Noname manuscript No.
(will be inserted by the editor)

Implementation of the Generic Equation (25) on the Journal Paper *"Use of Flip Ambiguity Probabilities in Robust Sensor Network Localization"*

Anushiya A Kannan · Barış Fidan · Guoqiang Mao

Received: date / Accepted: date

This note describes the implementation methodology of Equation (25) of the Jounal Paper "Use of Flip Ambiguity Probabilities in Robust Sensor Network Localization". Sample values of each of $\overline{d}_{AD}$, $\overline{d}_{BD}$ and $\overline{d}_{CD}$ are taken to span the distance range $[0, R]$ at a step size of R/N. Integration is implemented using the MATLAB function *trapz*. In choice of N there is a trade-off between accuracy and computational cost. In our case, N was chosen to be 100.

```
function NumericalCalculationOfFlip(xy,R,N,sigma)
    %xy - true coordinates of A,B,C and D
    %transmission range R
    %Number of steps N
    %Standard deviation of Gaussian noise sigma

    % create distance matrix containing possible values
    % x axis representing measured distance dAD,
    % y axis representing measured distance dBD, and
    % z axis representing measured distance dCD
    Delta=R/N;
    d=Delta:Delta:R;
    for i=1:N,
```

The work was supported by National ICT Australia-NICTA. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

Anushiya A Kannan
School of Electrical and Information Engineering, University of Sydney, NSW 2006, Australia.
Tel.: +61-2-88122431
Fax: +61-2-88122431
E-mail: anushiya@ee.usyd.edu.au

Barış Fidan
University of Waterloo, Ontario, Canada
E-mail: fidan@uwaterloo.ca

Guoqiang Mao
The University of Sydney and National ICT Australia
E-mail: Guoqiang.Mao@nicta.com.au

```
        dAD(:,:,i)=ones(N,1)*d;
        dBD(:,:,i)=d'*ones(1,N);
        dCD(:,:,i)=i*Delta*ones(N,N);
    end

    %Create Identity Matrices
    I_zeta_AB=Calculate_I_zeta_XY(dAD,dBD,xy,R);

    xy_changedOrder=[xy(2,:);xy(3,:);xy(1,:);xy(4,:)];
    I_zeta_BC=Calculate_I_zeta_XY(dBD,dCD,xy_changedOrder,R);

    xy_changedOrder=[xy(3,:);xy(1,:);xy(2,:);xy(4,:)];
    I_zeta_AC=Calculate_I_zeta_XY(dAD,dCD,xy_changedOrder,R)
    % Define normalised Gaussian transfer functions
    measured_dCD=shiftdim((dCD(1,1,:)),2)';
    CD=norm(xy(3,:)-xy(4,:));
    TF=1/(sigma*sqrt(2*pi))*exp(-(dCD-CD).^2/(2*sigma^2));
    NF=cdf('normal',R,CD,sigma)-cdf('normal',0,CD,sigma);
    YCD=TF/NF; clear TF NF

    measured_dBD=dBD(:,1,1)';
    BD=norm(xy(2,:)-xy(4,:));
    TF=1/(sigma*sqrt(2*pi))*exp(-(dBD(:,:,1)-BD).^2/(2*sigma^2));
    NF=cdf('normal',R,BD,sigma)-cdf('normal',0,BD,sigma);
    YBD=TF/NF;

    measured_dAD=dAD(1,:,1);
    AD=norm(xy(1,:)-xy(4,:));
    TF=1/(sigma*sqrt(2*pi))*exp(-(measured_dAD-AD).^2/(2*sigma^2));
    NF=cdf('normal',R,AD,sigma)-cdf('normal',0,AD,sigma);
    YAD=TF/NF;

    % Calculate the analytical probability P(zeta_AB_BC_AC_AD/ABCD)
    Y=YCD.*I_AB.*I_BC.*I_AC;
    for i=1:N,
        for j=1:N,
            YC(i,j)=trapz(measured_dCD,shiftdim((Y(i,j,:)),2)');
        end
    end
    Y=YBD.*YC;
    for i=1:N,
        YB(1,i)=trapz(measured_dBD,Y(:,i)');
    end
    Y=YAD.*YB;
    P_zeta_AB_BC_AC_ABCD=trapz(measured_dAD,Y);
end
```

0.1 Function Calculating $I_{\zeta_{XY}}$

In order to implement Equation (25), it is necessary to calculate the three dimensional indicator function $I_{\zeta_{XY}}$ which is set when there is a flipped realization with respect to line XY.

This function uses two other functions to calculate the corresponding λ_C and to create an indicator function δ_{ZD} which is set to one when $R_{D_1}^{AB} \subset H_{\overline{C}}$ and reset to zero when $R_{D_1}^{AB} \subset H_C$. These functions are explained in Subsections 0.2 and 0.3 respectively.

This function ignores the negligible flips (See Section 6 of Journal Paper) by not calculating λ_C for the cases of non-intersecting circles $\mathcal{C}(p(A), \overline{d}_{AD})$, $\mathcal{C}(p(B), \overline{d}_{BD})$ and setting them to be zero.

The matlab code of this function is as follows.

```
function I_zeta_XY=Calculate_I_zeta_XY(dXD,dYD,xy,R)
    %True distance XY between points X and Y
    XY=norm(xy(1,:)-xy(2,:));
    %Create a 3D dimensional condition matrix indicating
    %whether the circles centered at X and Y intersect.
    CirclesIntersect=(abs(dXD-XY)≤dYD).*(dYD≤min(dXD+XY,R));
    %Calculate Lambda_C
    Lamda_C=CirclesIntersect.*Calculate_lambda_C(dYD,dXD,xy);
    %create Indicator indicating whether
    %Z and D are on same side of XY or not
    sameSide=decide_ZandDonSameSideOfXY(xy);
    %make it a 3-dimensional matrix
    △_ZD=sameSide.*ones(N,N,N);
    I_zeta_XY=((△_ZD.*Lamda_C)≤dZD).*(dZD≤(min(△_ZD*R+Lamda_C,R)));
end
```

0.2 Function Calculating λ_C

From Proposition $A.1$ of the Journal paper, we know that

$$\lambda_C = \frac{\|p(C) - p_C^{AB}(D)\| + \|p(C) - p_{\overline{C}}^{AB}(D)\|}{2} \tag{0.1}$$

In order to calculate, $\|p(C) - p_C^{AB}(D)\|$ and $\|p(C) - p_{\overline{C}}^{AB}(D)\|$, lets extract $\triangle p(A)p(B)p(C)$, $\triangle p(A)p(B)p_C^{AB}(D)$ and $\triangle p(A)p(B)p_{\overline{C}}^{AB}(D)$ as shown in Figure 0.1. Since $p(C)$ is a ver-

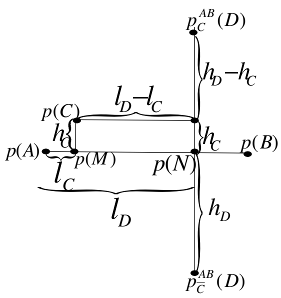

Fig. 0.1 Closer look at $\triangle p(A)p(B)p(C)$, $\triangle p(A)p(B)p_C^{AB}(D)$ and $\triangle p(A)p(B)p_{\overline{C}}^{AB}(D)$ from Figure $A.1$ of the Journal paper.

tex in both right angled triangles $p(A)p(M)p(C)$ and $p(B)p(M)p(C)$, using pythagoras theorem l_C can be calculated as

$$l_C = \frac{\|p(A) - p(C)\|^2 - \|p(B) - p(C)\|^2 + \|p(A) - p(B)\|^2}{2\|p(A) - p(B)\|} \tag{0.2}$$

Like wise, l_D can be calculated as

$$
\begin{aligned}
l_D &= \frac{\|p(A) - p_C^{AB}(D)\|^2 - \|p(B) - p_C^{AB}(D)\|^2 + \|p(A) - p(B)\|^2}{2\|p(A) - p(B)\|} \\
&= \frac{\bar{d}_{AD}^2 - \bar{d}_{BD}^2 + \|p(A) - p(B)\|^2}{2\|p(A) - p(B)\|}
\end{aligned} \tag{0.3}
$$

Since h_C is the height of the $\triangle p(A)p(B)p(C)$ and h_D is the height of the $\triangle p(A)p(B)p_C^{AB}(D)$ (equivalently can consider $\triangle p(A)p(B)p_{\bar{C}}^{AB}(D)$ as well), they can be written as

$$
h_C = \frac{2K_{\triangle p(A)p(B)p(C)}}{\|p(A) - p(B)\|} \tag{0.4}
$$

$$
h_D = \frac{2K_{\triangle p(A)p(B)p_C^{AB}(D)}}{\|p(A) - p(B)\|} \tag{0.5}
$$

where $K_{\triangle p(A)p(B)p(C)}$ and $K_{\triangle p(A)p(B)p_C^{AB}(D)}$ are the area of $\triangle p(A)p(B)p(C)$ and $\triangle p(A)p(B)p_C^{AB}(D)$ respectively and can be easily calculated using Heron's formula.

Thus $\|p(C) - p_C^{AB}(D)\|$ and $\|p(C) - p_{\bar{C}}^{AB}(D)\|$ can be calculated as

$$
\begin{aligned}
\|p(C) - p_C^{AB}(D)\| &= \sqrt{(l_D - l_C)^2 + (h_D - h_C)^2} \\
\|p(C) - p_{\bar{C}}^{AB}(D)\| &= \sqrt{(l_D - l_C)^2 + (h_D + h_C)^2}
\end{aligned}
$$

and substituted in (0.1) to calculate λ_C.

The matlab code of this function is as follows.

```matlab
function lambda_C=Calculate_lambda_C(dBD,dAD,xy)
    AB=norm(xy(1,:)-xy(2,:));
    BC=norm(xy(2,:)-xy(3,:));
    AC=norm(xy(1,:)-xy(3,:));

    %l_C=(AC^2-BC^2+AB^2)/(2*AB);
    %l_D=(dAD.^2-dBD.^2+AB^2)/(2*AB);
    l=((AC^2-BC^2)-(dAD.^2-dBD.^2))/(2*AB);

    sABC=(AB+BC+AC)/2;
    kABC=sqrt(sABC*(sABC-AB)*(sABC-BC)*(sABC-AC));

    sABD=(dAD+dBD+AB)/2;
    kABD=sqrt(sABD.*(sABD-AB).*(sABD-dBD).*(sABD-dAD));

    %h_C=2*kABC/AB;
    %h_D=2*kABD./AB;
    h_D1=2*(kABC-kABD)/AB;
    h_D2=2*(kABC+kABD)/AB;

    CD1=sqrt(l.^2+h_D1.^2);
    CD2=sqrt(l.^2+h_D2.^2);

    lambda_C=(x_CD1+x_CD2)/2;

    clear AB BC AC l sABC kABC sABD kABD h_D1 h_D2 CD1 CD2;
end
```

0.3 Function Deciding Whether Z and D Are On Same Side Of XY

The is a simple matlab code of this function using cross product and checking the signs to make a decision.

```
function sameSide=decide_ZandDonSameSideOfXY(xy)
    cZ = cross([xy(2,:)-xy(1,:) 0],[xy(3,:)-xy(1,:) 0]);
    cD = cross([xy(2,:)-xy(1,:) 0],[xy(4,:)-xy(1,:) 0]);
    if (sign(cZ(3))*sign(cD(3))==-1)
        sameSide=0;
    else %(sign(cZ(3))*sign(cD(3))==1)
        sameSide=1;
    end
end
```