The Doodle Verse

1. Introduction

The DoodleVerse will make your dreams come true.

2. Overview

The problem can be broken up into two main tasks: (1) taking the input image and extracting the ideal feature points, and (2) finding the ideal set of stars whose formation matches this set of feature points.

We start by framing this as an optimization problem. If $\mathbb{S}$ is our set of stars and F is the matrix defined by our feature points, then we are looking for the subset $S \subset \mathbb{S}$ that minimizes the following: $||TF - S||_F$. Here T is a map from the Cartesian "feature space" to the space that our stars lie in that preserves the shape of our feature points (i.e. doesn't distort angles and relative distances).

The number of stars in our $\mathbb{S}$ set (which has been restricted to stars visible with the naked eye) is around 3000, so for n feature points, there are around 3000 choose n possibilities for S. The search space is too large for a brute force method. We will proceed by breaking this problem into smaller problems to the point where a random search will yield sufficiently quick and accurate results.

3. Warmup

blah

a. Mollweide Projection

One issue is that our extracted feature points will have Cartesian coordinates in the (x, y)-plane and our star data set is in spherical coordinates. Here is a readout of the first 10 stars of our data set:

RA	Dec	Mag
float64	float64	float64
0.02662528	−77.06529438	4.78
0.03039927	−3.02747891	5.13
0.03266433	−6.01397169	4.37
0.03886504	−29.72044805	5.04
0.06232565	−17.33597002	4.55
0.07503398	−10.50949443	4.99
0.08892938	−5.70783255	4.61
0.13976888	29.09082805	2.07
0.15280269	59.15021814	2.28
0.15583908	−27.9879039	5.42

The "RA" or "right ascension" measures the distance from a central meridian and ranges from 0 hours to 24 hours (which corresponds to $0° - 360°$). To convert to degrees we need to multiply this column by 15. The "Dec" or "declination" measures the distances above or below the equator and ranges from $-90°$ to $90°$.

To convert from spherical coordinates to Cartesian coordinates, we need to pick a center point and project the stars in the neighborhood of this point onto a plane. If we attempt to project all or most of the stars, this will distort the stars far from our center point, which would be undesirable considering we are trying to match shapes. This restricts our search to small neighborhoods at a time.

The Mollweide projection of a star with spherical coordinates (λ, ϕ) centered around (λ_c, ϕ_c) can be obtained as follows:

$$x = R\frac{2\sqrt{2}}{\pi}(\lambda - \lambda_c)cos(\theta)$$

$$y = R\sqrt{2}sin(\theta)$$

Where θ is the angle defined by: $2\theta + sin(2\theta) = \pi sin(\phi - \phi_c)$. Since θ is implicitly defined, we cannot solve for it directly. But the following iteration will converge to its value after a few iterations (it converges slow for points far from our center point, but that is not a concern for us).

$$\theta_0 = \phi - \phi_c$$

$$\theta_{k+1} = \theta_k + \frac{2\theta_k + sin(2\theta_k) - \pi(sin\phi - \phi_c)}{2 + 2cos(2\theta_k)}$$

Here is a Python script which will compute the Mollweide projection for given spherical coordinates:

```
def project(ra, dec, c_ra, c_dec):
        # Finds the Mollweide projection coordinates for the point (ra,dec) around
        # point (c_ra,c_dec).

        dec = dec - c_dec

        # Find theta.
        theta_0 = dec
        epsilon = 10**-6
        error = 1+epsilon

        while error > epsilon:
            m = (2*theta_0+np.sin(2*theta_0)-np.pi*np.sin(dec))/(2+2*np.cos(2*theta_0))
            theta_1 = theta_0 - m
            error = np.abs(theta_1 - theta_0)
            theta_0 = theta_1

        # Compute (x,y) coordinates.
        x = 2*np.sqrt(2)*(ra-c_ra)*np.cos(theta_0)/np.pi
        y = np.sqrt(2)*np.sin(theta_0)

        return [x,y]
```

Use this script to convert the points found in (something) to (x,y) coordinates centered at the point (???).

b. Preliminary Search

We wish to determine if there is possible good match for our feature points in the set obtained from the previous section.

c. Procrustes

Here we address the problem of finding the ideal transformation T between our set of feature points F and a predetermined subset of stars $S \subset \mathbb{S}$. Here we are assuming F and S are the same size $(2 \times k)$ and represent their respective collections of points in Cartesian coordinates

(each column is a point). Finding this transformation will be necessary to determine the fit of a possible match.

We want to preserve the shape of F, and for now let's assume S and F are normalized, so we don't want to change the scale of F: our transformation will preserve distances and angles between points (and thus the shape). This simplifies the problem because it constrains T to be orthogonal. Also, let's assume that both sets of points are centered around the origin (i.e. their mean is $\begin{bmatrix}0\\0\end{bmatrix}$). Basically, we are looking for the best rotation of F that minimizes the distance between its points and the points in S. Precisely, we are looking for the orthogonal T that minimizes $||TF - S||_F^2$.

Let $U\Sigma V^T$ be the SVD of the matrix SF^T. Show that the orthogonal T that minimizes $||TF - S||_F^2$ is equal to UV^T.

Hint: one proof takes advantage of the symmetry of the Frobenius inner product:

$$tr(AB^T) = \langle A, B\rangle_F = \langle B, A\rangle_F = tr(BA^T)$$

Also remember that multiplying a vector by an orthogonal matrix doesn't change its Frobenius norm.

(Move this to lab section?!?)

Now let's generalize this to our actual problem: how do we find such a transformation when the sets of points are not normalized or centered around the origin? First, it's easy to reduce to problem to the previous case by normalizing F and S and shifting them by their mean:

$$\hat{F} = \frac{F}{||F||_F} - mean(F) \quad \text{and} \quad \hat{S} = \frac{S}{||S||_F} - mean(S)$$

Now we can easily find the matrix $\hat{T}$ that minimizes $||\hat{T}\hat{F} - \hat{S}||_F$.

Write a formula to find F^*, the transformed set of points that most closely matches S.

Hint: we want $mean(F^*) = mean(S)$ and $||F^*||_F = ||S||_F$ to ensure the transformed points are centered around the same point and have the same scale.

d. Evaluating the Match

Brightness stuff

4. Lab

1. Question 1....
2. Question 2...
3. Question 3....