 1
        print img + ' process successfully!'
        break

if __name__ == '__main__':
    main()
```

附录七 MATLAB 绘图代码

问题一绘图代码

```
clear;
syms W;
% Question 1 solution (direct computation)

% W = 9;
```

```
th = (39 + 54/60 + 26/3600)/180*pi; % Lat
D = -(116 + 23/60 + 29/3600)/180*pi; % Long
Y = 2015; % Year
N = 294; % DayOfYear
Lg = 3;

Ly = shadow_length_calculator(Lg, th, D, Y, N, W);
h = ezplot(Ly, 9:15);
Ly_fun = @(x) double(subs(Ly, x));
x_min = fminunc(Ly_fun, 9, 15);
hold on
plot(x_min, Ly_fun(x_min), 'ro');
text(12, 4, sprintf('\\fontsize{12}W=%f', x_min));
text(12, 3.8, sprintf('\\fontsize{12}Ly=%f', Ly_fun(x_min)));
xlabel('\fontsize{14} W \rightarrow');
ylabel('\fontsize{14} Ly \rightarrow');
title('')
```

问题二多项式拟合代码

```
clear;
% Question 2 attempt (got the longitude, polyfit the [x, y]) (found polyfit realy inaccurative)


% [ float(row.split(',')[0].split(':')[0])+float(row.split(',')[0].split(':')[1])/60 for row in
open('data/appendix_1.csv').read().split('\r\n')[3:] ]
time = [14.7, 14.75, 14.8, 14.85, 14.9, 14.95, 15.0, 15.05, 15.1, 15.15, 15.2, 15.25, 15.3, 15.35,
15.4, 15.45, 15.5, 15.55, 15.6, 15.65, 15.7];
% [ float(row.split(',')[1]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
x = [1.0365, 1.0699, 1.1038, 1.1383, 1.1732, 1.2087, 1.2448, 1.2815, 1.3189, 1.3568, 1.3955,
1.4349, 1.4751, 1.516, 1.5577, 1.6003, 1.6438, 1.6882, 1.7337, 1.7801, 1.8277];
% [ float(row.split(',')[2]) for row in open('data/appendix_1.csv').read().split('\r\n')[3:] ]
```

```
y = [0.4973, 0.5029, 0.5085, 0.5142, 0.5198, 0.5255, 0.5311, 0.5368, 0.5426, 0.5483, 0.5541,
0.5598, 0.5657, 0.5715, 0.5774, 0.5833, 0.5892, 0.5952, 0.6013, 0.6074, 0.6135];
% [ math.sqrt(x[i]**2 + y[i]**2) for i in xrange(0,21) ]
LengthOfShadow = [1.149625826084296, 1.1821989764840775, 1.215296955480429,
1.2490510517989248, 1.2831953397670988, 1.3179931486923595, 1.3533640493230192,
1.389387091490345, 1.4261528564638504, 1.4633998530818568, 1.5014816215991456,
1.5402318169678226, 1.5798533159758852, 1.6201445151590645, 1.6612706131151542,
1.7032906328633408, 1.7462059099659466, 1.790050915476987, 1.8350142724240595,
1.8808750011630226, 1.927918447445327];

syms x_min;
coefs = polyfit(time, LengthOfShadow, 2);
parabola =coefs(1) * x_min * x_min + coefs(2) * x_min + coefs(3);

ezplot(parabola, 12:16)
dy = diff(parabola);
x_min = solve(dy);
W = double(x_min);

hold on
plot(W, double(subs(parabola, W)), 'ro');
text(12.6, 0.7, sprintf('\\fontsize{12}W=%f', W));
text(12.6, 0.6, sprintf('\\fontsize{12}Ly=%f', double(subs(parabola, W))));
xlabel('\fontsize{14} W \rightarrow');
ylabel('\fontsize{14} Ly \rightarrow');
title('')
```

问题二相关度分析代码

```
clear, clc;
% Question 2 correlation bewteen stick length and shadow length computer
```

```
Longitude = [3.0317, -2.1488, -2.2838, -2.2806, -1.9595, -1.9595, -1.9595, -1.919, -1.819, -1.819,
-1.819, -1.8309, -1.6056, -1.6056, -1.5976, -1.5976, -2.6069, -2.5382, 2.9784, 2.9784, -1.9817, -
1.9443, -1.9443, -1.9443, -1.7342, -1.7342, -1.7342, -1.7342, -1.6111, -1.6351, -1.6351, -1.6351]
Stick_length = [0.1426, 1.001, 0.9374, 0.9444, 1.8113, 1.8113, 1.8113, 1.95, 2.3104, 2.3104,
2.3104, 2.2655, 3.1097, 3.1097, 3.3815, 3.3815, 0.4051, 0.4812, 0.1335, 0.1335, 1.7363, 1.8631,
1.8631, 1.8631, 2.6563, 2.6563, 2.6563, 2.6563, 3.2947, 3.1514, 3.1514, 3.1514]

c = corrcoef(Stick_length, Longitude)

% c =
%
%    1.0000   -0.4499
%   -0.4499    1.0000

% 相关系数     相关程度
% 0.00-±0.30   微相关
% ±0.30-±0.50  实相关
% ±0.50-±0.80  显著相关
% ±0.80-±1.00  高度相关
```

问题二误差分析代码

```
clear, clc;
% Question 2 solution error analysis
% 0.421695 -1.734231 2.656331 (24.1613437418, ?99.3641170008, ????????????????) 0.0

time = [14.7, 14.75, 14.8, 14.85, 14.9, 14.95, 15.0, 15.05, 15.1, 15.15, 15.2, 15.25, 15.3, 15.35,
15.4, 15.45, 15.5, 15.55, 15.6, 15.65, 15.7];
LengthOfShadow = [1.149625826084296, 1.1821989764840775, 1.215296955480429,
1.2490510517989248, 1.2831953397670988, 1.3179931486923595, 1.3533640493230192,
```

```
1.389387091490345, 1.4261528564638504, 1.4633998530818568, 1.5014816215991456,
1.5402318169678226, 1.5798533159758852, 1.6201445151590645, 1.6612706131151542,
1.7032906328633408, 1.7462059099659466, 1.790050915476987, 1.8350142724240595,
1.8808750011630226, 1.927918447445327];

syms W;
th = 0.421695; % Lat
D = -1.734231; % Long
Y = 2015; % Year
N = 107; % DayOfYear
Lg = 2.656331;

Ly = shadow_length_calculator(Lg, th, D, Y, N, W);
h = ezplot(Ly, 14.7:16);
hold on
% axis([14.5, 15.9, 1, 2])
plot(time, LengthOfShadow, 'r^');
xlabel('\fontsize{14} W \rightarrow');
ylabel('\fontsize{14} Ly \rightarrow');
title('')
```

问题三第一问误差分析代码

```
clear, clc;
% 0.504278 -1.504183 2.186512 102.986708 2 2 (28.8930011013, -86.1833375153, 西藏自治区日喀则地区聂拉木县) 0.0
% 0.711311 -1.439355 2.124679 191.374165 3 2 (40.7551182212, -82.4689667211, 新疆维吾尔自治区阿克苏地区沙雅县) 4.50059515808
% 0.644784 -1.620742 3.079904 197.013940 3 3 (36.9434018976, -92.8616762796, 青海省海西蒙古族藏族自治州格尔木市) 3.76251303906
```

```
time = [12.683333333333334, 12.733333333333333, 12.783333333333333,
12.833333333333334, 12.883333333333333, 12.933333333333334, 12.983333333333333,
13.033333333333333, 13.083333333333334, 13.133333333333333, 13.183333333333334,
13.233333333333333, 13.283333333333333, 13.333333333333334, 13.383333333333333,
13.433333333333334, 13.483333333333333, 13.533333333333333, 13.583333333333334,
13.633333333333333, 13.683333333333334];
LengthOfShadow = [1.2472562046347977, 1.2227945902726263, 1.198921486169966,
1.1754289642509241, 1.1524395732531925, 1.1299174704375536, 1.1078354796629324,
1.0862542059757467, 1.0650810720316084, 1.0444462647738273, 1.024264126092484,
1.0046403137441777, 0.9854909081265032, 0.9667904943678335, 0.9485847352767173,
0.9309278812024055, 0.9137517496563277, 0.897109051342143, 0.8809737623788804,
0.8654922587753169, 0.8505044679482877];

syms W

x = 12.6:0.1:13.8
y1 = @(x) double(subs(shadow_length_calculator(2.186512, 0.504278, -1.504183, 2015,
102.986708, W), x))
y2 = @(x) double(subs(shadow_length_calculator(2.124679, 0.711311, -1.439355, 2015,
191.374165, W), x))
y3 = @(x) double(subs(shadow_length_calculator(3.079904, 0.644784, -1.620742, 2015,
197.013940, W), x))

plot(x, y1(x), 'r-', x, y2(x), 'g-', x, y3(x), 'b-', time, LengthOfShadow, 'r^')
xlabel('\fontsize{14} W \rightarrow');
ylabel('\fontsize{14} Ly \rightarrow');
title('')
```

问题三第二问误差分析代码

```
clear, clc;
```

```
% 0.497622 -1.956489 2.877535 11.012725 1 1.5 (28.5116403929,
112.098562364, ??????????????????????????) 0.0
% 0.664148 -1.824183 6.423762 103.220454 2 6 (38.0528773721,
104.51798696, ??????????????????????????????????????????????????) 0.1067
% 0.439697 -1.780341 4.827020 294.774313 4 4.5 (25.1927823646,
102.006025394, ??????????????????????????????????????????) 0.0
time = [13.15, 13.2, 13.25, 13.3, 13.35, 13.4, 13.45, 13.5, 13.55, 13.6, 13.65, 13.7, 13.75, 13.8,
13.85, 13.9, 13.95, 14.0, 14.05, 14.1, 14.15];
LengthOfShadow = [3.5331421836659787, 3.546768028783388, 3.561797643044871,
3.5781007154634423, 3.5957507825209465, 3.614934280177165, 3.635425983292742,
3.657218272129789, 3.68054111510794, 3.7051678356047515, 3.731278025020382,
3.7589179107823036, 3.7880878883679556, 3.818701014743102, 3.8508096187685,
3.884585219814337, 3.9199118280900147, 3.9568759924971113, 3.9955347902377225,
4.0357508347270405, 4.077863059250519];

syms W

x = 13.1:0.1:14.2
y1 = @(x) double(subs(shadow_length_calculator(2.877535, 0.497622, -1.956489, 2015,
11.012725, W), x))
y2 = @(x) double(subs(shadow_length_calculator(6.423762, 0.664148, -1.824183, 2015,
103.220454, W), x))
y3 = @(x) double(subs(shadow_length_calculator(4.827020, 0.439697, -1.780341, 2015,
294.774313, W), x))

plot(x, y1(x), 'r-', x, y2(x), 'g-', x, y3(x), 'b-', time, LengthOfShadow, 'r^')
xlabel('\fontsize{14} W \rightarrow');
ylabel('\fontsize{14} Ly \rightarrow');
title('')
```

问题三相关性分析代码

```
Longitude = [167.76204241430503, -147.8231111437524, -147.8231111437524, -147.8231111437524, -120.13779048303101, -120.13779048303101, -120.13779048303101, -120.13779048303101, -111.56061228992259, -111.56061228992259, -111.56061228992259, -111.56061228992259, -106.12897239208239, -107.22332178078226, -107.88795282313401, -108.74165993787892, -104.20383420044281, -104.20383420044281, -104.51896098776477, -104.51896098776477]
Stick_length = [0.5789, 0.8278, 0.8278, 0.8278, 2.5744, 2.5744, 2.5744, 2.5744, 4.1821, 4.1821, 4.1821, 4.1821, 5.7497, 5.3576, 5.1356, 4.8933, 6.5802, 6.5802, 6.4238, 6.4238]

c = corrcoef(Stick_length, Longitude)
```

问题三解离散程度分析代码

```
clear, clc;
std =
[12.9403543913,4.59300903024,6.10650574716,0.0,5.06066288688,8.58503973153,1.9644054234,7.0305318739,0.0,6.5073,10.9594467711,7.81308265169,8.45104689926,6.85420135952,0.320516001941,0.811162694455,0.0,0.0,0.0,4.04321280265,2.47180970748,0.0,2.33857169911,0.0,10.3473244301,4.81938807079,0.0,0.0,1.95955647812,0.1067,0.458560451304,0.0,0.0,17.01665,2.2261,17.9996,1.64445,16.7225510689,0.0,22.2474565016, 0.0, 1.84285875798, 0.0, 0.0, 31.6190263609, 1.49493305201, 0.0, 0.0, 5.82843757001, 3.20593944227, 4.50059515808, 3.76251303906, 12.4434171563, 2.71200185322, 2.76690786382, 16.1002463936, 6.8976491849, 12.7720127512, 17.9430055125, 2.02233833298, 8.99172526114, 2.48129268565, 1.32649111098, 0.0, 20.2145917701, 0.0, 32.5926557606, 1.87503160173, 0.0, 0.551240190601, 0.0, 6.52360817107];
hist(std)
xlabel('\fontsize{14} \bfStandard Diviation \rightarrow');
title('')
```

问题四误差分析代码

```
clear, clc;
% 0.7163   -1.9658    0.0010 (41.0409668652, ?112.632043367, ??????????????????) 0.0
```

```
% [ float(row.split(' ')[3]) for row in
open('data/solution/t4_video_raw_data.txt').read().split('\n')[4:-1] ]
time = [8.91, 8.92666666667, 8.94333333333, 8.96, 8.97666666667, 8.99333333333, 9.01,
9.02666666667, 9.04333333333, 9.06, 9.07666666667, 9.09333333333, 9.11, 9.12666666667,
9.14333333333, 9.16, 9.17666666667, 9.19333333333, 9.21, 9.22666666667, 9.24333333333,
9.26, 9.27666666667, 9.29333333333, 9.31, 9.32666666667, 9.34333333333, 9.36,
9.37666666667, 9.39333333333, 9.41, 9.42666666667, 9.44333333333, 9.46, 9.47666666667,
9.49333333333, 9.51, 9.52666666667, 9.54333333333, 9.56, 9.57666666667, 9.59333333333];
% [ float(row.split(' ')[4]) for row in
open('data/solution/t4_video_raw_data.txt').read().split('\n')[4:-1] ]
LengthOfShadow = [2.35666758645, 2.34787871273, 2.33323062896, 2.31272339535,
2.29507849824, 2.28036614856, 2.26864718569, 2.25393756337, 2.23642018095,
2.23044140093, 2.21286219948, 2.19810742691, 2.18931744377, 2.17759748164,
2.15996986486, 2.14238960492, 2.13062511753, 2.11890473493, 2.10421346649, 2.0924928864,
2.07194459878, 2.06608421765, 2.05726025776, 2.03674861963, 2.01913746431,
2.01034664841, 1.99569529627, 1.97515733415, 1.96343611501, 1.95171490058,
1.94290189397, 1.91945921709, 1.91358059991, 1.89598484972, 1.88719370686, 1.8813329453,
1.86668104261, 1.85495952173, 1.84029828981, 1.83150710622, 1.81978552847,
1.81686458697];
% adjust_alpha = 0.0085:0.000193:0.0166;
% for i = 1:42
%     LengthOfShadow(i) = (1 + adjust_alpha(i)) * LengthOfShadow(i);
% end


syms W
th = 0.7163; % Lat
D = -1.9658; % Long
Y = 2015; % Year
N = 193; % DayOfYear
Lg = 2;
```

```
Ly = shadow_length_calculator(Lg, th, D, Y, N, W);
y = @(x) double(subs(Ly, x));
x = 8.85:0.05:9.6;
h = plot(x, y(x));
hold on
% axis([14.5, 15.9, 1, 2])
plot(time, LengthOfShadow, 'r^');
xlabel('\fontsize{14} W \rightarrow');
ylabel('\fontsize{14} Ly \rightarrow');
title('')
```

附录八 辅助代码

方差计算代码

```
#! /usr/bin/env python
import argparse
import re
import numpy
import math

parser = argparse.ArgumentParser()
parser.add_argument('file', default='script/t2_raw_solution.txt')
parser.add_argument('offset', default=0)
parser.add_argument('number', default=3)
args = parser.parse_args()

r = re.compile('[ \t\b\r]+')
raw = open(args.file).read()
offset = int(args.offset)
n = int(args.number)
data =  [[] for i in xrange(n)]
cnt = 0
# print raw.split('\n')
for row in raw.split('\n'):
        if offset > 0:
                offset -= 1
                continue
        cnt += 1
        if cnt % 5 ==0:
                continue
        # print str(cnt) + ": " + row
```

```
        a = row.split(' ')
        while '' in a:
                a.remove('')
        for i in xrange(n):
                data[i].append(float(a[i]))

cnt -= cnt / 5 + 1

print data[1][100:120]
print data[2][100:120]

while cnt > 0:
        std = 0.
        for i in xrange(n):
                std += numpy.std(numpy.array(data[i][0:4]))
                data[i] = data[i][4:]
        cnt -= 4
        # print std
```

系统版本替换代码

```
#! /usr/bin/env python

import argparse

parser = argparse.ArgumentParser()
parser.add_argument('param')
args = parser.parse_args(['10.10.5'])
args = parser.parse_args()
```

```
first = '<?xml version="1.0" encoding="UTF-8"?>\n<!DOCTYPE plist PUBLIC "-
//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-
1.0.dtd">\n<plist
version="1.0">\n<dict>\n\t<key>ProductBuildVersion</key>\n\t<string>14F27</strin
g>\n\t<key>ProductCopyright</key>\n\t<string>1983-2015 Apple
Inc.</string>\n\t<key>ProductName</key>\n\t<string>Mac OS
X</string>\n\t<key>ProductUserVisibleVersion</key>\n\t<string>10.10.5</string>\n\
t<key>ProductVersion</key>\n\t<string>'
second = '</string>\n</dict>\n</plist>\n'

print(args.param)
target = open('/System/Library/CoreServices/SystemVersion.plist',
'w').write(first+args.param+second)
```

latex 表格代码生成器

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

# target_string:
'\cellcolor[rgb]{0.74,0.90,0.70}{}&\cellcolor[rgb]{.7,.9,.9}4&\cellcolor[rgb]{.8,.8,.9}5&\cellc
olor[rgb]{.8,.8,.9}5\\'



color = ['FUCK', [188., 255., 255.], [222., 242., 217.], [255., 255., 255.], [254., 228., 211.], [253.,
201., 168.]]

t3_1_global_color = [1,5,2,3,1,1,1,1,1,1,1,1,1,1,4]
t3_1_north_color = [2,2,5,1,3,1,1,1,4,1,5,1,1,1,2]
t3_1_global_location = ["非洲以南的海上","斯里兰卡以西的海上","印度洋中心","印度洋
中心","中东地区里海北侧","巴基斯坦北侧的乡下","西藏自治区日喀则地区聂拉木县","
```

印度东侧曼尼普尔近锡尔杰尔","印度洋北侧","印度洋北侧","新疆维吾尔自治区阿克苏地区沙雅县","青海省海西蒙古族藏族自治州格尔木市","非洲南侧","非洲东南侧","印度洋中心","印度洋中心"]

t3_1_north_location = ["孟买西侧阿拉伯海域", "马尔代夫西北方向阿拉伯海域", "斯里兰卡东南方向印度洋海域", "印度尼西亚棉兰", "舍甫琴柯堡西侧里海海域", "阿富汗中部地区", "西藏自治区日喀则地区吉隆县", "西藏自治区那曲地区那曲县", "马哈奇卡拉东侧里海海域", "塔吉克斯坦西南角", "新疆维吾尔自治区阿克苏地区沙雅县", "青海省海西蒙古族藏族自治州", "阿曼东侧阿拉伯海海域", "印度索姆纳特东侧", "印度金奈东侧阿拉伯海海域", "孟加拉湾安达曼群岛西侧"]

t3_2_global_color = [3, 2, 3, 1, 1, 1, 2, 1, 2, 2, 2, 1, 1, 1, 2, 1, 2, 2, 1, 1]

t3_2_north_color = [1, 1, 1, 4, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 4, 1, 3, 1, 1, 5]

t3_2_global_location = ["南极洲海岸边", "南极洲海岸边", "广西壮族自治区梧州市藤县 321 国道", "澳大利亚与大洋洲的中点", "印度洋南侧", "澳大利亚与大洋洲的中点", "俄罗斯外贝加尔边疆区", "澳大利亚西南侧", "澳大利亚西侧", "蒙古曼达勒敖包", "澳大利亚西南侧", "俄罗斯萨哈共和国", "澳大利亚西侧", "俄罗斯西伯利亚平原南侧", "俄罗斯与蒙古的国境线附近", "非洲南侧", "印度洋中心", "印度洋中心", "印度洋中心", "印度洋中心"]

t3_2_north_location = ["蒙古巴彦德勒格尔", "湖南省益阳市桃江县", "湖南省永州市零陵区", "柬埔寨东北侧", "越南陆地最南端", "北冰洋中心", "俄罗斯西伯利亚平原中心", "蒙古共和国北侧", "蒙古共和国南侧", "内蒙古自治区阿拉善盟阿拉善左旗", "北冰洋中心", "俄罗斯西伯利亚平原北岸", "俄罗斯西伯利亚平原中部", "蒙古共和国与俄罗斯国境线北侧", "蒙古共和国与俄罗斯国境线北侧", "蒙古共和国乌兰巴托", "内蒙古自治区巴彦淖尔市乌拉特中旗呼四线", "内蒙古自治区鄂尔多斯市鄂托克前旗上海街", "云南省楚雄彝族自治州禄丰县", "云南省临沧市云县漫湾镇嘎止村"]

```
starter = '\cellcolor[rgb]{.9,.9,.9}$%d\sim%d$ &'
cell = '\cellcolor[rgb]{%.2f,%.2f,%.2f}{%s}'
ender = '\\\\\n'

out = open('latex.txt', 'w')
```

```
def cell_filler(c, s):
        return cell % (color[c][0]/255, color[c][1]/255, color[c][2]/255, s)

for Lg_min in xrange(5):
        out.write(starter % (Lg_min*1.5, (Lg_min+1)*1.5,))
        for i in xrange(4):
                out.write(cell_filler(t3_2_north_color[i*4 + Lg_min], t3_2_north_location[i*4
+ Lg_min]))
                if i != 3:
                        out.write('&')
        out.write(ender)
```

弧度转角度代码

```
#! /usr/bin/env python

import argparse
import math

parse = argparse.ArgumentParser()
parse.add_argument('latitude')
parse.add_argument('longitude')
args = parse.parse_args()

latitude = float(args.latitude)
longitude = float(args.longitude)

print str(latitude * 180 / math.pi) + ', ' + str(-longitude * 180 / math.pi)
```