Laboratório VISGRAF

Instituto de Matemática Pura e Aplicada

360 Camera Rig Calibration

Rafael Romeiro and Luiz Velho

Technical Report TR-25-06 Relatório Técnico

September - 2025 - Setembro

360 Camera Rig Calibration

Rafael Romeiro[1] and Luiz Velho[1]

[1]IMPA - Instituto de Matematica Pura e Aplicada

1 Introduction

1.1 Motivation

Omnidirectional cameras are increasingly popular in computer vision and robotics due to their wide field of view (FOV), enabling a complete 360° view of the surrounding scene. They are particularly useful in applications such as autonomous navigation, panoramic imaging, surveillance, and immersive virtual reality environments.

1.2 Definition of the Problem

Accurate camera calibration is an essential prerequisite for metric measurements, 3D reconstruction, and robust perception tasks. This work presents a comprehensive calibration framework applicable to both single omnidirectional cameras and rigs composed of multiple such cameras. The approach is based on a generalized camera model and uses 2D calibration patterns to estimate intrinsic and extrinsic parameters.

1.3 Related Work

Traditional calibration approaches focus on specific sensor designs, such as parabolic mirrors or fisheye lenses, which limits their applicability to different hardware configurations. In contrast, generalized models, such as those implemented in the OCamCalib toolbox by Scaramuzza [8], provide flexibility to calibrate various catadioptric and dioptric systems. Our methodology is based on Scaramuzza's work.

2 Mathematical Fundamentals

2.1 Projective Geometry and Camera Calibration

Camera calibration relates 3D world coordinates to 2D image coordinates. In the case of central omnidirectional cameras, the single-viewpoint property ensures that all incident light rays pass through a single effective viewpoint, simplifying ray direction computation and enabling direct application of epipolar geometry.

The projection of a 3D point X to an image point u' can be expressed as:

$$u' = \pi(K[R|t]X)$$

where K contains intrinsic parameters, and R, t describe extrinsic parameters.

2.2 Calibration of Omnidirectional Cameras

Omnidirectional imaging systems can be:

- **Catadioptric**: Combining lenses and mirrors (e.g., hyperbolic, parabolic, elliptic) with conventional cameras. Catadioptric systems with hyperbolic mirrors (hyper-catadioptric) work with perspective cameras; those with parabolic mirrors (para-catadioptric) pair with orthographic cameras. Both can satisfy the single-viewpoint property under ideal conditions.
- **Dioptric**: Using wide-angle or fisheye lenses combined with regular cameras. Dioptric systems, however are non-central.

The generalized camera model accommodates both types: Dioptric and Catadioptric. Calibration involves recovering the projection function g mapping image coordinates to 3D ray directions, without assuming a specific mirror or lens geometry. This model can give some information about the mirror of the omnidirectional system and besides it provides a theoretical projection function.

All central catadioptric cameras can be modeled by a unit sphere and a perspective camera, such that the projection of 3D points can be performed in two steps. First, one projects the point onto the sphere, to the intersection of the sphere and the line joining its center and the 3D point. There are two such intersection points, r±. These points are then projected into a normalized plane π resulting in two points, q±. Finally these points are projected into the perspective plane π giving again two points x±, one of which is physically true. Figure 1 shows the projection of a 3D point using the general sphere camera model.

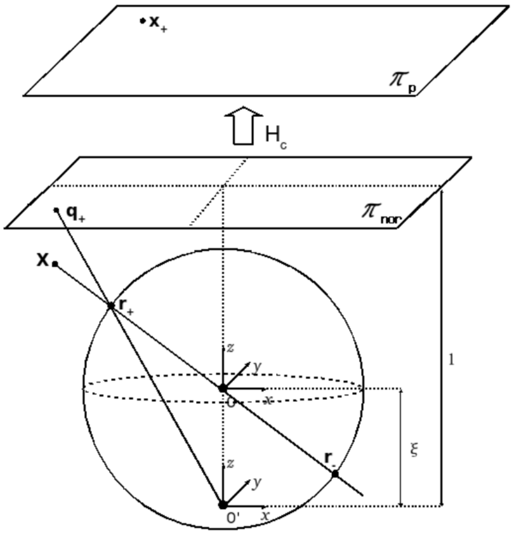

Figure 1: Projection of a 3D point in the image plane using the Sphere Camera Model.(from [3])

The various methods to calibrate omnidirectional cameras require either: i) images of a 2D pattern, ii) images of lines or iii) one image of a 3D pattern. Most of these approaches use the sphere camera model, but there is another approach based on a distortion function. All approaches can deal with any central catadioptric or fish-eye systems and use a non-linear step which allows them to have a reprojection error less than 1 pixel.

Here we will adopt the distortion approach associated with a 2D Pattern (Distortion-2D), described in the next section, because it is very general and practical to implement.

3 Projection-Distortion Function Camera Calibration

3.1 Overview

The Distortion-2D technique by Scaramuzza [6] requires the camera to observe a planar calibration pattern (checkerboard, dot grid) from multiple orientations. No prior knowledge of motion is needed. The projection function is approximated using a Taylor polynomial expansion.

3.2 Generalized Camera Model

We distinguish between:

- Image plane coordinates $U' = [u', v']^T$
- Sensor plane coordinates $U'' = [u'', v'']^T$

They are related by an affine transformation:

$$U'' = AU' + t$$

where $A \in \mathbb{R}^{2\times 2}$ and $t \in \mathbb{R}^2$.

The projection function is:

$$g(u'', v'') = (u'', v'', f(\rho''))^T$$

with rotational symmetry in f, where ρ'' is the radial distance from the sensor axis. The function f is modeled as:

$$f(\rho'') = a_0 + a_1\rho'' + a_2\rho''^2 + \cdots + a_N\rho''^N$$

The calibration goal is to estimate $A, t, a\_0, \ldots, a\_N$, and N such that the reconstructed rays match the projection geometry.

3.3 Single Omnidirectional Camera Calibration

3.3.1 Camera Calibration - Overview

1. Ellipse to Circle Normalization: Transform the outer boundary of the field of view into a centered circle to simplify parameter estimation.

2. Intrinsic Parameter Estimation: Estimate A, t, and polynomial coefficients by minimizing reprojection error.

3. Extrinsic Parameter Estimation: For each image, compute the pose of the calibration pattern relative to the camera.

See Figure 2 (b) and (c).

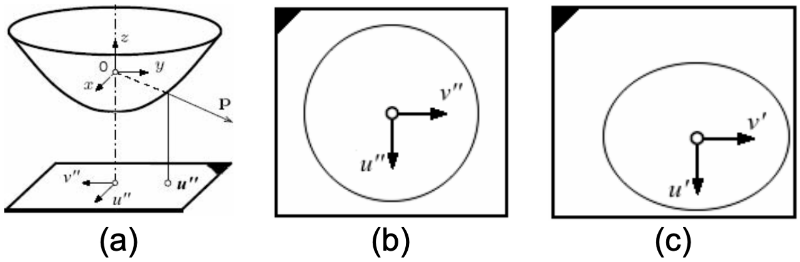

Figure 2: (a) Coordinate system in the catadioptric case. (b) Sensor plane, in metric coordinates. (c) Camera image plane, in pixel coordinates. [from [6]]

3.3.2 Camera Calibration - Derivation

The camera calibration can be derived from this basic equation:

$$\begin{bmatrix} u \\ v \\ f(\rho) \end{bmatrix} \times \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = 0 \tag{1}$$

Equation (1) encodes the **collinearity constraint** that lies at the heart of camera calibration. It states that, for a given observed image point (u, v), the corresponding 3D point in world coordinates — after being transformed into the camera frame — must lie on the ray that passes through the camera center and the observed image point.

Formally, the equation is:

$$\begin{bmatrix} u \\ v \\ f(\rho) \end{bmatrix} \times \left(R \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} + t \right) = 0, \tag{2}$$

where R is the 3×3 rotation matrix and t is the translation vector that together map the world point to camera coordinates. The vector

$$\mathbf{d}(u, v) = \begin{bmatrix} u \\ v \\ f(\rho) \end{bmatrix} \tag{3}$$

represents the **back-projected ray direction**, where $f(\rho)$ corrects for lens distortion as a function of the radial distance $\rho = \sqrt{u^2 + v^2}$.

The cross product being zero means the two vectors are **parallel**:

$$R \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} + t = \lambda \, \mathbf{d}(u, v), \qquad \lambda > 0. \tag{4}$$

This expresses that the transformed point lies exactly along the back-projected ray, with λ representing its depth along that ray.

In geometric terms, Equation (1) enforces that the camera center, the 3D world point, and its measured image projection are **collinear**.

This representation has several advantages:

- It avoids explicit perspective division by depth, making the resulting system of equations **linear** in the unknowns.
- It provides **two independent equations per correspondence**, suitable for linear least-squares solution using methods like SVD.
- It naturally incorporates the effect of radial distortion via $f(\rho)$, allowing its parameters to be estimated iteratively.

Overall, Equation (1) provides a compact and elegant way to relate 3D points to their observed image positions, forming the basis for estimating both extrinsic (pose) and intrinsic (calibration) parameters of the camera during the calibration process.

3.3.3 Camera Calibration - Steps

The calibration method proceeds through eight main steps, designed to progressively estimate and refine the camera parameters. The overall approach moves from a linear initialization to a full non-linear optimization, ensuring both robustness and accuracy. The steps are summarized as follows:

1. **Initialization:** Start with a simple initial guess, typically assuming the intrinsic calibration matrix $A = I$ (no skew, focal length equal to 1, and principal point at the origin) and in-plane translation $t = (0, 0)^\top$. This provides a neutral baseline for the subsequent parameter estimation.
2. **Partial Pose Estimation:** Use Equation (4), which does not involve the radial distortion function $f(\rho)$, to solve for the in-plane rotation elements $r_{11}, r_{12}, r_{21}, r_{22}$ and translations t_1, t_2. This is done via a linear least-squares solution (e.g., using SVD). At this stage, r_{31}, r_{32}, t_3 remain undetermined.
3. **Enforce Orthonormality:** Impose the orthonormality constraints on the first two columns of the rotation matrix to compute the remaining rotation elements r_{31}, r_{32} consistently. This ensures that R is a valid rotation matrix.
4. **Estimate Distortion and Depth Translation:** With r_{31}, r_{32} known, use Equations (2) and (3) to solve for the depth translation t_3 and estimate the radial distortion parameters of $f(\rho)$. This step incorporates the non-linear effects of lens distortion.
5. **Refine Extrinsic Parameters:** Using all three equations (2, 3, and 4), recompute $r_{11}, r_{12}, r_{21}, r_{22}, t_1, t_2, t_3$, now keeping $f(\rho)$ fixed. This provides a consistent estimate of the full camera pose.
6. **Iterative Refinement:** Alternate between solving for the pose (extrinsic parameters) and refining the distortion parameters until convergence. This iterative process reduces projection error progressively.
7. **Re-estimate Principal Point:** Update the assumption on t (principal point) from $(0, 0)^\top$ to an estimated value that better matches the observed image data. Repeat the estimation steps to further improve accuracy.
8. **Final Non-Linear Optimization:** Perform a global non-linear bundle adjustment over all parameters $(R, t, A, f(\rho))$, minimizing the total reprojection error over all point correspondences. This step yields the final calibrated camera parameters.

This staged procedure ensures numerical stability and provides good initial values for the final non-linear optimization, avoiding poor local minima and producing accurate intrinsic, extrinsic, and distortion parameters.

3.3.4 Camera Calibration - Computation Process

First we expand equation (1) int the following:

$$v\,(x\,r_{31} + y\,r_{32} + t_3) - g(\rho)\,(x\,r_{21} + y\,r_{22} + t_2) = 0 \tag{5}$$
$$g(\rho)\,(x\,r_{11} + y\,r_{12} + t_1) - u\,(x\,r_{31} + y\,r_{32} + t_3) = 0 \tag{6}$$
$$u\,(x\,r_{21} + y\,r_{22} + t_2) - v\,(x\,r_{11} + y\,r_{12} + t_1) = 0 \tag{7}$$

Using the above equations, we perform this sequence of computations:

1. Assume $A = I$ and $t = (0, 0)^\intercal$
2. Apply Equation 7 on every point correspondence to create a linear system and use SVD to solve it, estimating r_{11}, r_{12}, r_{21}, r_{22}, t_1 and t_2.

3. Use orthonormality constraint to uniquely compute r_{31} and r_{32} (still can't determine t_3).
4. Apply Equations 5 and 6 on every point correspondence to create a linear system and use pseudo-inverse to solve it, fixing r_{11}, r_{12}, r_{21}, r_{22}, t_1 and t_2 to estimate the distortion parameters of $f(\rho)$ and also t_3.
5. Apply Equations 5, 6 and 7 on every point correspondence to create a linear system and use SVD to solve it, fixing the distortion parameters of $f(\rho)$ to estimate r_{11}, r_{12}, r_{21}, r_{22}, t_1, t_2 and t_3.
6. Apply Equations 5 and 6 on every point correspondence to create a linear system and use pseudo-inverse to solve it, fixing r_{11}, r_{12}, r_{21}, r_{22}, t_1, t_2 and t_3 to estimate the distortion parameters of $f(\rho)$.
7. Repeat steps 1 to 6 while searching for a better image center than the initial guess $t = (0, 0)^\intercal$.
8. Perform a non-linear refinement that bundle adjusts R, T, A, t and the distortion parameters of $f(\rho)$.

4 Omnidirectional Rig Calibration

4.1 Rig Setup

An omnidirectional camera rig consists of multiple individual cameras arranged to collectively capture a full 360° spherical view. Each camera must be rigidly mounted, and the rig must maintain fixed relative poses between cameras. The rig geometry can vary, ranging from simple stereo fisheye pairs to complex arrays combining catadioptric and dioptric units.

4.2 Rig Calibration

The method for rig calibration encompasses two parts:

1. Intrinsic Calibration: Calibrate each camera individually using the method described in Section 3.
2. Extrinsic Calibration: Determine the rigid transformations between the coordinate frames of the individual cameras.
   (a) Capture a calibration pattern visible in overlapping fields of view.
   (b) Detect and match feature points across cameras.
   (c) Solve for relative poses via bundle adjustment, ensuring all cameras are aligned in a common rig coordinate frame.

4.3 Formulation

We define the full calibration problem as the joint estimation of:

- Intrinsic parameters for each camera.
- Extrinsic parameters for each camera relative to the rig frame.
- A unified projection model for all cameras.

4.3.1 Notation

Let:

- C_i be camera i
- K_i intrinsic parameters of C_i
- R_i, t_i extrinsic parameters of C_i in the rig frame
- g_i projection function of C_i

5 Traditional Rig Calibration

5.1 Overview

The traditional framework operates in two main phases:

1. Single-Camera Calibration: Determine K_i and g_i for each C_i.
2. Multi-Camera Optimization: Solve for R_i, t_i jointly, refining K_i and g_i to minimize global reprojection error.

5.2 Steps

1. Pattern Acquisition: Capture calibration pattern images with each camera.
2. Feature Extraction: Detect 2D feature points in each image.
3. Intrinsic Estimation: Apply the generalized model fitting for each camera.
4. Pairwise Alignment: Estimate relative poses for overlapping camera pairs.
5. Global Bundle Adjustment: Optimize all parameters to ensure consistency across the rig.
6. Validation: Test calibration accuracy with known 3D points or scenes.

6 Unified End-to-End Calibration Framework

Here we propose an unified calibration framework that can be applied to all omnidirectional calibration problems, from single camera to rig calibration. Furthermore, it allows the calibration procedure to be performed into different combination of steps, either individual or integrated.

The framework is based on the following model, that extends the basic calibration equation Eq. (1) from a single camera / pattern to many cameras and patterns.

From the image point (u', v'), the sensor point (u, v) is given by $\begin{bmatrix} u \\ v \end{bmatrix} = A_s \begin{bmatrix} u' \\ v' \end{bmatrix} + \boldsymbol{t}_s$, where $A_s \in \mathbb{R}^{2\times 2}$ and $t_s \in \mathbb{R}^2$

$$\text{Sensor Point}\left\{\underbrace{\begin{bmatrix} u \\ v \\ f(\rho) \end{bmatrix}}_{\substack{\text{View}\\\text{Direction}}}\right. \times \left(\underbrace{R_c}_{\substack{\text{Camera}\\\text{Rotation}\\\mathbb{R}^{3\times3}}} \left(\underbrace{R_m}_{\substack{\text{Model}\\\text{Rotation}\\\mathbb{R}^{3\times3}}} \underbrace{\begin{bmatrix} x \\ y \\ 0 \end{bmatrix}}_{\substack{\text{Model}\\\text{Point}}} + \underbrace{\boldsymbol{t}_m}_{\substack{\text{Model}\\\text{Translation}\\\mathbb{R}^3}} \right) + \underbrace{\boldsymbol{t}_c}_{\substack{\text{Camera}\\\text{Translation}\\\mathbb{R}^3}} \right) = 0 \tag{8}$$

where $\rho = \sqrt{u^2 + v^2}$ and $\times$ denotes the cross product.

We will assume $\boldsymbol{t}_c = \begin{bmatrix} 0 & 0 & d \end{bmatrix}^\intercal$ and denote $R_m = \begin{bmatrix} \boldsymbol{r}_1 | \boldsymbol{r}_2 | \boldsymbol{r}_3 \end{bmatrix}$. Expressing the cross product as matrix multiplication we have:

$$\begin{gathered} \begin{bmatrix} 0 & -f(\rho) & v \\ f(\rho) & 0 & -u \\ -v & u & 0 \end{bmatrix} \left(R_c \left(\begin{bmatrix} | & | & | \\ \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} + \boldsymbol{t}_m \right) + \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix} \right) = 0 \\ \therefore \quad \begin{bmatrix} 0 & -f(\rho) & v \\ f(\rho) & 0 & -u \\ -v & u & 0 \end{bmatrix} R_c \Big(x\boldsymbol{r}_1 + y\boldsymbol{r}_2 + \boldsymbol{t}_m \Big) + \begin{bmatrix} vd \\ -ud \\ 0 \end{bmatrix} = 0 \end{gathered} \tag{9}$$

This leads to 3 linear equations over the variables $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{t}_m$:

$$\begin{bmatrix} 0 & -f(\rho) & v \end{bmatrix} R_c \Big(x\boldsymbol{r}_1 + y\boldsymbol{r}_2 + \boldsymbol{t}_m \Big) = -vd \tag{10}$$

$$\begin{bmatrix} f(\rho) & 0 & -u \end{bmatrix} R_c \Big(x\boldsymbol{r}_1 + y\boldsymbol{r}_2 + \boldsymbol{t}_m \Big) = ud \tag{11}$$

$$\begin{bmatrix} -v & u & 0 \end{bmatrix} R_c \Big(x\boldsymbol{r}_1 + y\boldsymbol{r}_2 + \boldsymbol{t}_m \Big) = 0 \tag{12}$$

With an initial guess for R_c, Equation 12 can be applied on every point correspondence $(u, v) \leftrightarrow (x, y, 0)$ to build a linear system independent of $f(\rho)$ and d to compute the values of $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{t}_m$ up to a scale through SVD. Imposing orthonormality constraints over $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ we can recover the correct scale and also determine $\boldsymbol{r}_3$. Later, when an estimate of $f(\rho)$ and d becomes available, all Equations 10, 11 and 12 can be used when building the linear system to compute $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{t}_m$ through the pseudo-inverse.

With the values for $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{t}_m$, we can compute $\begin{bmatrix}x' & y' & z'\end{bmatrix}^\intercal = R_c\Big(x\boldsymbol{r}_1 + y\boldsymbol{r}_2 + \boldsymbol{t}_m\Big)$. Assuming the lens distortion to be an n-th degree polynomial with $f(\rho) = \sum_{i=0}^{n} a_i\rho^i$, Equations 10 and 11 can be respectively rewritten as:

$$\begin{bmatrix}0 & -\sum_{i=0}^{n} a_i\rho^i & v\end{bmatrix}\begin{bmatrix}x' \\ y' \\ z'\end{bmatrix} = -vd \quad \therefore \quad vd - y'\sum_{i=0}^{n} a_i\rho^i = -vz' \tag{13}$$

$$\begin{bmatrix}\sum_{i=0}^{n} a_i\rho^i & 0 & -u\end{bmatrix}\begin{bmatrix}x' \\ y' \\ z'\end{bmatrix} = ud \quad \therefore \quad -ud + x'\sum_{i=0}^{n} a_i\rho^i = uz' \tag{14}$$

Equations 13 and 14 can be used to build a linear system over the variables d, a_0, a_1, a_2, ..., a_n which can be solved through the pseudo-inverse.

To summarize the complete process for a collection of cameras and models:

1. **For each camera:** Assume an initial guess for A_s, $\boldsymbol{t}_s$, R_c.
2. **For each model (with its viewing cameras):** Build a linear system from Equation 12 to compute $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{t}_m$ through SVD while imposing orthonormality constraints to determine the correct scale.
3. **For each camera (with its visible models):** Build a linear system from Equations 13 and 14 to compute d, a_0, a_1, a_2, ..., a_n through the pseudo-inverse.
4. **For each model (with its viewing cameras):** Build a linear system from Equations 10, 11 and 12 to compute $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{t}_m$ through the pseudo-inverse.
5. Perform a non-linear refinement to bundle adjust A_s, $\boldsymbol{t}_s$, R_c, $\boldsymbol{t}_c$, R_m, $\boldsymbol{t}_m$ and the distortion parameters a_0, a_1, a_2, ..., a_n.

Note: step 3 can be performed individually for each camera as described or, alternatively, unified in a single linear system to compute parameters d, a_0, a_1, a_2, ..., a_n that are shared among all cameras.

References

[1] Jakarto. py-ocamcalib. https://github.com/jakarto3d/py-OCamCalib. Accessed: 2025-08-08.

[2] Christopher Mei and Patrick Rives. Single view point omnidirectional camera calibration from planar grids. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3945–3950, 2007.

[3] Luis Puig, J. Bermudez, Peter Sturm, and J. J. Guerrero. Calibration of omnidirectional cameras in practice: A comparison of methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 414–421, 2011.

[4] Rameau. Mc-calib. https://github.com/rameau-fr/MC-Calib. Accessed: 2025-08-08.

[5] François Rameau, Jinsun Park, Oleksandr Bailo, and In Kweon. Mc-calib: A generic and robust calibration toolbox for multi-camera systems. *Computer Vision and Image Understanding*, 217:103353, 01 2022.

[6] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A flexible technique for accurate omnidirectional camera calibration from structure and motion. In *Proceedings of the Fourth IEEE International Conference on Computer Vision Systems (ICVS)*, pages 45–45, 2006.

[7] Steffen Urban, Jens Leitloff, and Stefan Hinz. Improved wide-angle, fisheye and omnidirectional camera calibration. *ISPRS Journal of Photogrammetry and Remote Sensing*, 108:72–79, 2015.

[8] Davide Scaramuzza et al. Vincente. Ocamcalib: Omnidirectional camera calibration toolbox for matlab. https://sites.google.com/site/scarabotix/ocamcalib-omnidirectional-camera-calibration-toolbox-for-matlab. Accessed: 2025-08-08.

[9] Yan Zhang, Lina Zhao, and Wanbao Hu. A survey of catadioptric omnidirectional camera calibration. *International Journal of Information Technology and Computer Science (IJITCS)*, 5(3):13–20, 2013.