# 本 選 課 題 解 説

スーパーコン０９審査委員会
大阪大学・東京工業大学

## ☆恒星写真探索問題☆

晴れた夜に空を見上げるとたくさんの星が見える．星座は，地球から見て近い星をつないだものだが，たとえば同じおうし座に属する星でも，すばる（プレアデス星団）は地球から 410 光年，アルデバランは 60 光年の距離にあり，実際の距離が近いわけではない．

遠い将来，人類が太陽系外に進出したとき，遠くの惑星から空を見上げたら，やはりたくさんの星が見えるだろうが，見えている星の配置は違い，星座の形もまったく違っているだろう．

とすれば，空に見える星の写真を見れば，それがどの星から撮られた写真なのかがわかるはずだ．たとえば，遠い恒星系の惑星に不時着した宇宙船から，星を写した写真が送られてきたら，どの星に救助に向かうべきかがわかるに違いない．

## ☆課題☆

恒星写真探索問題

ある空間内に存在するすべての星の座標が与えられている．それに対して様々な星から撮影した「全天写真」が数多く与えられる．これらの写真から，その各々が，どの星からどの方向を向いて撮影されたものかを計算してほしい．多数の写真が出題されるので，制限時間内に最もたくさんの写真に対して正解を出したチームが優勝である．

用語解説（本課題のためのもの）

- 恒星（もしくは星）： 課題の対象となる「星」は，すべて光を出す恒星のこと．この課題の想定している世界では（すばらしい技術により）恒星から写真を撮ることも可能であると仮定する．
- 全天写真： ある星を中心として半径 $r$ 以内に含まれるすべての星を，半径 1 の球面に射影（次に説明）したもの．つまり，半径 $r$ より遠い星は写らないものとする．正確には，撮影した星を $A$，半径 $r$ 以内のすべての星を $B_1, ..., B_n$ としたとき，こちら $B_1, ..., B_n$ の射影座標（次に説明）を表す $n$ 組のデータのことを，全天写真 もしくは（全天）写真データ と呼ぶことにする（図１）．
- 射影： 撮影した星を $A$，その星から半径 $r$ 以内のすべての星を $B_1, ..., B_n$ とする．このとき，$A$ から各 $B_i$ への線分（場合によってはそれを延長したもの）と，$A$ を中心とする半径 1 の球面との交点を，星 $B_i$ の 射影 と呼ぶことにする（図２）．

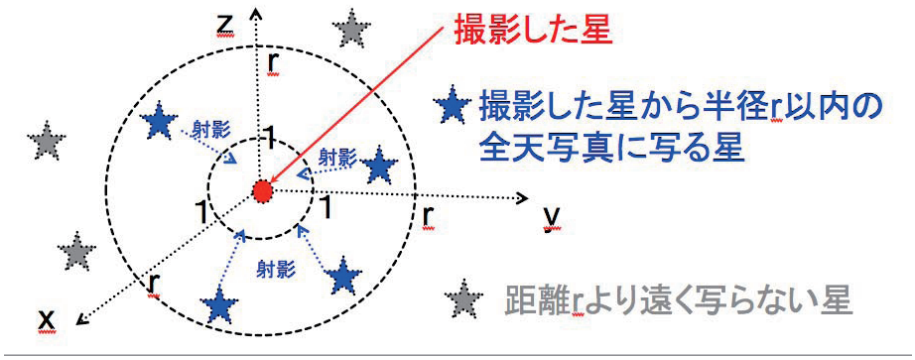


図 1: 全天写真と射影

- 射影座標： 撮影した星を $A$，その星から半径 $r$ 以内のすべての星を $B_1, ..., B_n$ とする．このとき，各星 $B_i$ の射影を，ある星（たとえば $B_1$）の射影の座標を星 $A$ を原点として $(0, 0, 1)$ となるように（単位球を回転させ）表した座標を（星 $B_1$ を軸とする）射影座標 と呼ぶことにする（図３）．また，$B_1$ を射影座標の 基準星 と呼ぶことにする．

問題例の詳細

入力データは，すべての星の絶対座標と，全天写真 1000 枚分の写真データからなる．1000 枚分のデータはすべて最初に与えられるので，どの順序で解いてもよい．

(1) 全空間の星の座標データ

(1-1) 全空間の星の座標データは $(x, y, z)$ の三次元座標として与えられる．

(1-2) 星の数：練習問題と第一次選抜では 10 万個，決勝では 50 万個または 100 万個．(第一次選抜，決勝については下記の審査の項を参照)

(1-3) 全ての星の座標の範囲は，$-3000 \leq x, y, z \leq 3000$ とする．

(2) 写真に写る星の射影座標データ

(2-1) 写真に写る星の射影座標も $(x', y', z')$ の三次元座標の組として与えられる．ただし，定義

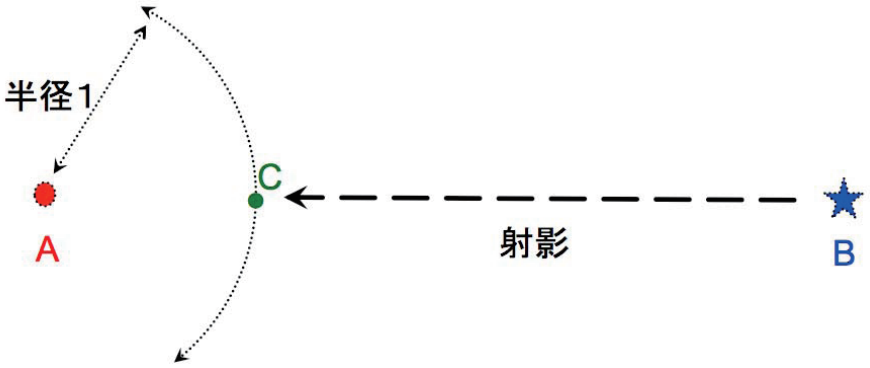


図 2: 射影

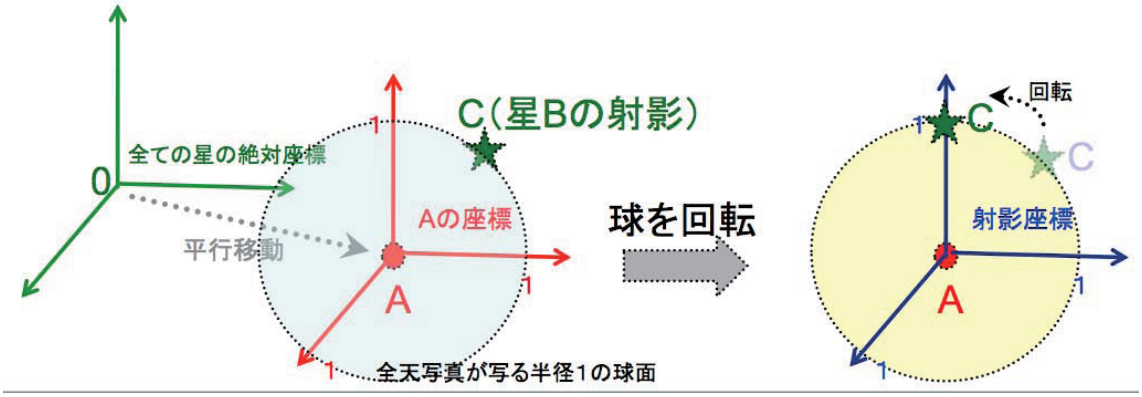


図 3: 射影座標と基準星

から $x'^2 + y'^2 + z'^2 = 1$ である．

(2-2) 写真に写る範囲：中心の星から距離 $r$ 以下の星がすべて写る．ただし，$r$ の値は写真ごとに微妙に違う．$50 \leq r \leq 51$ の範囲の値であることだけがわかっている．

(2-3) 写真に含まれる星の数 $n$ は $50 \leq n \leq 10000$ の範囲にあるとする．したがって，距離 $r$ 以内に 50 個未満もしくは 10000 個より多くの星がある星は最初から候補として排除してよいが，そのような星は多くない．

(2-4) 写真データの最初の星は基準星とする．つまり，写真データの最初の座標はかならず $(0, 0, 1)$ である．

プログラミングの注意

(1) プログラミング言語は C 言語．基本的に ANSI 準拠．GCC や Visual C++などに特有の仕様は，スーパーコンピュータ（SX9）では使えない可能性が高い．PC でコンパイルできたものが SX9 でコンパイルできるとは限らないことに注意．

(2) データの読み込みと解答の出力には実施委員会で用意した関数を使う．この関数は書き換えてはならない．巨大なデータの入力時間を節約するために，バイナリ化した入力データを専用の入力用関数で読み込む．これらの関数が定義されたヘッダーファイル (sc09.h) を参加者専用ウェブページ (URL は会場で公表する) に掲載するのでプログラムソースに読み込むこと．これらの入出力関数を用いたサンプルプログラムソースや各自で作ったデバッグ用のテキストデータをバイナリ化するプログラムも上記ウェブページに掲載するので参照のこと．

(3) 解答はひとつの写真についての解が出るたびに出力すること．解答の出力用の関数には，写真の番号，撮影した星の番号，基準星の番号を与える（詳細は上記ウェブページに掲載するヘッダーファイル (sc09.h) を参照）．時間制限内に出力されたすべての結果が審査に使われる．ただし，ひとつの写真に対して複数の解が出力された場合は，最初のものが使われる（最後のものではないことに注意）．

(4) 使用可能メモリーサイズは 256GB まで．ひとつの配列として取れるサイズに上限はない．

(5) その他 SX9 でのコンパイルやジョブの投入の方法の詳細，ベクトル化や並列化に関する注意

などもコンテスト Wiki に掲載するので参照すること．

審査

(1) コンテスト最終日に提出されたプログラムを実施委員会が実行させて審査する．ただし，並列 CPU 数は，各チームが申告した数を用いる．
(2) 写真ひとつに対して，撮影した星の番号が正解で，基準星の番号が不正解の場合１点を与える，撮影した星の番号と基準星の番号の両方が正解の場合には２点を与える．撮影した星の番号が不正解で，基準星の番号だけが正解の場合は０点．
(3) 解答した全ての写真の点数の合計点により審査する．同点の場合には最後の解が出力されるまでの実行時間がより短いものを優位とする．
(4) 第一次選抜では最大 8CPU 並列，時間制限１０分で審査する．第一次選抜により，東西３チームずつ合計６チームを決勝進出チームとする．決勝は最大 16CPU 並列，時間制限２０分で審査する．（コンテスト期間中に練習のために流す場合には，最大 4CPU 並列，時間制限１０分とする）

**☆課題攻略のヒントと注意☆**

課題への取り組み方

写真を撮った方向がわからないので工夫が必要である．同じ星からの写真であっても撮影方向が違えば各点は違う座標で表現される．そこで，ひとつの考え方は，撮影方向によって変化しない量を計算することである．たとえば，写真に写っている特定の二点間の距離は撮影方向によって変化しない．このような「回転に対して不変な量」で比較するのが有効な方法だろう．

また，写真に写っているすべての点がどの星であるかを決定するとなると多くの計算量を要するが，問題ではそこまで要求していない．候補となる星から撮影した写真が，問題の写真と完全に一致することを確認しなくても，いくつかの「回転に対して不変な量」で候補を絞り込んで，候補がひとつに決まればよい．いかにして計算をさぼるかが重要である．

精度に関する注意

通常の数値計算では残念ながら誤差を避けることはできない．この点を忘れないでおこう．具体的には 2 つの重要な点がある．一つは誤差を仮定してプログラムを作ることであり，もう一つは大きな誤差を生み出さないという点である．

たとえば，星の位置が同じかどうかをチェックする際，変数 (x, y, z) に保持されている入力した写真データの座標と，変数 (x0, y0, z0) に保持されている計算した座標が一致するかを確かめる必要が出てくる．その際，

```
if(x == x0 && y == y0 && z == z0)
```

のようなテストはまったく役に立たない可能性がある．計算による誤差のためにキッチリと等しくならない場合があるからだ．

もちろん，出題者側が写真データを計算する際にも誤差は生じている．もしも，出題者とまったく同じ計算で (x0, y0, z0) の値を計算したならば，キッチリと等しくなるかもしれない．しかし，

式の上では同じでも，加算や乗算の順序で，誤差の出方が微妙に違う場合も少なくないし，その場合にはキッチリ等しくなるとは思えない．したがって，

$$\texttt{if(|x-x0|<}\epsilon\texttt{ \&\& |y-y0|<}\epsilon\texttt{ \&\& |z-z0|<}\epsilon\texttt{)}$$

のように，ある程度の誤差（$\epsilon$）の範囲内での比較を考えるべきである．では，$\epsilon$ をどの程度の値にすればよいか？これはデータの性質によって決まってくる．今回のデータに対する許容誤差範囲については，例題や今後ヒントとして与えられる情報に応じて判断してほしい．また，あくまで一般論だが，今回の課題では安全を見て許容誤差範囲を少し大きくとっておく方がよい．それにより複数の解の候補が出てくる可能性があるが，多数の星が，すべて許容誤差範囲の射影座標を持つ可能性は非常に低いと考えてよいだろう．

もう一つは，計算によっては大きな誤差を出さない注意である．計算によっては同じような計算をしても大きな誤差が出てしまう場合がある．例えば，$10000.0+0.00001$ は正確には $10000.00001$ であるが，(C 言語の float 型などのように) 精度が 6 桁程度しかない場合には $10000.0+0.00001 = 10000.0$ となってしまう．このような誤差を「情報落ち」といい，特に，絶対値の大きい値と小さい値の足し算や引き算を多数繰り返す場合には問題となる．情報落ちを避けるためには，多数の足し算や引き算を行う際に，計算の順序を変更して，小さいもの同士，大きいもの同士を先に計算し，絶対値が異なる数の計算の回数を必要最小限に抑えるといった工夫が必要である．

また，$123.456 - 123.455 = 0.001$ のような計算では，6 桁の精度 が 1 桁に落ちてしまうが，このような誤差を「桁落ち」という．桁落ちは，値が近い二つの数の引き算を行う場合に起こる．桁落ちを避けるためには，やはり計算の順序を変えるなどして，大きさの近い二つの数の引き算をなるべく避けることが肝要である．

情報落ちや桁落ちは，他の計算誤差（たとえば四捨五入などによる丸め誤差）などと比べて誤差の蓄積が顕著であり，気づかずにプログラムしてしまうことが多いので注意が必要である．もっとも，今回の課題は，わざわざ誤差が蓄積するような計算をしない限り，情報落ちや桁落ちのせいで正解が求まらないということにはしていないので，最初からこれらの誤差の問題に神経質になる必要はないだろう．どうしても解がうまく求まらない場合には，これらの誤差の問題を思い出してみて欲しい．