

Auto-Calibração de Câmeras em Visão Estéreo

Israel de Moraes Madalena, David Menotti
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFOP - Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brasil
email: israelmorais@yahoo.com.br, menottid@gmail.com

Resumo—Este trabalho visa o estudo de métodos de calibração automática de uma única câmera e de câmeras utilizados em um sistema de Visão Estéreo. Esta auto-calibração tem a finalidade de reconstruir tridimensionalmente uma cena. A calibração de câmeras é necessária para se obter parâmetros que tornarão possível mapear coordenadas tridimensionais do mundo a partir de coordenadas bidimensionais de imagens. O primeiro método a ser estudado, o de uma única câmera, visa obter os parâmetros intrínsecos de uma câmera. Já o segundo tem como objetivo a obtenção de parâmetros extrínsecos de duas câmeras, o que permitirá a construção de um sistema de Visão Estéreo. Uma breve análise de complexidade de tempo é realizada sobre os dois métodos de calibração estudados, e também é exemplificada a forma de aplicação desses métodos.

Keywords-Auto-calibração, Visão Estéreo, Multi-Câmeras.

I. INTRODUÇÃO

Visão estereo é o ramo da visão computacional que analisa o problema da reconstrução da informação tridimensional de objetos a partir de um par de imagens capturadas simultaneamente, mas com um pequeno deslocamento [1]. O primeiro passo da visão estereo é extrair características das imagens observadas por um par de câmeras e então, efetuar a fusão binocular das características observadas pelas duas câmeras. Após isso, utiliza-se a informação do estágio anterior juntamente com os dados geométricos do sistema estereo para recuperar a posição tridimensional do ponto no mundo, ou seja, a reconstrução da imagem 3D.

A visão estereo é uma das principais informações de profundidade na visão do ser humano. Ela possui este nome por precisar do uso de ambos os olhos para formar tridimensionalmente a imagem. O olho esquerdo e o olho direito sempre vêem imagens diferentes, apesar de muito parecidas. A partir dessas diferenças, a noção de profundidade é construída pelo cérebro, levando ao entendimento do volume do que está sendo visto. Como exemplo, tem-se a Figura 1, onde podem-se fixar o olho direito na imagem da direita e o olho esquerdo na imagem da esquerda. Dessa forma, o nosso cérebro gera uma imagem tridimensional, passando-nos uma ideia de profundidade.

A partir de duas câmeras posicionadas de forma correta, com suas posições e direcionamentos conhecidos, é possível determinar a posição de qualquer ponto neste espaço, desde que este ponto possa ser localizado dentro de cada uma

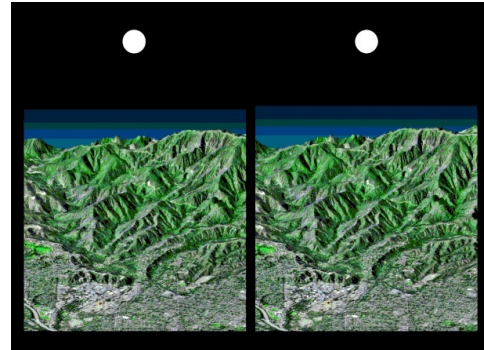


Figura 1: Visão Estéreo

das imagens capturadas pelas câmeras. Por isso, torna-se necessário que haja uma região comum, de proporção razoável, aparecendo em ambas imagens. A determinação da posição de um ponto dentro deste espaço pode ser obtida por triangulação. *Stereopsis* é definido, em [1], como a capacidade de determinação de profundidade baseada na informação de um par estereo.

A Figura 2 mostra um par de imagens estereo. Já a Figura 3 exibe a triangulação utilizada para reconstruir geometricamente a imagem.



Figura 2: Par de Imagens Estéreo

Este sistema envolvendo múltiplas câmeras com sobreposição na área de observação está sendo utilizado em diversas aplicações de monitoramento de ambientes devido a sua confiabilidade e exatidão na contagem e identificação de objetos, além do barateamento de câmeras digitais e de toda a tecnologia que envolve a área.

Ao inicializar tais sistemas, faz-se necessário a calibração das câmeras para reconstruções precisas, visando por

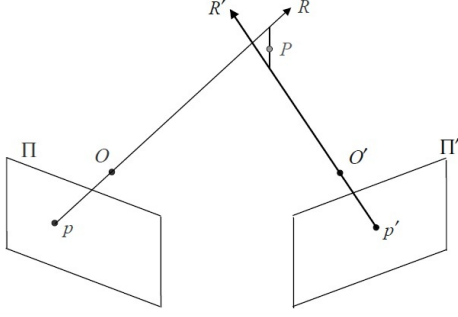


Figura 3: Triangulação

exemplo, a contagem de objetos. Na auto-calibração, não são usados objetos conhecidos previamente, pois é realizada apenas capturando-se diferentes imagens com uma câmera, de uma cena estática, sem alteração dos parâmetros intrínsecos dessa câmera. Após a captura, é feita a correspondência entre pontos presentes nestas imagens.

O restante do artigo está organizado da seguinte forma. A Seção II apresenta os parâmetros necessários para a calibração de câmeras. Os métodos estudados neste trabalho são descritos na Seção III, e suas análise de complexidades são apresentadas na Seção IV. Os experimentos realizados para ilustrar o funcionamento dos métodos de calibração são descritos na Seção V, e finalmente, na Seção VI, as conclusões e trabalhos futuros são apontados.

II. PARÂMETROS DE CALIBRAÇÃO DE CÂMERAS

[2] lista uma sequência de parâmetros internos de uma câmera que são definidos como parâmetros intrínsecos, dentre eles:

- Distância focal: A distância focal em pixels.
- Ponto principal: coordenadas do ponto principal.
- Coeficiente de inclinação: define o ângulo entre os eixos x e y do pixel.
- Distorções: coeficientes de distorção de imagem (radial e tangencial)

A. Parâmetros intrínsecos

Seja P um ponto no espaço de coordenadas da matriz que representa uma imagem de referência da câmera. Considerando ainda que o ponto do plano imagem de acordo com os parâmetros intrínsecos (fc, cc, α_c, kc), tem-se uma projeção da imagem normalizada (x_n), *i.e.*,

$$x_d = \begin{bmatrix} x_d(1) \\ x_d(2) \end{bmatrix} = (1 + kc(1)r^2 + kc(2)r^4 + kc(5)r^6)x_n + dx \quad (1)$$

onde

$$dx = \begin{bmatrix} 2kc(3)xy + kc(4)(r^2 + 2x^2) \\ kc(3)(r^2 + 2y^2) + 2kc(4)xy \end{bmatrix} \quad (2)$$

é a matriz de distorção tangencial.

Uma vez que a distorção é aplicada, o pixel de coordenadas $x_{pixel} = [x_p, y_p]$ da projeção de P no plano imagem é

$$\begin{cases} x_p = fc(1)[x_d(1) + \alpha_c \times x_d(2)] + cc(1) \\ y_p = fc(2)x_d(2) + cc(2) \end{cases} \quad (3)$$

e o pixel de coordenadas do vetor x_{pixel} que é relacionado com o vetor x_d , *i.e.*,

$$KK = \begin{bmatrix} fc(1) & \alpha_c \times fc(1) & cc(1) \\ 0 & fc(2) & cc(2) \\ 0 & 0 & 1 \end{bmatrix}. \quad (4)$$

O Programa 1 mostra a implementação da normalização, utilizada para se fazer o mapeamento inverso da imagem.

```
function [xn] = normalize(x_kk,fc,cc,kc,alpha_c)

if nargin < 5,
    alpha_c = 0;
    if nargin < 4;
        kc = [0;0;0;0;0];
        if nargin < 3;
            cc = [0;0];
            if nargin < 2,
                fc = [1;1];
            end;
        end;
    end;
end;

x_distort = [(x_kk(1,:) - cc(1))/fc(1);(x_kk(2,:) - cc(2))/fc(2)];

x_distort(1,:) = x_distort(1,:) - alpha_c * x_distort(2,:);

if norm(kc) ~= 0,
    xn = comp_distortion_oulu(x_distort,kc);
else
    xn = x_distort;
end;
```

Programa 1: Normalize

B. Parâmetros extrínsecos

Os parâmetros extrínsecos fornecem a posição da origem do sistema de coordenadas da câmera em relação à origem das coordenadas do mundo. Os parâmetros extrínsecos são:

- Rotação: Um conjunto de matrizes de rotação 3x3.
- Translação: Um conjunto de vetores 3x1.

Para calibração dos parâmetros extrínsecos, pode-se considerar um *grid* de calibração como o apresentado na Figura 4, onde são montados os planos de coordenadas de referência $O(X, Y, Z)$.

Então, seja P um ponto de coordenadas espaço vetorial $XX = [X, Y, Z]$ no quadro de referência e $XX_c = [X_c, Y_c, Z_c]$ o vetor de coordenadas de P no referencial da câmera, tem-se que XX e XX_c serão relacionados uns aos outros por meio da equação $XX_c = Rc_l * XX + Tc_l$.

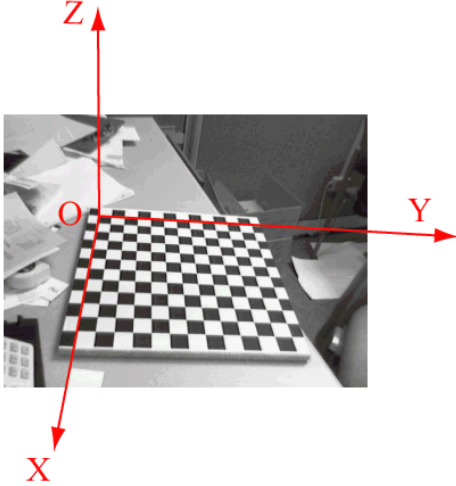


Figura 4: *Grid* de Calibração

O vetor Tc_l é o vetor de coordenadas da origem do padrão do *grid* (O) no quadro de referência. A terceira coluna da matriz Rc_l é o vetor normal à superfície do plano que contém a grade planar. Esta mesma relação vale para todos os parâmetros extrínsecos.

Uma vez que as coordenadas de um ponto são expressas no *grid* de referência da câmera, o ponto pode ser projetado sobre o plano da imagem usando os parâmetros intrínsecos da câmera.

III. MÉTODOS

Nesta seção, dois métodos de calibração de câmeras são descritos.

A. Calibração de uma Câmera

O primeiro método a ser analisado é o proposto por [3], [4], que propõe a calibração de uma câmera usando a observação de um padrão planar mostrado em orientações diferentes. Pode-se escolher a movimentação da câmera ou do padrão planar, o qual não precisa ser reconhecido. O procedimento proposto consiste em uma solução de forma fechada, seguida de um refinamento não-linear com base no critério de máxima verossimilhança. Este método proposto por [3], [4] determina a calibração de uma câmera utilizando-se de um tabuleiro de xadrez, onde deve-se identificar seus cantos em uma imagem e calcula-se toda a divisão do tabuleiro. Caso haja algum valor errado, ou canto mal detectado, insere-se no parâmetro de calibração o erro encontrado e uma nova calibração é efetuada. Dessa forma, o programa de calibração visa a representatividade de uma imagem 3D em uma imagem 2D, sendo que a imagem 3D é identificada por $M = [X, Y, Z]$ e a imagem 2D é denotada por $m = [u, v]$. A câmera é representada por

$$s\tilde{m} = A \begin{bmatrix} R & t \end{bmatrix} \tilde{M}, \quad (5)$$

onde os valores R e t representam os parâmetros extrínsecos do sistema de coordenadas da câmera, e A representa a matrix intrínseca da câmera

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (6)$$

onde (u_0, v_0) são as coordenadas do ponto principal e α e β são os fatores escalares na imagem sobre os eixos u e v . γ representa o parâmetro de assimetria dos dois eixos. A Figura 5 exibe um conjunto de imagens que podem ser utilizadas neste processo de calibração.

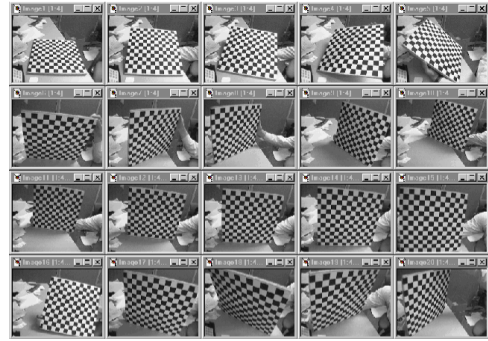


Figura 5: Imagens planares

O problema colocado por Zhang [3], [4] solicita somente a identificação por parte do usuário do erro visualizado. O programa realiza também a contagem automática dos quadrados, não sendo necessário a inserção pelo usuário da quantidade dos mesmos. A etapa de calibração consiste em dois passos, sendo a inicialização e a otimização não-linear.

A inicialização calcula uma solução de forma fechada para os parâmetros de calibração, não incluindo a distorção da lente, o que nos mostra o Programa 2.

```

if ~exist('two_focals_init'),
    two_focals_init = 0;
end;

if ~exist('est_aspect_ratio'),
    est_aspect_ratio = 1;
end;

check_active_images;

if ~exist(['x_' num2str(ind_active(1)) ]),
    click_calib;
end;

fprintf(1, '\nInitialization of the intrinsic
parameters - Number of images: %d\n', length(
ind_active));

for kk = 1:n_ima,
    eval(['x_kk = x_' num2str(kk) '']);
    eval(['X_kk = X_' num2str(kk) '']);
    if (isnan(x_kk(1,1))),

```

```

    if active_images(kk),
        fprintf(1, 'WARNING: Cannot calibrate
            with image %d. Need to extract
            grid corners first.\n', kk)
        fprintf(1, '          Set active_images
            (%d)=1; and run Extract grid
            corners.\n', kk)
    end;
    active_images(kk) = 0;
end;
if active_images(kk),
    eval([ 'H_' num2str(kk) ' =
        compute_homography(x_kk, X_kk(1:2,:)); '
    ]);
else
    eval([ 'H_' num2str(kk) ' = NaN*ones(3,3); '
    ]);
end;
end;

check_active_images;

% initial guess for principal point and distortion
:

if ~exist('nx'), [ny, nx] = size(I); end;

c_init = [nx; ny]/2 - 0.5; % initialize at the
    center of the image
k_init = [0; 0; 0; 0; 0]; % initialize to zero (no
    distortion)

A = [];
b = [];

% matrix that subtract the principal point:
Sub_cc = [1 0 -c_init(1); 0 1 -c_init(2); 0 0 1];

for kk=1:n_ima,

    if active_images(kk),

        eval([ 'Hkk = H_' num2str(kk) ' ; ' ]);

        Hkk = Sub_cc * Hkk;

        % Extract vanishing points (direct and
            diagonals):

        V_hori_pix = Hkk(:, 1);
        V_vert_pix = Hkk(:, 2);
        V_diag1_pix = (Hkk(:, 1)+Hkk(:, 2))/2;
        V_diag2_pix = (Hkk(:, 1)-Hkk(:, 2))/2;

        V_hori_pix = V_hori_pix/norm(V_hori_pix);
        V_vert_pix = V_vert_pix/norm(V_vert_pix);
        V_diag1_pix = V_diag1_pix/norm(V_diag1_pix
        );
        V_diag2_pix = V_diag2_pix/norm(V_diag2_pix
        );

        a1 = V_hori_pix(1);
        b1 = V_hori_pix(2);
        c1 = V_hori_pix(3);

        a2 = V_vert_pix(1);
        b2 = V_vert_pix(2);
        c2 = V_vert_pix(3);

        a3 = V_diag1_pix(1);
        b3 = V_diag1_pix(2);

        c3 = V_diag1_pix(3);

        a4 = V_diag2_pix(1);
        b4 = V_diag2_pix(2);
        c4 = V_diag2_pix(3);

        A_kk = [a1*a2 b1*b2;
            a3*a4 b3*b4];

        b_kk = -[c1*c2; c3*c4];

        A = [A; A_kk];
        b = [b; b_kk];

    end;

end;

% use all the vanishing points to estimate focal
    length:

% Select the model for the focal. (solution to
    Gerd's problem)
if ~two_focals_init
    if b'*(sum(A')) < 0,
        two_focals_init = 1;
    end;
end;

if two_focals_init
    % Use a two focals estimate:
    f_init = sqrt(abs(1./(inv(A'*A)*A'*b))); % if
        using a two-focal model for initial guess
else
    % Use a single focal estimate:
    f_init = sqrt(b'*(sum(A')) / (b'*b)) * ones
        (2,1); % if single focal length model is
        used
end;

if ~est_aspect_ratio,
    f_init(1) = (f_init(1)+f_init(2))/2;
    f_init(2) = f_init(1);
end;

alpha_init = 0;

% f_init = sqrt(b'*(sum(A')) / (b'*b)) * ones(2,1)
    ; % if single focal length model is used

% Global calibration matrix (initial guess):

KK = [f_init(1) alpha_init*f_init(1) c_init(1); 0
    f_init(2) c_init(2); 0 0 1];
inv_KK = inv(KK);

cc = c_init;
fc = f_init;
kc = k_init;
alpha_c = alpha_init;

fprintf(1, '\n\nCalibration parameters after

```

```

145      initialization:\n\n');
      fprintf(1, 'Focal Length:          fc = [ %3.5f
      %3.5f ]\n', fc);
      fprintf(1, 'Principal point:        cc = [ %3.5f
      %3.5f ]\n', cc);
      fprintf(1, 'Skew:                alpha_c = [ %3.5f ]
      => angle of pixel = %3.5f degrees\n', alpha_c
      , 90 - atan(alpha_c)*180/pi);
      fprintf(1, 'Distortion:          kc = [ %3.5f
      %3.5f %3.5f %3.5f %5.5f ]\n', kc);

```

Programa 2: Inicializa a Calibração dos Parâmetros

A etapa de otimização não-linear minimiza o erro total, no sentido dos mínimos quadrados, sobre todos os parâmetros de calibração. A otimização é feita usando o cálculo do gradiente sob matriz Jacobiana, o que pode-se ver em no Programa 3.

```

5      if ~exist('n_ima'),
          data_calib; % Load the images
          click_calib; % Extract the corners
      end;

      check_active_images;
      check_extracted_images;
      check_active_images;
10     deactivated_images = [];

      recompute_extrinsic = (length(ind_active) < 100);
      % if there are too many images, do not spend
      % time recomputing the extrinsic parameters
      % twice..

      if ~exist('rosette_calibration', 'var')
15         rosette_calibration = 0;
      end;

      if (rosette_calibration)
          est_dist = ones(5,1);
20     end;

      go_calib_optim_iter;

      if ~isempty(deactivated_images),
          param_list_save = param_list;
          fprintf(1, '\nNew optimization including the
          images that have been deactivated during
          the previous optimization.\n');
          active_images(deactivated_images) = ones(1,
          length(deactivated_images));
          deactivated_images = [];
          go_calib_optim_iter;
25     if ~isempty(deactivated_images),
          fprintf(1, ['List of images left deactivated
          : ' num2str(deactivated_images) '\n' ]
          );
          end;
          param_list = [param_list_save(:,1:end-1)
          param_list];
30     end;

```

Programa 3: Otimização a não-linear

B. Calibração de Múltiplas Câmeras

O modelo proposto por Svoboda *et al.* [5] é a calibração de um sistema de visão estéreo, usando a adequação

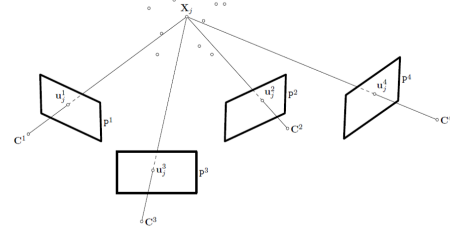


Figura 6: Sistema Multi-Câmera

dos parâmetros intrínsecos e extrínsecos e utilização destes resultados para retificação da imagem e triangulação estéreo 3D. [5] utiliza o princípio da detecção de um ponto comum em ambas imagens para então extrair os parâmetros que serão avaliados e comparados, originando os valores para a calibração dos parâmetros da câmera. Neste modelo, pode-se utilizar um ponto brilhante ou um ponto em que se destaque e esteja presente em todas as câmeras, para que, a imagem seja reconstruída. A Figura 6 exibe um sistema multi-camera e uma imagem, a qual o ponto X_j representa o ponto na cena que tem que ser comum a todas as câmeras. A quantidade de câmeras no sistema de visão estéreo depende na representatividade que se quer obter da cena. Por exemplo, se deseja obter todos os pontos do corpo humano, quanto maior o número de câmeras utilizadas, mais preciso será a representação requerida.

Para utilizar o método proposto por Svoboda *et al.* [5], é necessário somente a matriz de dados W contendo os pontos da imagem. Essa matriz pode conter alguns pontos diferentes, porém, quanto mais completa, mais precisa e estável será a calibração. Após a obtenção dessa matriz, procura-se os pontos semelhantes nas outras matrizes, o que é chamado de correspondência.

Então, este modelo pode ser executado nos seguintes passos:

- Encontrar o ponto comum nas imagens.
- Descartar pontos detectados incorretamente.
- Estimar os pontos perdidos.
- Otimizar a estrutura se necessário.
- Realizar a classificação e fatoração da matriz.
- Atualizar as matrizes usando o método euclidiano.
- Detectar pontos incorretos e realizar a correção destes.
- Estimar os parâmetros de distorção não-linear

Na Figura 7, ilustra-se o processo de calibração de 2 câmeras utilizando-se o método proposto em [5].

IV. ANÁLISE DE COMPLEXIDADE

Nos métodos [3], [4] e [5] que foram analisados, a complexidade varia conforme o número de imagens existentes

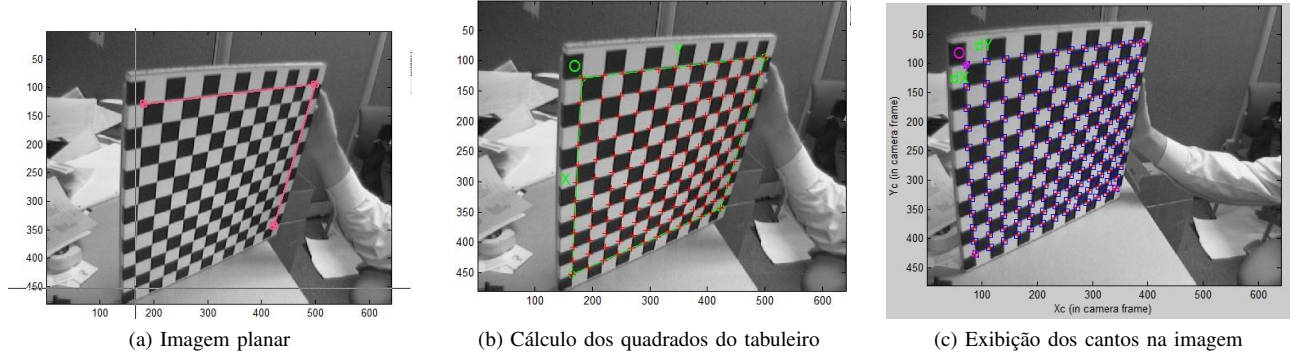


Figura 8: Calibração da Câmera - Sequência de Imagens

```
Focal Length:      fc = [ 630.45285   626.06099 ] ± [ 4.50475   4.14859 ]
Principal point:   cc = [ 298.91927   237.64385 ] ± [ 7.60724   7.33838 ]
Skew:              alpha_c = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:        kc = [ -0.27605   0.13082   -0.00295   0.00704   0.00000 ] ± [ 0.03050   0.12628   0.00141   0.00197   0.00000 ]
Pixel error:       err = [ 0.71770   0.68640 ]
```

(a) Parâmetros intrínsecos de uma única câmera

```
Intrinsic parameters of left camera:

Focal Length:      fc_left = [ 533.00371   533.15260 ] ± [ 1.07629   1.10913 ]
Principal point:   cc_left = [ 341.58612   234.25940 ] ± [ 1.24041   1.33065 ]
Skew:              alpha_c_left = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:        kc_left = [ -0.28947   0.10326   0.00103   -0.00029   0.00000 ] ± [ 0.00596   0.02055   0.00030   0.00037   0.00000 ]

Intrinsic parameters of right camera:

Focal Length:      fc_right = [ 536.98262   536.56938 ] ± [ 1.19786   1.15677 ]
Principal point:   cc_right = [ 326.47209   249.33257 ] ± [ 1.36588   1.34252 ]
Skew:              alpha_c_right = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:        kc_right = [ -0.28936   0.10677   -0.00078   0.00020   0.00000 ] ± [ 0.00488   0.00866   0.00027   0.00062   0.00000 ]

Extrinsic parameters (position of right camera wrt left camera):

Rotation vector:   om = [ 0.00611   0.00409   -0.00359 ]
Translation vector: T = [ -99.84929   0.82221   0.43647 ]
```

(b) Antes da calibração

```
Intrinsic parameters of left camera:

Focal Length:      fc_left = [ 533.52331   533.52700 ] ± [ 0.83147   0.84055 ]
Principal point:   cc_left = [ 341.60377   235.19287 ] ± [ 1.23937   1.20470 ]
Skew:              alpha_c_left = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:        kc_left = [ -0.28838   0.09714   0.00109   -0.00030   0.00000 ] ± [ 0.00621   0.02155   0.00028   0.00034   0.00000 ]

Intrinsic parameters of right camera:

Focal Length:      fc_right = [ 536.81376   536.47649 ] ± [ 0.87631   0.86541 ]
Principal point:   cc_right = [ 326.28655   250.10121 ] ± [ 1.31444   1.16609 ]
Skew:              alpha_c_right = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:        kc_right = [ -0.28943   0.10690   -0.00059   0.00014   0.00000 ] ± [ 0.00486   0.00883   0.00022   0.00055   0.00000 ]

Extrinsic parameters (position of right camera wrt left camera):

Rotation vector:   om = [ 0.00669   0.00452   -0.00350 ] ± [ 0.00270   0.00308   0.00029 ]
Translation vector: T = [ -99.80198   1.12443   0.05041 ] ± [ 0.14200   0.11352   0.49773 ]
```

(c) Após a calibração

Figura 9: Parâmetros obtidos no processo de calibração

para processamento.

O método proposto em [3], [4] possui uma complexidade de tempo de execução de $f(n) = 2O(n)$, onde tem-se duas estruturas de repetição `for`, sendo uma para a identificação dos cantos da imagem e outra para o cálculo dos parâmetros

intrínsecos e extrínsecos.

Já o modelo proposto em [5] possui uma complexidade de tempo $f(n) = 100O(n^2)$, sendo que a constante 100 é o número máximo de iterações que o método é executado, lembrando que é feito o cálculo do gradiente juntamente

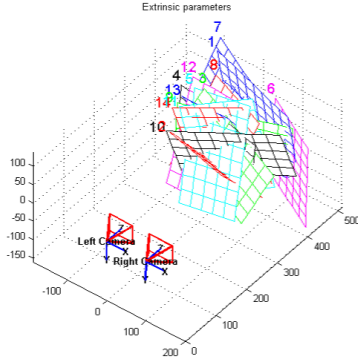


Figura 7: Visão Multi-Câmera

com as matrizes jacobianas. Dentro deste *loop* também há 2 estruturas *for*, as quais são responsáveis pelo cálculo dos parâmetros intrínsecos e extrínsecos respectivamente.

V. EXPERIMENTOS

Para o método de [3], [4] foram utilizadas 20 imagens contendo o tabuleiro de xadrez visto de diversas perspectivas e posições. Para a efetividade do método, identifica-se os 4 cantos do tabuleiro clicando-se com o *mouse* sobre a imagem. Após isso, o programa conta automaticamente os quadrados existentes no tabuleiro e traça a perspectiva dos demais para que o usuário veja se está corretamente alinhado ou não. Após a análise do usuário, aceita-se o resultado exibido ou não. Caso seja reprovado, o usuário pode inserir em milímetros a proporção de distorção da imagem.

A seguir tem-se uma sequência de imagens que compõem a execução do programa para calibração da câmera. Na Figura 8a, observa-se a imagem preparada para a identificação dos cantos. Já a Figura 8b exibe a projeção dos demais pontos. A Figura 8c exibe a consolidação dos cantos na imagem após n imagens processadas, enquanto a Figura 10 ilustra a câmera fixa e as projeções das n imagens. Concluindo, a Figura 11 exibe os erros encontrados para análise do usuário. Os valores obtidos para a calibração intrínseca da câmera podem ser vistos na Figura 9a.

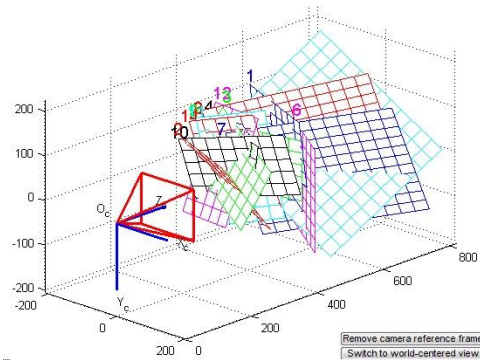


Figura 10: Visão da câmera

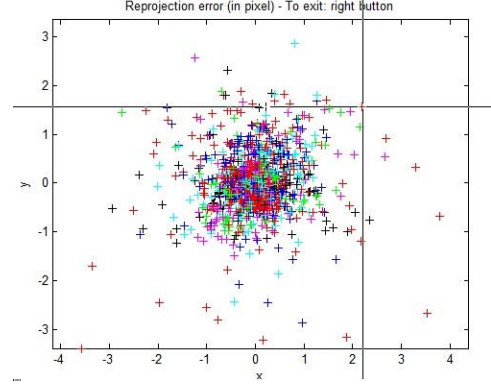


Figura 11: Gráfico de erros

Para o modelo proposto em [5], a realização dos experimentos e testes dependem dos resultados previamente calculados, podendo-se aproveitar dos resultados obtidos do método de [3], [4]. Portanto, como primeiro passo neste experimento, deve-se carregar os parâmetros extrínsecos e intrínsecos previamente, o que pode-se observar na Figura 9b. Após isso, o programa calcula automaticamente usando 14 imagens da câmera direita e 14 da câmera esquerda os novos parâmetros de calibração extrínsecos e intrínsecos. Na Figura 9c, visualiza-se o resultado obtido para a calibração na visão estereó e na Figura 12 ilustra-se se os parâmetros extrínsecos originados de cada imagem na calibração.

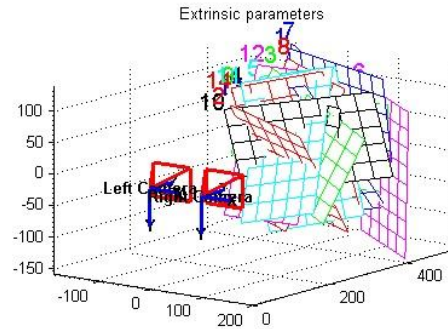


Figura 12: Parâmetros extrínsecos

Após a execução desses passos, têm-se os parâmetros de calibração das câmeras direita e esquerda, possibilitando a refatoração das imagens, dando ao cenário uma visão 3D.

VI. CONCLUSÕES

Visão Estéreo é um campo dentro da área de visão computacional importante para a representação tridimensional de uma cena. Neste sentido, torna-se importante a precisão e confiabilidade do sistema.

Esta confiabilidade e precisão são melhoradas por meio de calibração das câmeras, reduzindo assim o percentual de erro do resultado obtido. Apresentou-se os métodos de Zhang [3], [4] e Svoboda *et al.* [5] para calibração de câmeras, os quais

diferem no modo de calibração e na quantidade de câmeras utilizadas.

O método proposto por Zhang [3], [4] efetua a calibração individual da câmera usando a detecção de cantos em uma imagem planar. Utilizou-se para exemplificar o uso deste método imagens de um tabuleiro de xadrez.

Svoboda *et al.* [5] propôs a calibração de sistemas multi-câmeras usando a detecção de um ponto principal em uma sequência de imagens, onde este ponto tem que ser comum em todas as imagens.

Outra diferença entre os dois métodos é o tempo de execução, sendo que o método baseado na detecção de cantos [3], [4] executa com um tempo de $2O(n)$ e o proposto por Svoboda *et al.* [5] executa em $100O(n^2)$.

Percebe-se que apesar de ser mais rápido, o método de calibração utilizando a detecção de cantos pode ser falho ao ponto do usuário ter que interagir com o sistema mostrando os cantos principais e inserindo alguns parâmetros. Já o método de Svoboda *et al.* [5] utiliza dados previamente carregados e efetua a calibração automática dos parâmetros extrínsecos e intrínsecos da câmera.

REFERÊNCIAS

- [1] E. Trucco and A. Verri, *Introductory Techniques for 3D Computer Vision*, 1st ed. Prentice Hall, 1998, ISBN: 978-0132611084.
- [2] H. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 1997, pp. 1106–1112.
- [3] Z. Zhang, "A flexible new technique for camera calibration," Microsoft Research, Microsoft Corporation, Technical Report MSR-TR-98-71, 1998. [Online]. Available: <http://research.microsoft.com/~zhang>
- [4] —, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.
- [5] T. Svoboda, D. Martinec, and T. Padjla, "A convenient multi-camera self-calibration for virtual environments," Center for Machine Perception, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, Technical Report, 2005. [Online]. Available: <ftp://cmp.felk.cvut.cz/pub/cmp/articles/svoboda/svobodaPRESENCE2005.pdf>