CosmoAI model

Hans Georg Schaathun

September 23, 2022

1 Images

The simulator maintains three images.

1. The actual source. This is an image of the remote galaxy drawn with the origin in the line of sight through the centre of the lens. Its centre of mass is at P_{ACT}.
2. The apparent source, used only as an intermediate calculation. It is the actual source, translated so that it lies behind the distorted image as seen by the observer, and then rotated to fall on the x axis.
3. The distorted image. This is the image seen. It is drawn in a coordinate system in the lens plane, with the origin in the centre of mass of the lens.

2 Variables and Parameters

Class Variables	Mathematical Notation	Calculation
`CHI`	$\chi = \frac{chi_L}{\chi_S}$	User Setting
`size`	Image size (not part of the model)	User Setting
`einsteinR`	R_E	User Setting
`sourceSize`	σ	User Setting
`nterms`	Number of terms after truncation	User Setting
`actualX`, `actualY`	P_{ACT}	User Setting
`apparentX`, `apparentY`	$P_{\text{APP}} = \rho_1 P_{\text{ACT}}$	`updateXY()` – not used
`actualAbs`	$\|\|P_{\text{ACT}}\|\|$	`updateXY()`
`apparentAbs`	$\|\|P_{\text{APP}}\| = \rho_1 \|\|P_{\text{ACT}}\|\|$	`updateXY()`
`alphas_val[m][s]`, `betas_val[m][s]`	α, β **TODO** (floating point values)	`calculateAlphaBeta()`
`alphas_v[m][s]`, `betas_l[m][s]`	α, β **TODO** (algebraic expressions)	`initAlphaBeta()` loading from `50.txt`

Intermediate Variables	Mathematical Notation	Calculation
`xi1`, `xi2`	Cartesian coordinates $\xi = (\xi_1, \xi_2)$ of a point in the apparent source. Should possibly be η	Return value from `getDistortedPos`
`ratio1`, `ratio2`	ρ_1, ρ_2	`updateXY`
`r`, `theta`	Polar coordinates (r, θ) of a point in the distorted image (lens plane)	Arguments to `getDistortedPos()`
`c_p`	$c_+ = 1 + s/(m+1)$	`getDistortedPos()`
`c_m`	$c_+ = 1 - s/(m+1)$	`getDistortedPos()`

3 The `distort()` function

```
void Simulator::distort(int begin, int end, const cv::Mat& src, cv::Mat& dst)
    // Iterate over the pixels in the image distorted image.
    // (row,col) are pixel co-ordinates
    for (int row = begin; row < end; row++) {
        for (int col = 0; col < dst.cols; col++) {

            int row_, col_;  // pixel co-ordinates in the apparent image
            std::pair<double, double> pos ;

            // Set coordinate system with origin at x=R
            double x = (col - apparentAbs - dst.cols / 2.0) * CHI;
            double y = (dst.rows / 2.0 - row) * CHI;

            // Calculate distance and angle of the point evaluated
            // relative to center of lens (origin)
            double r = sqrt(x * x + y * y);
            double theta = atan2(y, x);

            pos = this->getDistortedPos(r, theta);

            // Translate to array index
            row_ = (int) round(src.rows / 2.0 - pos.second);
            col_ = (int) round(apparentAbs + src.cols / 2.0 + pos.first);

            // If (x', y') within source, copy value to imgDistorted
            if (row_ < src.rows && col_ < src.cols && row_ >= 0 && col_ >= 0)
                auto val = src.at<uchar>(row_, col_);
                dst.at<uchar>(row, col) = val;
```

```
                }
            }
        }
}
```

Suppose the distorted image is an $m \times n$ matrix. We rewrite the pixel coordinates (i, j) as (x, y) to get a canonical Cartesian coordinate system centered at the apparent location of the source.

$$x = (j - ||P_{APP}|| - n/2) \cdot \chi \tag{1}$$
$$y = (-i + m/2) \cdot \chi \tag{2}$$

Given the Cartesian coordinates (x, y), we find Polar coordinates (r, θ) as

$$r = \sqrt{x^2 + y^2} \tag{3}$$
$$\theta = \begin{cases} \tan^{-1} \frac{y}{x}, \text{ if } x \geq 0 \\ \pi + \tan^{-1} \frac{y}{x}, \text{ if } x < 0 \end{cases} \tag{4}$$

The `getDistortedPos` method implements the main coordinate distortion functions and map $(r, \theta) \mapsto \xi = (\xi_1, \xi_2)$.

- **TODO** There is a likely scaling error in this formula.
- **TODO** We should use η for the position in the source plane and ξ in the lens plane. Then $\xi = \chi\eta$. This is not done consistently at the moment.