HW#5 : Camera Calibration
ECE661

WONJO JUNG
Civil Engineering Dept.
Purdue University

Step I : Define the corners as the intersections of the fitted straight lines

To calibrate cameras, I used total 39 images which are on the homework web-site.
From the original image (figure 1.a), first detect edges using the Canny operator (figure 1.b). Then detect lines by the Hough transformation (figure 1.c). Since I get multiple responses for a single line, lines are refined (figure 1.d). Then vertical lines and horizontal lines are grouped and their intersections are acquired and labeled (figure 1.e).

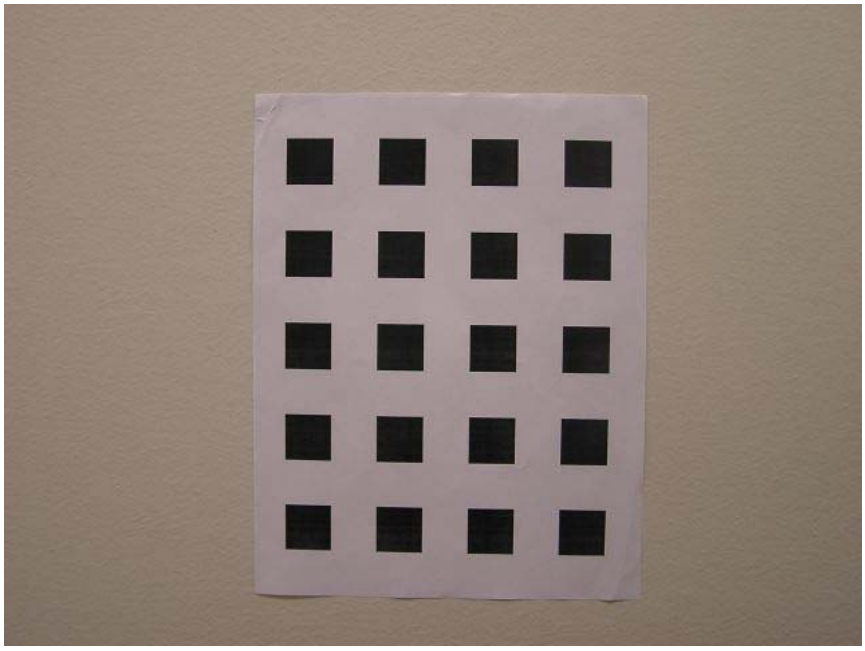


Figure 1.a. original image

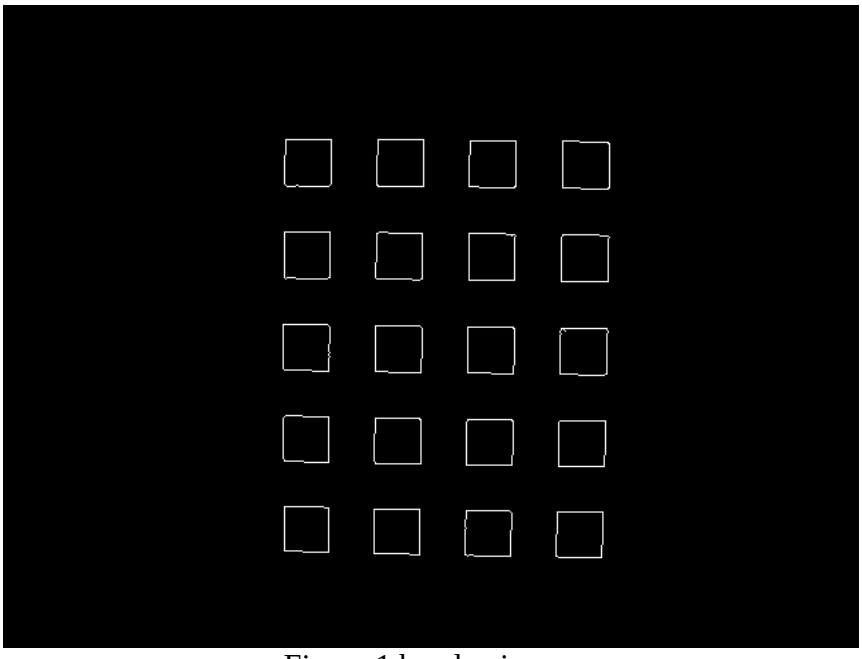


Figure 1.b. edge image

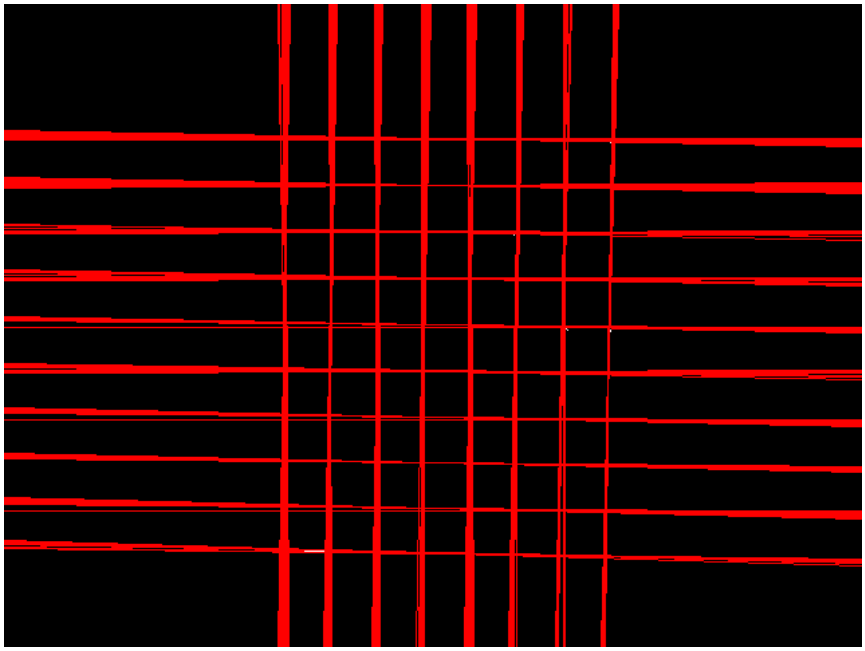


Figure 1.c. detected lines from edge image (initial responses)

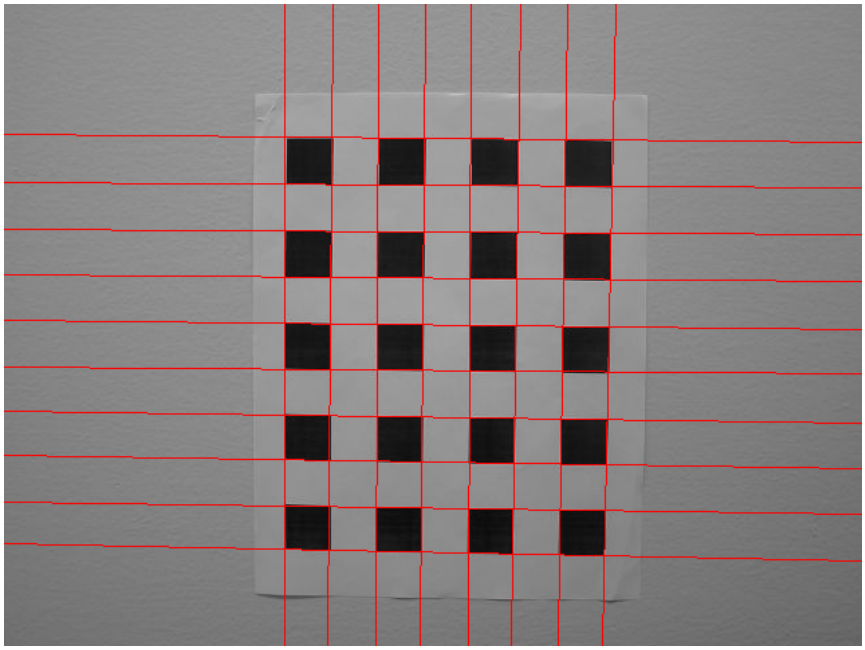


Figure 1.d. refined lines from the multiple responses for a single line

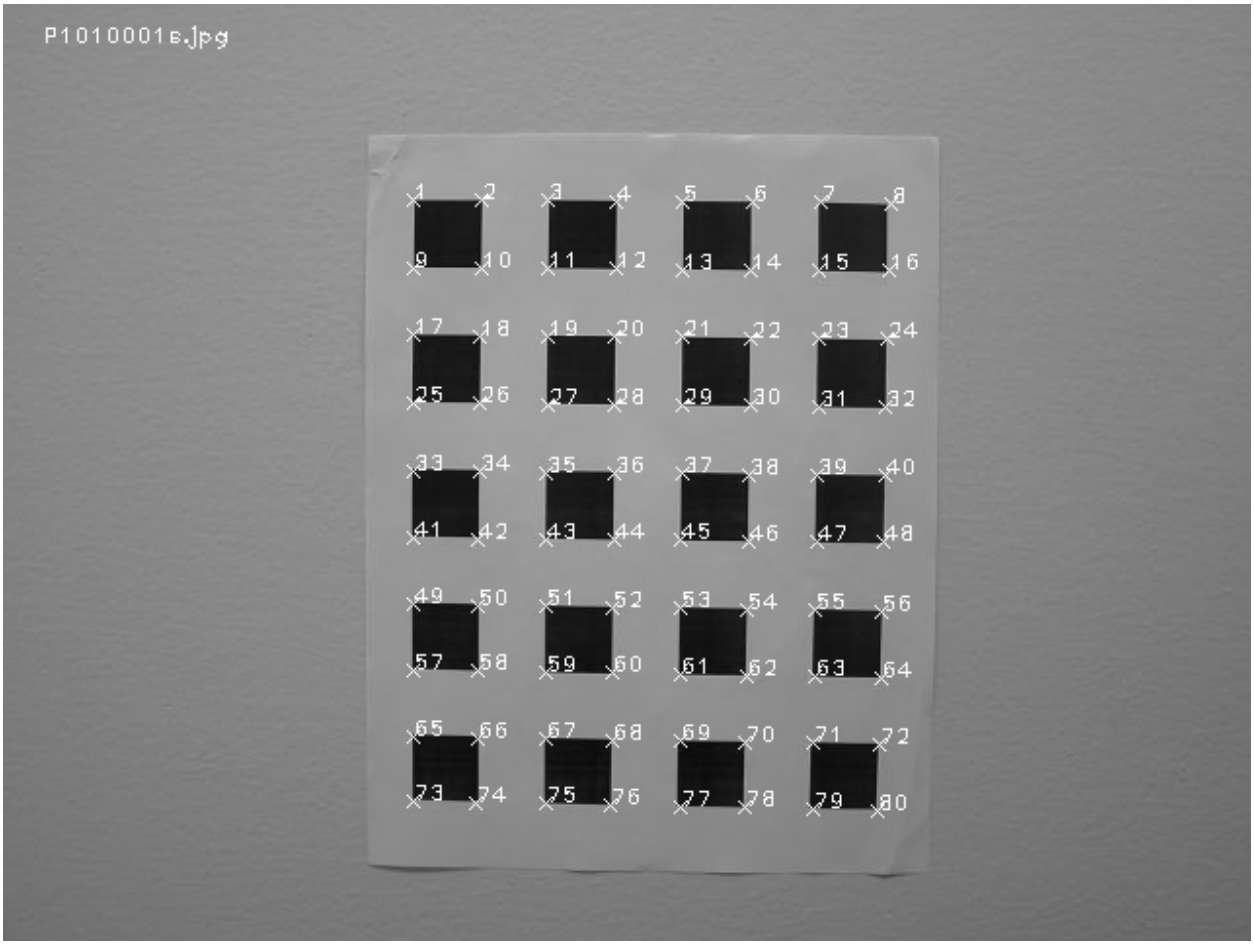


Figure 1.e. detected corners

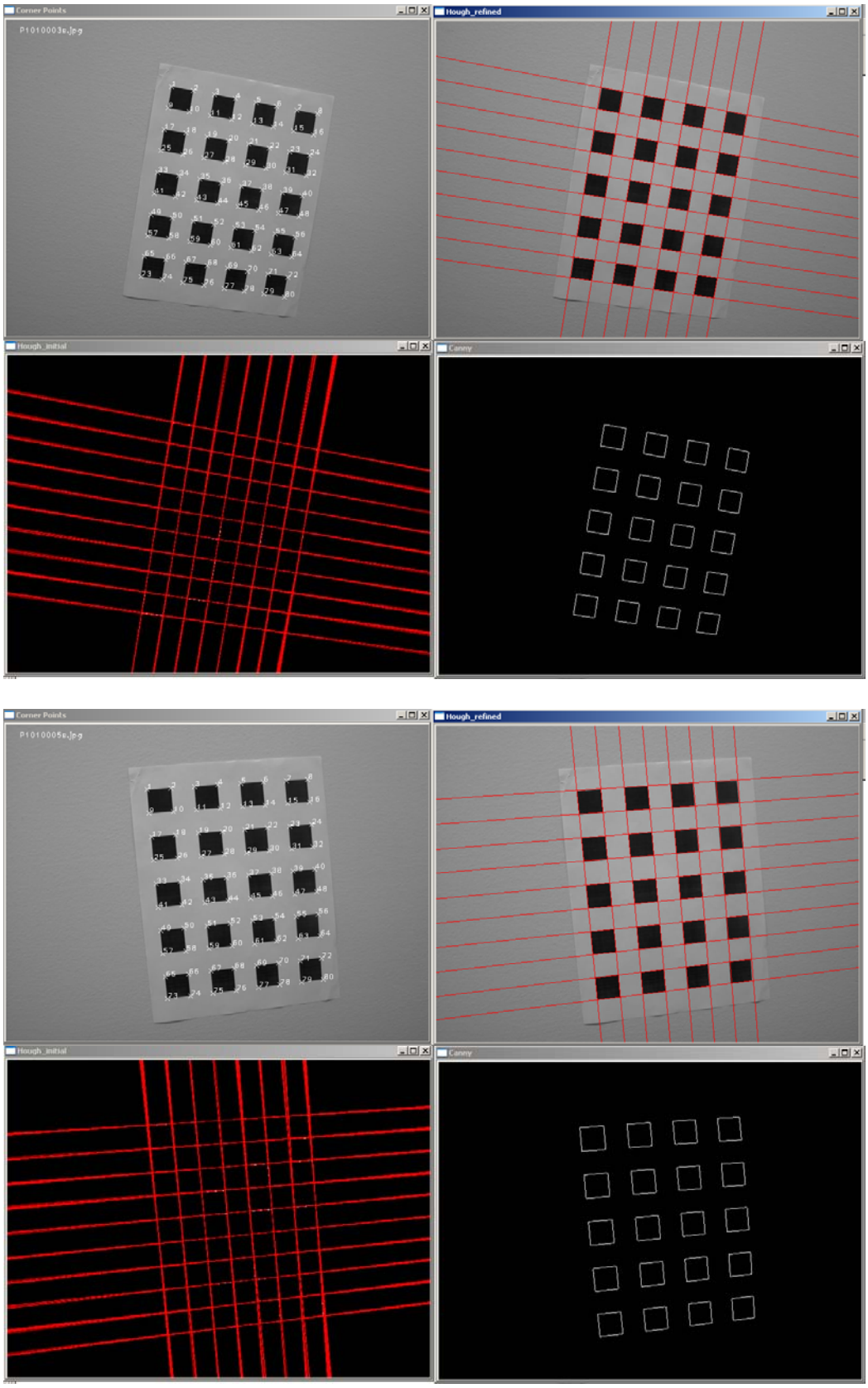
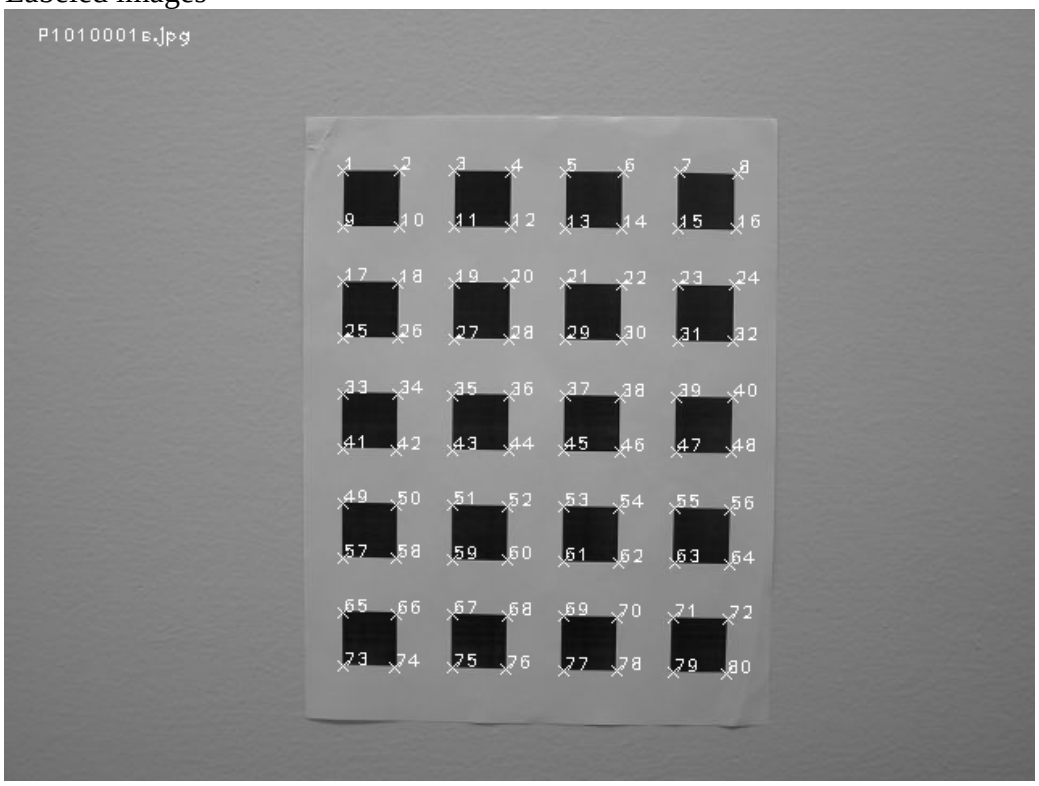
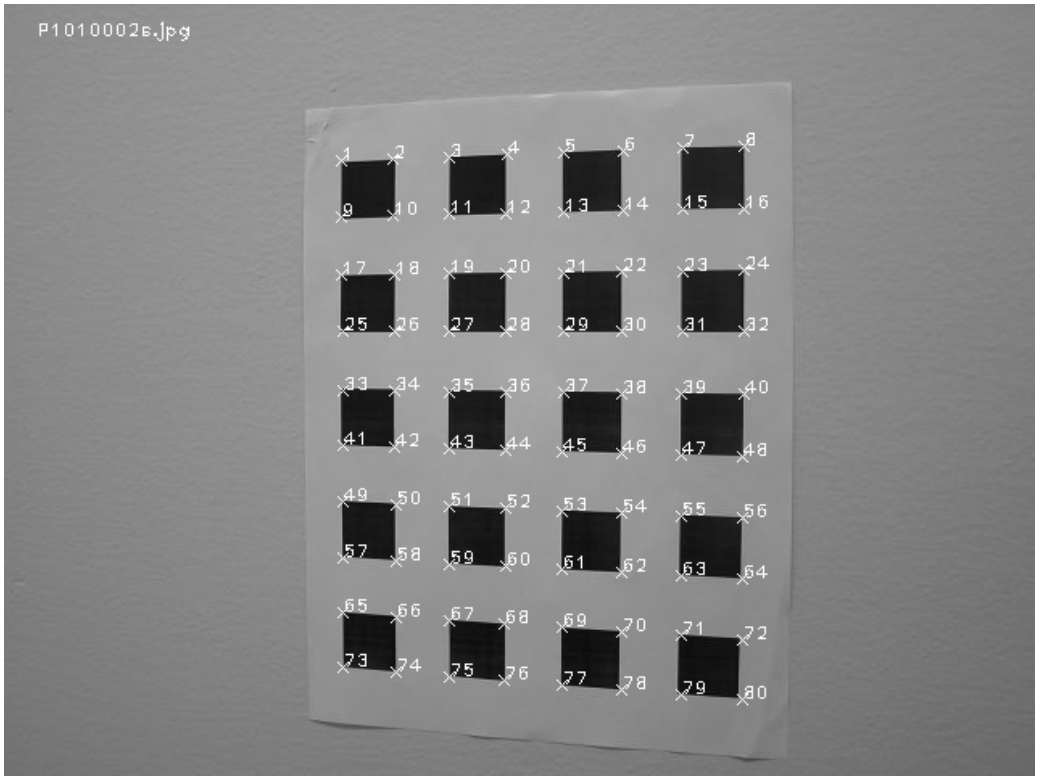
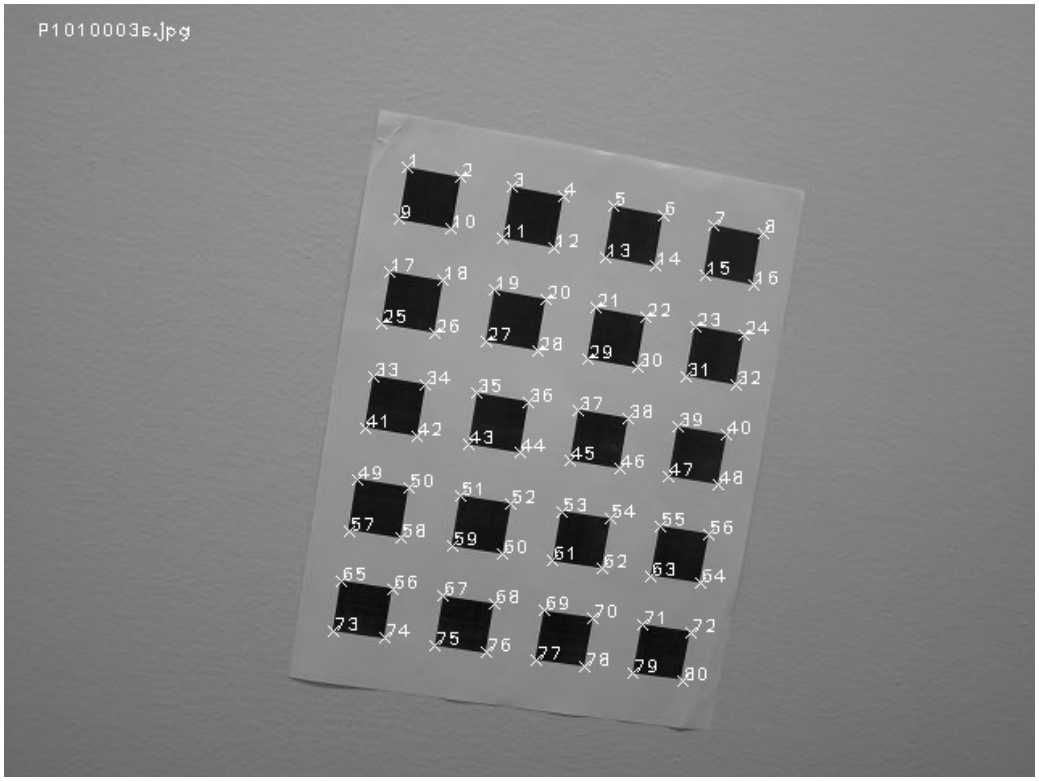


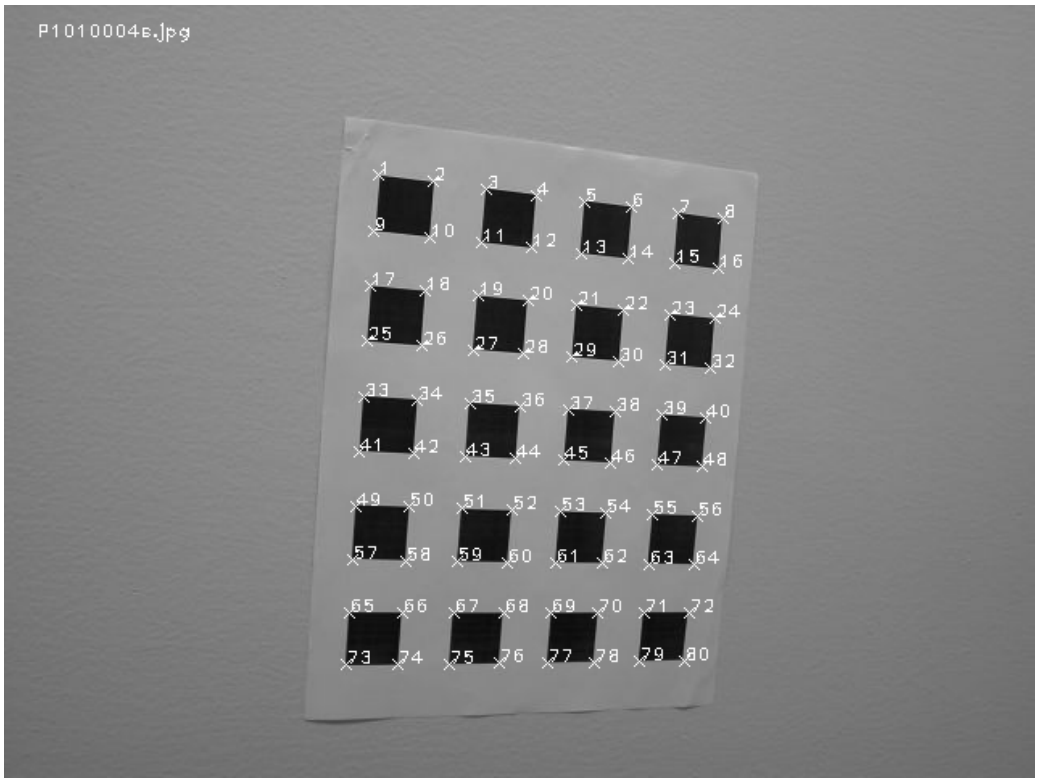
Figure 2. Examples of Corner Detection

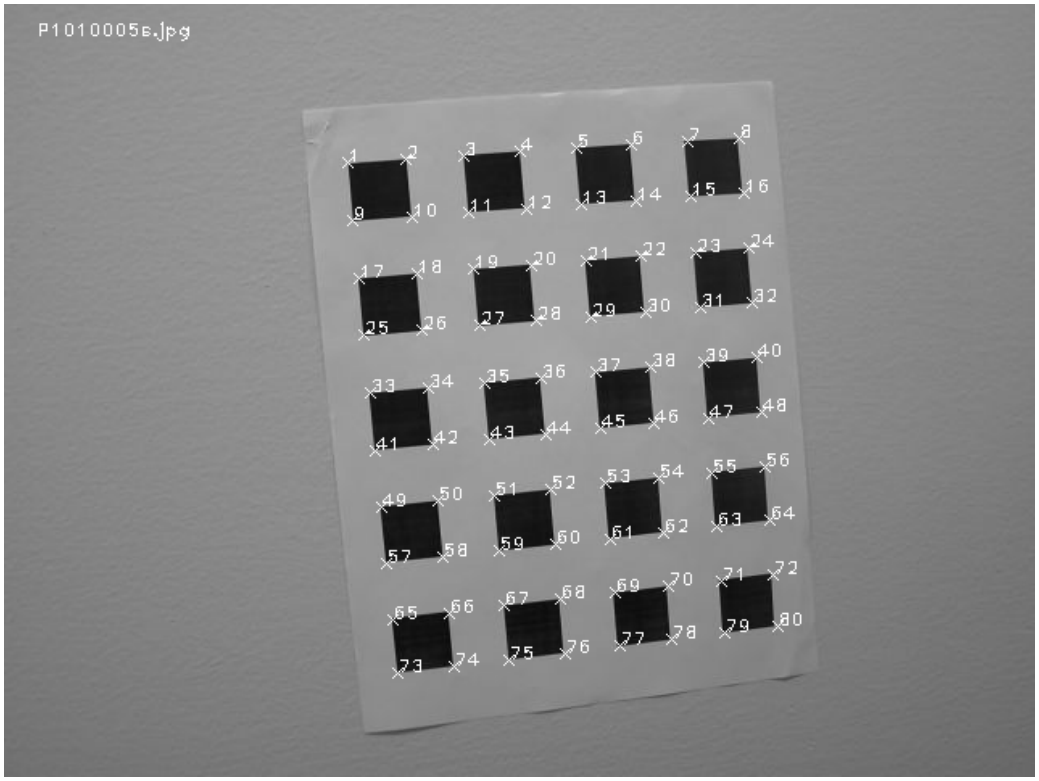
Labeled images

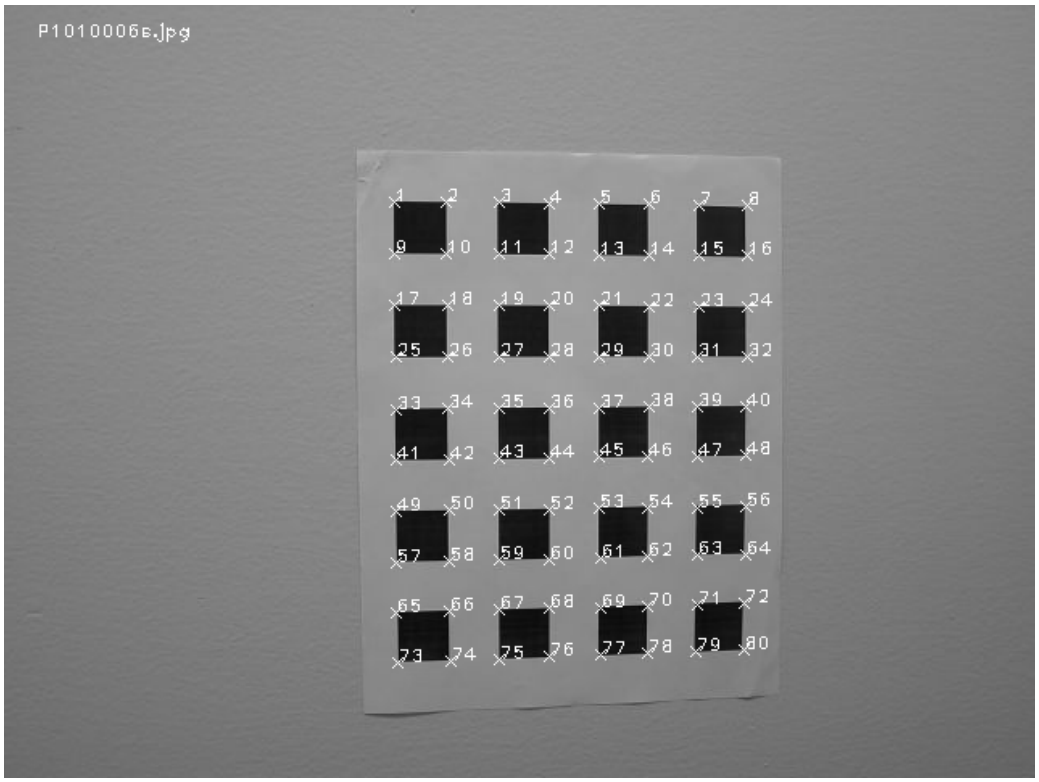





P1010003в.jpg


P1010004в.jpg





And so on (omitted)

Step II : Compute the intrinsic parameters and the extrinsic parameters using the algorithm described in Section 3.1 of Zhang's report.

1. estimate Homographies H for 39 images
2. from the estimated homographies, determine absolute conic ω
3. from the estimated absolute conic, determine intrinsic parameters and K
4. estimate extrinsic parameters
5. refine rotation matrix R so that $R^T R = I$

Step III : Refine all the camera parameters by applying the Levenberg-Marquardt method

The Jacobian matrix J is very large and the matrix K should be estimated using all images. To check the homography for each image, I applied the Levenberg-Marquardt method and checked errors. After removing 4 images having larg errors, then I processed following algorithm.

$$f = \min \sum_{i=1}^{35} \sum_{j=1}^{80} \left\| m_{ij} - \hat{m}(K, R_i, t_i, M_j) \right\|$$

1. estimate errors (geometric distances) using $P^{(k)}$ ($P^{(0)}$ is given by step II)
   $P = \left[\alpha, \beta, u_0, v_0, k, w_{x_i}, w_{y_i}, w_{z_i}, t_{X_i}, t_{Y_i}, t_{Z_i} \right]^T \quad (i = 1, 2, \cdots 35)$
   $errors(P, M_j) = d^T d,$ where $d =$ (detected corner pixel- estimate corner pixel using P)
   size of P : 5 (intrinsic parameters) + 6 (extrinsic parameters) $\times$ number of images
   size of d : 80(number of corners) $\times$ 2(x & y) $\times$ number of images
2. compute Jacobian J
3. compute Hessian H
4. damping $Hm = H + \lambda I$ so that the second derivative be non-negative
5. estimate $Ptemp = -Hm^{-1}(J^T d)$
6. estimate $dtemp$(detected corner pixel- estimate corner pixel using $Ptemp$)
7. if($errors(Ptemp, M_j) < errors(P, M_j)$
   $P^{(k+1)} = Ptemp$, $d = dtemp$, reduce damping factor λ ,
8. else amplify damping factor λ and go to the step 4.

Source Code

<main function>

```
void main()
{
	////////////////////////////////////
	// camera calibration
	////////////////////////////////////
	Calib_Corners("imagefilenames.txt","calib_report.txt",39,1);
	 cvWaitKey(0);
}
```

<Included functions>

```
// Homography
void Homography(double **datapoints, int N, CvMat* H);
void Homography(double **datapoints1, double **datapoint2, int N, CvMat* H);
void Homography(CvMat* datapoints1, CvMat* datapoints2, int N, CvMat* H);

// construct Normalization matrix for image coordinates
double NormalizationMatrixImg(int **Corners1,int N1,CvMat *MT);
double NormalizationMatrixImg(double **Corners1,int N1,CvMat *MT);
double NormalizationMatrixImg(CvMat *Corners1,int N1,CvMat *MT);

// draw detected lines(Hough) on the edge image (Canny)
void Edge_HoughCanny(IplImage *img, double **Corners, int viewEdgeFlag, int viewHoughFlag, int viewRefiedHoughFlag);

// calibration with only corner points
void Calib_Corners(char *imagefilenames, char* reportfilename,int Nfiles, int viewCornersFlag);

// set V matrix & b vector
void SetV(CvMat*H, int i, CvMat *V);

// transformation from Rotation matrix to Rodrigues representation
void R2Rodrigues(CvMat *R, CvMat *W);
void Rodrigues2R(CvMat *W, CvMat *R);

// REFINE camera parameters (Levenverg-Marquit Method)
void RefineCamera(CvMat *Rt, CvMat *K,CvMat *estK,CvMat *ObjectPoints, IplImage *img);

// Set Jacobian matrix
void setJall(CvMat *P,CvMat *J, int Nfiles, CvMat *ObjectPoints);
void setJ(CvMat *P,CvMat *J,CvMat *ObjectPoints);

// estimate geometric distances
double ErrorsGD(CvMat* P,CvMat* ObjectPoints,CvMat* ImgPoints,int N, CvMat* d,IplImage *img,int Viewflag);

// display
void CheckRtK(CvMat* Rt,CvMat* K,CvMat* ObjectPoints, IplImage *img);

// magnitude of residuals
double Residuals(CvMat* Rt,CvMat* K,CvMat* ObjectPoints,CvMat* ImgPoints,CvMat* d);
double Residuals(CvMat* Rt,CvMat* K,CvMat* ObjectPoints,CvMat* ImgPoints);
```

< function Calib_Corners >

Input : - imagefilenames : textfilename containing image file names
- reportfilename : report filename
- N : number of image files
- viewCornersFlag (if 1, show else do nothing)
- Allflag (if 1, apply LM to all images else apply LM to the each image)

Output : report in textfile format

```
void Calib_Corners(char *imagefilenames, char* reportfilename, int Nfiles, int viewCornersFlag, int Allflag){

	int i,j,k;
	double ***Corners;
	FILE *imgfn=fopen(imagefilenames,"rt");
	char tempimgfn[20];
	IplImage* img1,*temp,*temp2;
	Corners=new double **[Nfiles];
	for(i=0;i<Nfiles;i++) Corners[i]=new double *[80];
	for(i=0;i<Nfiles;i++)
	{
		for(j=0;j<100;j++) Corners[i][j]=new double [2];
	}
	// assign matrices
	CvMat* T  = cvCreateMat(3,3,CV_64FC1);			// Normalizing matrix for image coord.
	CvMat* Tp  = cvCreateMat(3,3,CV_64FC1);			// Normalizing matrix for object coord.
	CvMat* Tinv  = cvCreateMat(3,3,CV_64FC1);		// inverse of T
	CvMat* Tpinv  = cvCreateMat(3,3,CV_64FC1);		// inverse of Tp
	CvMat* ObjectPoints = cvCreateMat(3,80,CV_64FC1);// object coordinates
	CvMat* ImgPoints = cvCreateMat(3,80,CV_64FC1);		// image coordinates
	CvMat* nObjectPoints = cvCreateMat(3,80,CV_64FC1);	// normalized object coordinates
	CvMat* nImgPoints = cvCreateMat(3,80,CV_64FC1);		// normalized image coordinates
	CvMat* Hn  = cvCreateMat(3,3,CV_64FC1);			// estimated homography by normalized
coord.
	CvMat* Hntemp  = cvCreateMat(3,3,CV_64FC1);		// temporary matrix for calulation only
	CvMat *H[39];						// I don't
know how to dynamically assign CvMat
	CvMat* Hinv  = cvCreateMat(3,3,CV_64FC1);		// inverse of the estimated Homography
	CvMat* V  = cvCreateMat(2*Nfiles,6,CV_64FC1);		// Vb=0 for camera calibration
	CvMat* b  = cvCreateMat(6,1,CV_64FC1);			// elements of the absolute conic


	for(i=0;i<10;i++)
	{
		for(j=0;j<8;j++)
		{
			cvmSet(ObjectPoints,0,i*8+j,j*21.5/9);
			cvmSet(ObjectPoints,1,i*8+j,i*21.5/9);
			cvmSet(ObjectPoints,2,i*8+j,1);
		}
	}
	for(i=0;i<Nfiles;i++)
	{
		// Load images
		fscanf(imgfn,"%s",tempimgfn);
		img1 = cvLoadImage(tempimgfn,0);
		temp = cvLoadImage(tempimgfn,0);
		temp2 = cvLoadImage(tempimgfn,0);
		printf("Processing image %s. \n",tempimgfn);
		// Find & display lines and corner points
		Edge_HoughCanny(temp,Corners[i],1,1,1);		//ONOFF
		// display corners
		CvPoint Draw;
		char text[20];
		CvFont font;
		cvInitFont(&font, CV_FONT_HERSHEY_PLAIN, 0.8, 0.8, 0, 1);
```

```
		FILE *pointid;
		pointid=fopen("pointid.txt","rt");
		cvPutText(temp,tempimgfn,cvPoint(20,20),&font,CV_RGB(255,255,255));
		for(j=0;j<80;j++)
		{
			fscanf(pointid,"%s",text);
			Draw.x=(int)(Corners[i][j][0]+.5);
			Draw.y=(int)(Corners[i][j][1]+.5);
			draw_cross(temp,Draw,CV_RGB(255,255,255),3);
			cvPutText(temp,text,Draw,&font,CV_RGB(255,255,255));
		}
		if(viewCornersFlag==1)
		{
			cvNamedWindow( "Corner Points", 1 );
			cvShowImage( "Corner Points", temp );
			cvWaitKey(0);
		}
		// Estimate Homography
		// 1. Normalize
		for(j=0;j<80;j++)
		{
			cvmSet(ImgPoints,0,j,Corners[i][j][0]);
			cvmSet(ImgPoints,1,j,Corners[i][j][1]);
			cvmSet(ImgPoints,2,j,1);
		}
		NormalizationMatrixImg(ObjectPoints,80,T);
		cvMatMul(T,ObjectPoints,nObjectPoints);
		cvInvert(T,Tinv);
		NormalizationMatrixImg(ImgPoints,80,Tp);
		cvMatMul(Tp,ImgPoints,nImgPoints);
		cvInvert(Tp,Tpinv);
		// 2. estimate Hn
		Homography(nObjectPoints,nImgPoints,80,Hn);
		// 3. estimate H=inv(Tp)*Hn*T
		cvMatMul(Tpinv,Hn,Hntemp);
		H[i] = cvCreateMat(3,3,CV_64FC1);
		cvMatMul(Hntemp,T,H[i]);
		// Construct Vb=0
		SetV(H[i], i, V);
}
////////////////////////////////////////////
// estimate b (elements of the absolute conic)
CvMat* U  = cvCreateMat(Nfiles*2,Nfiles*2,CV_64FC1);
CvMat* D  = cvCreateMat(Nfiles*2,6,CV_64FC1);
CvMat* V2  = cvCreateMat(6,6,CV_64FC1);
CvMat* B = cvCreateMat(3,3,CV_64FC1);
cvSVD(V, D, U, V2, CV_SVD_V_T);
cvmSet(b,0,0,cvmGet(V2,5,0));
cvmSet(b,1,0,cvmGet(V2,5,1));
cvmSet(b,2,0,cvmGet(V2,5,2));
cvmSet(b,3,0,cvmGet(V2,5,3));
cvmSet(b,4,0,cvmGet(V2,5,4));
cvmSet(b,5,0,cvmGet(V2,5,5));
// setup B
cvmSet(B,0,0,cvmGet(b,0,0));
cvmSet(B,0,1,cvmGet(b,1,0));
cvmSet(B,1,0,cvmGet(b,1,0));
cvmSet(B,1,1,cvmGet(b,2,0));
cvmSet(B,0,2,cvmGet(b,3,0));
cvmSet(B,2,0,cvmGet(b,3,0));
cvmSet(B,1,2,cvmGet(b,4,0));
cvmSet(B,2,1,cvmGet(b,4,0));
cvmSet(B,2,2,cvmGet(b,5,0));


////////////////////////////////////////////
// estimate intrinsic parameters
CvMat* K = cvCreateMat(3,3,CV_64FC1);
CvMat* Kinv = cvCreateMat(3,3,CV_64FC1);
```

```
	double vo     = (cvmGet(B,0,1)*cvmGet(B,0,2)-cvmGet(B,0,0)*cvmGet(B,1,2))/(cvmGet(B,0,0)*cvmGet(B,1,1)-
cvmGet(B,0,1)*cvmGet(B,0,1));
	double lamda  =  cvmGet(B,2,2)-(cvmGet(B,0,2)*cvmGet(B,0,2)+vo*(cvmGet(B,0,1)*cvmGet(B,0,2)-
cvmGet(B,0,0)*cvmGet(B,1,2)))/cvmGet(B,0,0);
	double alpha  =  sqrt(lamda/cvmGet(B,0,0));
	double beta   =  sqrt(lamda*cvmGet(B,0,0)/(cvmGet(B,0,0)*cvmGet(B,1,1)-cvmGet(B,0,1)*cvmGet(B,0,1)));
	double gamma  =  -cvmGet(B,0,1)*alpha*alpha*beta/lamda;
	double uo     =  gamma*vo/beta-cvmGet(B,0,2)*alpha*alpha/lamda;
	cvmSet(K,0,0,alpha);
	cvmSet(K,0,1,gamma);
	cvmSet(K,0,2,uo);
	cvmSet(K,1,0,0);
	cvmSet(K,1,1,beta);
	cvmSet(K,1,2,vo);
	cvmSet(K,2,0,0);
	cvmSet(K,2,1,0);
	cvmSet(K,2,2,1);
	cvInvert(K,Kinv);

	////////////////////////////////////////////
	// estimate extrinsic parameters
	CvMat* h1     = cvCreateMat(3,1,CV_64FC1);
	CvMat* h2     = cvCreateMat(3,1,CV_64FC1);
	CvMat* h3     = cvCreateMat(3,1,CV_64FC1);
	CvMat* r1     = cvCreateMat(3,1,CV_64FC1);
	CvMat* r2     = cvCreateMat(3,1,CV_64FC1);
	CvMat* r3     = cvCreateMat(3,1,CV_64FC1);
	CvMat* t      = cvCreateMat(3,1,CV_64FC1);
	CvMat* Q      = cvCreateMat(3,3,CV_64FC1);
	CvMat* R      = cvCreateMat(3,3,CV_64FC1);
	CvMat* UU     = cvCreateMat(3,3,CV_64FC1);
	CvMat* DD     = cvCreateMat(3,3,CV_64FC1);
	CvMat* VV     = cvCreateMat(3,3,CV_64FC1);
	CvMat* Vt     = cvCreateMat(3,3,CV_64FC1);
	CvMat* Rt[39];
	double lamda1;
	imgfn=fopen(imagefilenames,"rt");
	for(i=0;i<Nfiles;i++)
	{
		fscanf(imgfn,"%s",tempimgfn);
		temp = cvLoadImage(tempimgfn,0);
		Rt[i] = cvCreateMat(3,4,CV_64FC1);
		// setup column vector h1,h2,h3 from H[i]
		cvmSet(h1,0,0,cvmGet(H[i],0,0));
		cvmSet(h1,1,0,cvmGet(H[i],1,0));
		cvmSet(h1,2,0,cvmGet(H[i],2,0));
		cvmSet(h2,0,0,cvmGet(H[i],0,1));
		cvmSet(h2,1,0,cvmGet(H[i],1,1));
		cvmSet(h2,2,0,cvmGet(H[i],2,1));
		cvmSet(h3,0,0,cvmGet(H[i],0,2));
		cvmSet(h3,1,0,cvmGet(H[i],1,2));
		cvmSet(h3,2,0,cvmGet(H[i],2,2));
		// estimate the rotation matrix R and the translation vector t
		cvMatMul(Kinv,h1,r1);
		cvMatMul(Kinv,h2,r2);
		cvMatMul(Kinv,h3,t);
		lamda1=1/sqrt(pow(cvmGet(r1,0,0),2)+pow(cvmGet(r1,1,0),2)+pow(cvmGet(r1,2,0),2));
		cvmSet(r1,0,0,cvmGet(r1,0,0)*lamda1);
		cvmSet(r1,1,0,cvmGet(r1,1,0)*lamda1);
		cvmSet(r1,2,0,cvmGet(r1,2,0)*lamda1);
		cvmSet(r2,0,0,cvmGet(r2,0,0)*lamda1);
		cvmSet(r2,1,0,cvmGet(r2,1,0)*lamda1);
		cvmSet(r2,2,0,cvmGet(r2,2,0)*lamda1);
		cvmSet(t,0,0,cvmGet(t,0,0)*lamda1);
		cvmSet(t,1,0,cvmGet(t,1,0)*lamda1);
		cvmSet(t,2,0,cvmGet(t,2,0)*lamda1);
		cvCrossProduct(r1,r2,r3);
		cvmSet(Q,0,0,cvmGet(r1,0,0));
		cvmSet(Q,1,0,cvmGet(r1,1,0));
		cvmSet(Q,2,0,cvmGet(r1,2,0));
```

```
                cvmSet(Q,0,1,cvmGet(r2,0,0));
                cvmSet(Q,1,1,cvmGet(r2,1,0));
                cvmSet(Q,2,1,cvmGet(r2,2,0));
                cvmSet(Q,0,2,cvmGet(r3,0,0));
                cvmSet(Q,1,2,cvmGet(r3,1,0));
                cvmSet(Q,2,2,cvmGet(r3,2,0));
                // Refine Q become orthogonal matrix R
                cvSVD(Q, DD, UU, VV, CV_SVD_V_T);
                cvMatMul(UU,VV,R);
                // Rt matrix
                cvmSet(Rt[i],0,0,cvmGet(R,0,0));
                cvmSet(Rt[i],1,0,cvmGet(R,1,0));
                cvmSet(Rt[i],2,0,cvmGet(R,2,0));
                cvmSet(Rt[i],0,1,cvmGet(R,0,1));
                cvmSet(Rt[i],1,1,cvmGet(R,1,1));
                cvmSet(Rt[i],2,1,cvmGet(R,2,1));
                cvmSet(Rt[i],0,2,cvmGet(R,0,2));
                cvmSet(Rt[i],1,2,cvmGet(R,1,2));
                cvmSet(Rt[i],2,2,cvmGet(R,2,2));
                cvmSet(Rt[i],0,3,cvmGet(t,0,0));
                cvmSet(Rt[i],1,3,cvmGet(t,1,0));
                cvmSet(Rt[i],2,3,cvmGet(t,2,0));
        }

        ////////////////////////////////////////////
        // report
        ////////////////////////////////////////////
        // 1. General
        FILE *report=fopen(reportfilename,"wt");
        fprintf(report,"================== Camera Calibration Report ====================\n");
        fprintf(report,"Number of images : %d\n",Nfiles);
        fprintf(report,"Image size : 640 X 480 pixel\n");
        fprintf(report,"Line detection method : the Hough transformation\n");
        fprintf(report,"Point detection method : intersecting lines acquired by the Hough transformation\n\n\n");
        // 2. Absolute Conic & intrinsic parameters
        fprintf(report,"================== Estimated absolute Conic ====================\n");
        for(i=0;i<3;i++)
        {
                fprintf(report,"| %16.8f  %16.8f  %16.8f |\n",cvmGet(B,i,0),cvmGet(B,i,1),cvmGet(B,i,2));
        }
        fprintf(report,"\n================== Estimated Instrinsic Parameters ====================\n");
        fprintf(report,"vo=%f\n",vo);
        fprintf(report,"lamda=%f\n",lamda);
        fprintf(report,"alpha=%f\n",alpha);
        fprintf(report,"beta=%f\n",beta);
        fprintf(report,"gamma=%f\n",gamma);
        fprintf(report,"uo=%f\n",uo);
        fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f  |\n",cvmGet(K,0,0),cvmGet(K,0,1),cvmGet(K,0,2));
        fprintf(report,"|     K    | = |  %16.8f  %16.8f  %16.8f  |\n",cvmGet(K,1,0),cvmGet(K,1,1),cvmGet(K,1,2));
        fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f  |\n",cvmGet(K,2,0),cvmGet(K,2,1),cvmGet(K,2,2));
        fprintf(report,"\n");

        // 3. Extrinsic parameters
        fprintf(report,"\n================== Estimated Extrinsic Parameters ====================\n");
        imgfn=fopen(imagefilenames,"rt");
        double dtd[39],dtdrefined[39];
        for(i=0;i<Nfiles;i++)
        {
                // estimate residuals

                fscanf(imgfn,"%s",tempimgfn);
                fprintf(report,"%s\n",tempimgfn);
                fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f  %16.8f
|\n",cvmGet(Rt[i],0,0),cvmGet(Rt[i],0,1),cvmGet(Rt[i],0,2),cvmGet(Rt[i],0,3));
                fprintf(report,"|   R   | t | = |  %16.8f  %16.8f  %16.8f  %16.8f
|\n",cvmGet(Rt[i],1,0),cvmGet(Rt[i],1,1),cvmGet(Rt[i],1,2),cvmGet(Rt[i],1,3));
                fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f  %16.8f
|\n",cvmGet(Rt[i],2,0),cvmGet(Rt[i],2,1),cvmGet(Rt[i],2,2),cvmGet(Rt[i],2,3));
                dtd[i]=Residuals(Rt[i],K,ObjectPoints,ImgPoints);
                fprintf(report,"errors(dtd) = %f pixels\n",dtd[i]);
```

```
		fprintf(report,"\n");
	}

	if (Allflag==0)  // perform LM algoritm for each image seperately
	{
		//////////////////////////////////////////
		// Refine Parameters (for each camera)
		CvMat* estK = cvCreateMat(3,3,CV_64FC1);
		imgfn=fopen(imagefilenames,"rt");
		for(i=0;i<Nfiles;i++)
		{
			fscanf(imgfn,"%s",tempimgfn);
			for(j=0;j<80;j++)
			{
				cvmSet(ImgPoints,0,j,Corners[i][j][0]);
				cvmSet(ImgPoints,1,j,Corners[i][j][1]);
				cvmSet(ImgPoints,2,j,1);
			}
			RefineCamera(Rt[i],K,estK,ObjectPoints,ImgPoints,temp2);
			// display Homography
			// CheckRtK(Rt[i],K,ObjectPoints,temp);
			fprintf(report,"\nRefined Instrinsic Parameters (%s) \n",tempimgfn);
			fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f
|\n",cvmGet(estK,0,0),cvmGet(estK,0,1),cvmGet(estK,0,2));
			fprintf(report,"|     K    | = |  %16.8f  %16.8f  %16.8f
|\n",cvmGet(estK,1,0),cvmGet(estK,1,1),cvmGet(estK,1,2));
			fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f
|\n",cvmGet(estK,2,0),cvmGet(estK,2,1),cvmGet(estK,2,2));
			// 3. Extrinsic parameters
			fprintf(report,"\nRefined Extrinsic Parameters (%s) \n",tempimgfn);
			fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f  %16.8f
|\n",cvmGet(Rt[i],0,0),cvmGet(Rt[i],0,1),cvmGet(Rt[i],0,2),cvmGet(Rt[i],0,3));
			fprintf(report,"|   R   | t | = |  %16.8f  %16.8f  %16.8f  %16.8f
|\n",cvmGet(Rt[i],1,0),cvmGet(Rt[i],1,1),cvmGet(Rt[i],1,2),cvmGet(Rt[i],1,3));
			fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f  %16.8f
|\n",cvmGet(Rt[i],2,0),cvmGet(Rt[i],2,1),cvmGet(Rt[i],2,2),cvmGet(Rt[i],2,3));
			dtdrefined[i]=Residuals(Rt[i],estK,ObjectPoints,ImgPoints);
			fprintf(report,"errors(dtd) = %f pixels\n",dtdrefined[i]);
			fprintf(report,"\n");
		}
		double improvementNumer=0;
		double improvementDenom=0;
		for(i=0;i<Nfiles;i++)
		{
			improvementNumer=improvementNumer+dtdrefined[i];
			improvementDenom=improvementDenom+dtd[i];
		}
		fprintf(report,"Improvement = ( sum of initial errors ) / (sum of refined errors)
= %f.\n",improvementNumer/improvementDenom);
	}
	else if (Allflag==1) // LM algorithm is applied to entire image set
	{
		//////////////////////////////////////////
		// Refine Parameters (using entire images)
		CvMat* J       = cvCreateMat(160*Nfiles,5+6*Nfiles,CV_64FC1); // Jacobian Matrix
		CvMat* JT      = cvCreateMat(5+6*Nfiles,160*Nfiles,CV_64FC1);          // Transposed Jacobian Matrix
		CvMat* JTd     = cvCreateMat(5+6*Nfiles,1,CV_64FC1);
		CvMat* Hessian = cvCreateMat(5+6*Nfiles,5+6*Nfiles,CV_64FC1);          // Hessian Matrix
		CvMat* Hessianinv = cvCreateMat(5+6*Nfiles,5+6*Nfiles,CV_64FC1);
		CvMat* I       = cvCreateMat(5+6*Nfiles,5+6*Nfiles,CV_64FC1); // Identity Matrix
		CvMat* Hessian_lm    = cvCreateMat(5+6*Nfiles,5+6*Nfiles,CV_64FC1);    // H_lm Matrix
		CvMat* Hessian_lminv = cvCreateMat(5+6*Nfiles,5+6*Nfiles,CV_64FC1);  // H_lm Matrix
		CvMat* P       = cvCreateMat(5+6*Nfiles,1,CV_64FC1);          // Parameters
		CvMat* Ptemp   = cvCreateMat(5+6*Nfiles,1,CV_64FC1);          // temporary P
		CvMat* dP      = cvCreateMat(5+6*Nfiles,1,CV_64FC1);          // delta P
		CvMat* Ppartial= cvCreateMat(5+6*Nfiles,1,CV_64FC1);          // Partial P
		CvMat* Rtemp   = cvCreateMat(3,3,CV_64FC1);
		CvMat* W       = cvCreateMat(3,1,CV_64FC1);          // Rodrigues representation of R
		CvMat* Wtemp   = cvCreateMat(3,1,CV_64FC1);          // Rodrigues representation of R
		CvMat* d       = cvCreateMat(160*Nfiles,1,CV_64FC1);
```

```
CvMat* dtemp   = cvCreateMat(160*Nfiles,1,CV_64FC1);
CvMat* dpartial= cvCreateMat(160,1,CV_64FC1);
CvMat* Homography    = cvCreateMat(3,3,CV_64FC1);
CvMat* HomographyTemp    = cvCreateMat(3,3,CV_64FC1);
double threshold=1e-6;
double error,error0,error2,delta=100;
int Iteration=0;
double damping=0.01;                        // create initial damping factor
// setup identity matrix
for(i=0;i<5+6*Nfiles;i++)
{
        for(j=0;j<5+6*Nfiles;j++)
        {
                if(i==j) cvmSet(I,i,j,1);
                else cvmSet(I,i,j,0);
        }
}
// setup J
// Assign Initial Values
cvmSet(P,0,0,cvmGet(K,0,0));    //au (alpha)
cvmSet(P,1,0,cvmGet(K,1,1));    //av (beta)
cvmSet(P,2,0,cvmGet(K,0,2));    //u0
cvmSet(P,3,0,cvmGet(K,1,2));    //v0
cvmSet(P,4,0,cvmGet(K,0,1));    //sk (lamda)
for(i=0;i<Nfiles;i++)
{
        // Convert R to W
        cvmSet(R,0,0,cvmGet(Rt[i],0,0));
        cvmSet(R,1,0,cvmGet(Rt[i],1,0));
        cvmSet(R,2,0,cvmGet(Rt[i],2,0));
        cvmSet(R,0,1,cvmGet(Rt[i],0,1));
        cvmSet(R,1,1,cvmGet(Rt[i],1,1));
        cvmSet(R,2,1,cvmGet(Rt[i],2,1));
        cvmSet(R,0,2,cvmGet(Rt[i],0,2));
        cvmSet(R,1,2,cvmGet(Rt[i],1,2));
        cvmSet(R,2,2,cvmGet(Rt[i],2,2));
        cvmSet(t,0,0,cvmGet(Rt[i],0,3));
        cvmSet(t,1,0,cvmGet(Rt[i],1,3));
        cvmSet(t,2,0,cvmGet(Rt[i],2,3));
        R2Rodrigues(R, W);
        cvmSet(P,5+6*i ,0,cvmGet(W,0,0));       //wx
        cvmSet(P,6+6*i ,0,cvmGet(W,1,0));       //wy
        cvmSet(P,7+6*i ,0,cvmGet(W,2,0));       //wz
        cvmSet(P,8+6*i ,0,cvmGet(t,0,0));       //tx
        cvmSet(P,9+6*i ,0,cvmGet(t,1,0));       //ty
        cvmSet(P,10+6*i,0,cvmGet(t,2,0));       //tz
}
// Initial geometric distance
error0=0;
for(i=0;i<Nfiles;i++)
{
        for(j=0;j<80;j++)
        {
                cvmSet(ImgPoints,0,j,Corners[i][j][0]);
                cvmSet(ImgPoints,1,j,Corners[i][j][1]);
                cvmSet(ImgPoints,2,j,1);
        }
        cvmSet(Ppartial,0,0,cvmGet(P,0,0));
        cvmSet(Ppartial,1,0,cvmGet(P,1,0));
        cvmSet(Ppartial,2,0,cvmGet(P,2,0));
        cvmSet(Ppartial,3,0,cvmGet(P,3,0));
        cvmSet(Ppartial,4,0,cvmGet(P,4,0));
        cvmSet(Ppartial,5,0,cvmGet(P,5+i*6,0));
        cvmSet(Ppartial,6,0,cvmGet(P,6+i*6,0));
        cvmSet(Ppartial,7,0,cvmGet(P,7+i*6,0));
        cvmSet(Ppartial,8,0,cvmGet(P,8+i*6,0));
        cvmSet(Ppartial,9,0,cvmGet(P,9+i*6,0));
        cvmSet(Ppartial,10,0,cvmGet(P,10+i*6,0));
        error0=error0+ErrorsGD(Ppartial,ObjectPoints,ImgPoints,80,dpartial);
        for(j=0;j<80;j++)
```

```
		{
			cvmSet(d,160*i+j*2   , 0 , cvmGet(dpartial,j*2   ,0));
			cvmSet(d,160*i+j*2+1 , 0 , cvmGet(dpartial,j*2+1 ,0));
		}
}
printf("Initial error=%f\n",error0);
// Iterate using LM method
int updateJ=1;
error=error0;
for(i=0;i<200;i++)
{
		if(updateJ==1)
		{
			// determine Jacobian matrix J
			setJall(P,J,Nfiles,ObjectPoints);
			// Hessian Matrix
			cvTranspose(J,JT);
			cvMatMul(JT,J,Hessian);
		}
		// apply damping factor
		cvScaleAdd(I,cvScalar(damping),Hessian,Hessian_lm);
		cvInvert(Hessian_lm,Hessian_lminv);
		// temporary update P Ptemp=P-inv(H_lm)*(J'*d)
		cvMatMul(JT,d,JTd);
		cvMatMul(Hessian_lminv,JTd,dP);
		cvScaleAdd(dP,cvScalar(-1),P,Ptemp);
		delta=0;
		for(j=0;j<5+6*Nfiles;j++) delta=delta+pow(cvmGet(dP,j,0),2);
		delta=sqrt(delta)/(5+6*Nfiles);
		// check errors by Ptemp
		error2=0;
		for(j=0;j<Nfiles;j++)
		{
			for(k=0;k<80;k++)
			{
				cvmSet(ImgPoints,0,k,Corners[j][k][0]);
				cvmSet(ImgPoints,1,k,Corners[j][k][1]);
				cvmSet(ImgPoints,2,k,1);
			}
			cvmSet(Ppartial,0,0,cvmGet(Ptemp,0,0));
			cvmSet(Ppartial,1,0,cvmGet(Ptemp,1,0));
			cvmSet(Ppartial,2,0,cvmGet(Ptemp,2,0));
			cvmSet(Ppartial,3,0,cvmGet(Ptemp,3,0));
			cvmSet(Ppartial,4,0,cvmGet(Ptemp,4,0));
			cvmSet(Ppartial,5,0,cvmGet(Ptemp,5+j*6,0));
			cvmSet(Ppartial,6,0,cvmGet(Ptemp,6+j*6,0));
			cvmSet(Ppartial,7,0,cvmGet(Ptemp,7+j*6,0));
			cvmSet(Ppartial,8,0,cvmGet(Ptemp,8+j*6,0));
			cvmSet(Ppartial,9,0,cvmGet(Ptemp,9+j*6,0));
			cvmSet(Ppartial,10,0,cvmGet(Ptemp,10+j*6,0));
			error2=error2+ErrorsGD(Ppartial,ObjectPoints,ImgPoints,80,dpartial);
			for(k=0;k<80;k++)
			{
				cvmSet(dtemp,160*j+k*2   , 0 , cvmGet(dpartial,k*2   ,0));
				cvmSet(dtemp,160*j+k*2+1 , 0 , cvmGet(dpartial,k*2+1 ,0));
			}
		}
		if(error2<error) //decrease damping factor and update
		{
			damping=damping/10;
			for(j=0;j<5+6*Nfiles;j++) cvmSet(P,j,0,cvmGet(Ptemp,j,0));
			for(j=0;j<160*Nfiles;j++) cvmSet(d,j,0,cvmGet(dtemp,j,0));
			error=error2;
			updateJ=1;
			printf("iteration %d ] error=%f\n",i+1,error);
			printf("              delta=%f\n",delta);
		}
		else  // increase damping factor and try again
		{
			updateJ=0;
```

```
                                        i=i-1;
                                        damping=damping*10;
                                }
                                if(delta<threshold) break;
                        }
                        printf("Number of iteration = %d\n",i);
                        printf("delta = %f\n",delta);
                        printf("error=%f\n",error);
                        printf("damping=%f\n",damping);
                        // Finally estimated K
                        cvmSet(K,0,0,cvmGet(P,0,0));
                        cvmSet(K,0,1,cvmGet(P,4,0));
                        cvmSet(K,1,1,cvmGet(P,1,0));
                        cvmSet(K,0,2,cvmGet(P,2,0));
                        cvmSet(K,1,2,cvmGet(P,3,0));
                        // Update Rt
                        for(i=0;i<Nfiles;i++)
                        {
                                cvmSet(W,0,0,cvmGet(P,5+6*i,0));
                                cvmSet(W,1,0,cvmGet(P,6+6*i,0));
                                cvmSet(W,2,0,cvmGet(P,7+6*i,0));
                                Rodrigues2R(W,R);
                                cvmSet(Rt[i],0,0,cvmGet(R,0,0));
                                cvmSet(Rt[i],1,0,cvmGet(R,1,0));
                                cvmSet(Rt[i],2,0,cvmGet(R,2,0));
                                cvmSet(Rt[i],0,1,cvmGet(R,0,1));
                                cvmSet(Rt[i],1,1,cvmGet(R,1,1));
                                cvmSet(Rt[i],2,1,cvmGet(R,2,1));
                                cvmSet(Rt[i],0,2,cvmGet(R,0,2));
                                cvmSet(Rt[i],1,2,cvmGet(R,1,2));
                                cvmSet(Rt[i],2,2,cvmGet(R,2,2));
                                cvmSet(Rt[i],0,3,cvmGet(P,8+6*i,0));
                                cvmSet(Rt[i],1,3,cvmGet(P,9+6*i,0));
                                cvmSet(Rt[i],2,3,cvmGet(P,10+6*i,0));
                        }
                        fprintf(report,"\nRefined Instrinsic Parameters\n",tempimgfn);
                        fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f  |\n",cvmGet(K,0,0),cvmGet(K,0,1),cvmGet(K,0,2));
                        fprintf(report,"|     K    | = |  %16.8f  %16.8f  %16.8f  |\n",cvmGet(K,1,0),cvmGet(K,1,1),cvmGet(K,1,2));
                        fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f  |\n",cvmGet(K,2,0),cvmGet(K,2,1),cvmGet(K,2,2));
                        imgfn=fopen(imagefilenames,"rt");
                        for(i=0;i<Nfiles;i++)
                        {
                                fscanf(imgfn,"%s",tempimgfn);
                                // 3. Extrinsic parameters
                                fprintf(report,"\nRefined Extrinsic Parameters (%s) \n",tempimgfn);
                                fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f  %16.8f
|\n",cvmGet(Rt[i],0,0),cvmGet(Rt[i],0,1),cvmGet(Rt[i],0,2),cvmGet(Rt[i],0,3));
                                fprintf(report,"|   R   | t | = |  %16.8f  %16.8f  %16.8f  %16.8f
|\n",cvmGet(Rt[i],1,0),cvmGet(Rt[i],1,1),cvmGet(Rt[i],1,2),cvmGet(Rt[i],1,3));
                                fprintf(report,"|          |  | %16.8f  %16.8f  %16.8f  %16.8f
|\n",cvmGet(Rt[i],2,0),cvmGet(Rt[i],2,1),cvmGet(Rt[i],2,2),cvmGet(Rt[i],2,3));
                                dtdrefined[i]=Residuals(Rt[i],K,ObjectPoints,ImgPoints);
                                fprintf(report,"errors(dtd) = %f pixels (%4.2f%%)\n",dtdrefined[i],dtdrefined[i]/dtd[i]*100);
                                fprintf(report,"\n");
                        }
                        double improvementNumer=0;
                        double improvementDenom=0;
                        for(i=0;i<Nfiles;i++)
                        {
                                improvementNumer=improvementNumer+dtdrefined[i];
                                improvementDenom=improvementDenom+dtd[i];
                        }
                        fprintf(report,"Improvement = sqrt( sum of refined errors ) / sqrt(sum of initial errors) = %f.\n",sqrt(error/error0));
                }
                fclose(report);
                cvReleaseImage(&img1);
                cvReleaseImage(&temp);
}
```

<Function RefineCamera>
Input : - Rt : 3X4 matrix (R|t)
- K : camera matrix containing intrinsic parameters
- estK : estimated K
- ObjectPoints : object coordinates of the calibration patterns
- ImgPoints : image coordinates of the calibration patterns
- img : image
Output : refined parameters Rt and estK

```
void RefineCamera(CvMat *Rt, CvMat *K, CvMat *estK,CvMat *ObjectPoints, CvMat *ImgPoints, IplImage *img){
	CvMat* J	= cvCreateMat(160,11,CV_64FC1);		// Jacobian Matrix
	CvMat* JT	= cvCreateMat(11,160,CV_64FC1);		// Transposed Jacobian Matrix
	CvMat* JTd	= cvCreateMat(11,1,CV_64FC1);
	CvMat* Hessian = cvCreateMat(11,11,CV_64FC1);	// Hessian Matrix
	CvMat* Hessianinv = cvCreateMat(11,11,CV_64FC1);
	CvMat* I	= cvCreateMat(11,11,CV_64FC1);		// Identity Matrix
	CvMat* Hessian_lm	= cvCreateMat(11,11,CV_64FC1);		// H_lm Matrix
	CvMat* Hessian_lminv = cvCreateMat(11,11,CV_64FC1);		// H_lm Matrix
	CvMat* U	= cvCreateMat(11,11,CV_64FC1);
	CvMat* D	= cvCreateMat(11,11,CV_64FC1);
	CvMat* V	= cvCreateMat(11,11,CV_64FC1);
	CvMat* P	= cvCreateMat(11,1,CV_64FC1);		// Parameters
	CvMat* Ptemp	= cvCreateMat(11,1,CV_64FC1);		// temporary P
	CvMat* dP	= cvCreateMat(11,1,CV_64FC1);		// delta P
	CvMat* R	= cvCreateMat(3,3,CV_64FC1);		// Rotation Matrix of the camera
	CvMat* Rtemp	= cvCreateMat(3,3,CV_64FC1);
	CvMat* t	= cvCreateMat(3,1,CV_64FC1);		// Translation vector of the camera
	CvMat* W	= cvCreateMat(3,1,CV_64FC1);		// Rodrigues representation of R
	CvMat* Wtemp	= cvCreateMat(3,1,CV_64FC1);		// Rodrigues representation of R
	CvMat* d	= cvCreateMat(160,1,CV_64FC1);
	CvMat* dtemp	= cvCreateMat(160,1,CV_64FC1);
	CvMat* Homography	= cvCreateMat(3,3,CV_64FC1);
	CvMat* HomographyTemp	= cvCreateMat(3,3,CV_64FC1);
	double threshold=1e-3;
	double error,error2,delta=100;
	int Iteration=0;
	double lamda=0.01;			// create initial damping factor
	int i,j;
	// setup identity matrix
	for(i=0;i<11;i++)
	{
		for(j=0;j<11;j++)
		{
			if(i==j) cvmSet(I,i,j,1);
			else cvmSet(I,i,j,0);
		}
	}
	// Convert R to W
	cvmSet(R,0,0,cvmGet(Rt,0,0));
	cvmSet(R,1,0,cvmGet(Rt,1,0));
	cvmSet(R,2,0,cvmGet(Rt,2,0));
	cvmSet(R,0,1,cvmGet(Rt,0,1));
	cvmSet(R,1,1,cvmGet(Rt,1,1));
	cvmSet(R,2,1,cvmGet(Rt,2,1));
	cvmSet(R,0,2,cvmGet(Rt,0,2));
	cvmSet(R,1,2,cvmGet(Rt,1,2));
	cvmSet(R,2,2,cvmGet(Rt,2,2));
	cvmSet(t,0,0,cvmGet(Rt,0,3));
	cvmSet(t,1,0,cvmGet(Rt,1,3));
	cvmSet(t,2,0,cvmGet(Rt,2,3));
	R2Rodrigues(R, W);

	// Assign Initial Values
	cvmSet(P,0,0,cvmGet(K,0,0));	//au (alpha)
	cvmSet(P,1,0,cvmGet(K,1,1));	//av (beta)
	cvmSet(P,2,0,cvmGet(K,0,2));	//u0
```

```
cvmSet(P,3,0,cvmGet(K,1,2));    //v0
cvmSet(P,4,0,cvmGet(K,0,1));    //sk (lamda)
cvmSet(P,5,0,cvmGet(W,0,0));    //wx
cvmSet(P,6,0,cvmGet(W,1,0));    //wy
cvmSet(P,7,0,cvmGet(W,2,0));    //wz
cvmSet(P,8,0,cvmGet(t,0,0));    //tx
cvmSet(P,9,0,cvmGet(t,1,0));    //ty
cvmSet(P,10,0,cvmGet(t,2,0));   //tz
// Display Initial parameter values
printf("Initial parameter values\n");
for(i=0;i<11;i++) printf("P(%2d)=%f\n",i,cvmGet(P,i,0));
// Initial geometric distance
error=ErrorsGD(P,ObjectPoints,ImgPoints,80,d,img,0);
// Iterate using LM method
int updateJ=1;
for(j=0;j<200;j++)
{
        if(updateJ==1)
        {
                // determine Jacobian matrix J
                setJ(P,J,ObjectPoints);
                // Hessian Matrix
                cvTranspose(J,JT);
                cvMatMul(JT,J,Hessian);
                // apply damping factor
                cvScaleAdd(I,cvScalar(lamda),Hessian,Hessian_lm);
                cvInvert(Hessian_lm,Hessian_lminv);
        }
        // temporary update P Ptemp=P+inv(H_lm)*(J'*d)
        cvMatMul(JT,d,JTd);
        cvMatMul(Hessian_lminv,JTd,dP);
        cvScaleAdd(dP,cvScalar(-1),P,Ptemp);
        delta=0;
        for(i=0;i<11;i++) delta=delta+pow(cvmGet(dP,i,0),2);
        delta=sqrt(delta)/11;
        // check errors by Ptemp
        error2=ErrorsGD(Ptemp,ObjectPoints,ImgPoints,80,d,img,0);
        if(error2<error) //decrease damping factor and update
        {
                lamda=lamda/10;
                for(i=0;i<11;i++) cvmSet(P,i,0,cvmGet(Ptemp,i,0));
                updateJ=1;
        }
        else  // increase damping factor and try again
        {
                updateJ=0;
                lamda=lamda*10;
        }
        if(delta<threshold) break;
}
printf("Number of iteration = %d\n",j);
printf("delta = %f\n",delta);
// Display Refined parameter values
printf("Refined parameter values\n");
for(i=0;i<11;i++) printf("P(%2d)=%f\n",i,cvmGet(P,i,0));
// Finally estimated K
cvmSet(estK,0,0,cvmGet(P,0,0));
cvmSet(estK,1,0,cvmGet(K,1,0));
cvmSet(estK,2,0,cvmGet(K,2,0));
cvmSet(estK,0,1,cvmGet(P,4,0));
cvmSet(estK,1,1,cvmGet(P,1,0));
cvmSet(estK,2,1,cvmGet(K,2,1));
cvmSet(estK,0,2,cvmGet(P,2,0));
cvmSet(estK,1,2,cvmGet(P,3,0));
cvmSet(estK,2,2,cvmGet(K,2,2));

// Update Rt
cvmSet(W,0,0,cvmGet(P,5,0));
cvmSet(W,1,0,cvmGet(P,6,0));
cvmSet(W,2,0,cvmGet(P,7,0));
```

```
	Rodrigues2R(W,R);
	cvmSet(Rt,0,0,cvmGet(R,0,0));
	cvmSet(Rt,1,0,cvmGet(R,1,0));
	cvmSet(Rt,2,0,cvmGet(R,2,0));
	cvmSet(Rt,0,1,cvmGet(R,0,1));
	cvmSet(Rt,1,1,cvmGet(R,1,1));
	cvmSet(Rt,2,1,cvmGet(R,2,1));
	cvmSet(Rt,0,2,cvmGet(R,0,2));
	cvmSet(Rt,1,2,cvmGet(R,1,2));
	cvmSet(Rt,2,2,cvmGet(R,2,2));
	cvmSet(Rt,0,3,cvmGet(P,8,0));
	cvmSet(Rt,1,3,cvmGet(P,9,0));
	cvmSet(Rt,2,3,cvmGet(P,10,0));
}
```

<Function Edge_HoughCanny>

Input : - src : source image
- Corners : 2 dimensional double array for corners in the image
- viewHoughFlag : if 1, show lines detected by the Hough transformation
- viewRefiedHoughFlag : if 1, show refined lines

Output : - Corners (sorted)

```
void Edge_HoughCanny(IplImage *src, double **Corners, int viewEdgeFlag, int viewHoughFlag, int viewRefiedHoughFlag){
   IplImage* dst = cvCreateImage( cvGetSize(src), 8, 1 );
   IplImage* color_dst = cvCreateImage( cvGetSize(src), 8, 3 );
	IplImage* color_dst2 = cvCreateImage( cvGetSize(src), 8, 3 );
   CvMemStorage* storage = cvCreateMemStorage(0);
   CvSeq* lines = 0;
	CvPoint pt1,pt2,pt3,pt4;
   int i,j;
   cvCanny( src, dst, 50, 500, 3 );
	if(viewEdgeFlag==1)
	{
		cvNamedWindow( "Canny", 1 );
		cvShowImage( "Canny", dst );
	}
   cvCvtColor( dst, color_dst, CV_GRAY2BGR );
	cvCvtColor( src, color_dst2, CV_GRAY2BGR );
	#if 1
		lines = cvHoughLines2( dst, storage, CV_HOUGH_STANDARD, .5, CV_PI/1000, 40, 0, 0 );
		// Grouping lines
		dnode *ptr;
		init_list();
		int N=0;
		int Ngroup=0;
		int *Index;
		double rho_g,theta_g,num_g;
		double threshold=20;
		Index=new int [lines->total];
		for(i=0;i<lines->total;i++) Index[i]=0;
		while(N<lines->total)
		{
			for( i = 0; i < lines->total; i++ )
			{
				float* line = (float*)cvGetSeqElem(lines,i);
				float rho = line[0];
				float theta = line[1];
				double a = cos(theta), b = sin(theta), c = tan(theta);
				if( fabs(b) < 0.001 )
				{
					pt1.x = pt2.x = cvRound(rho);
					pt1.y = 0;
					pt2.y = color_dst->height;
				}
				else if( fabs(a) < 0.001 )
```

```
{
		pt1.y = pt2.y = cvRound(rho);
		pt1.x = 0;
		pt2.x = color_dst->width;
}
else
{
		pt1.x = 0;
		pt1.y = cvRound(rho/b);

		if(pt1.y < 0)
		{
			pt1.x = cvRound(rho / a);
			pt1.y = 0;
		}

		if(pt1.y > color_dst->height)
		{
			pt1.x = cvRound((pt1.y - color_dst->height)*c);
			pt1.y = color_dst->height;
		}
		pt2.x = color_dst->width;
		pt2.y = cvRound(rho/b - color_dst->width/c);
		if(pt2.y < 0)
		{
			pt2.x = cvRound(rho/a);
			pt2.y = 0;
		}
		if(pt2.y > color_dst->height)
		{
			pt2.x = cvRound(-1.0 * ((color_dst->height - rho/b) * c));
			pt2.y = color_dst->height;
		}
}
if(Index[i]==0)
{
		rho_g=0;theta_g=0;num_g=0;
		//printf("point %d is grouped with \n",i);
		for( j = i; j < lines->total; j++ )
		{
			float* line2 = (float*)cvGetSeqElem(lines,j);
			float rho2 = line2[0];
			float theta2 = line2[1];
			double a = cos(theta2), b = sin(theta2), c = tan(theta2);
			if( fabs(b) < 0.001 )
			{
				pt3.x = pt4.x = cvRound(rho2);
				pt3.y = 0;
				pt4.y = color_dst->height;
			}
			else if( fabs(a) < 0.001 )
			{
				pt3.y = pt4.y = cvRound(rho2);
				pt3.x = 0;
				pt4.x = color_dst->width;
			}
			else
			{
				pt3.x = 0;
				pt3.y = cvRound(rho2/b);

				if(pt3.y < 0)
				{
					pt3.x = cvRound(rho2 / a);
					pt3.y = 0;
				}

				if(pt3.y > color_dst->height)
				{
					pt3.x = cvRound((pt3.y - color_dst->height)*c);
```

```
                                                pt3.y = color_dst->height;
                                        }
                                        pt4.x = color_dst->width;
                                        pt4.y = cvRound(rho2/b - color_dst->width/c);
                                        if(pt4.y < 0)
                                        {
                                                pt4.x = cvRound(rho2/a);
                                                pt4.y = 0;
                                        }
                                        if(pt4.y > color_dst->height)
                                        {
                                                pt4.x = cvRound(-1.0 * ((color_dst->height - rho2/b) *
c));
                                                pt4.y = color_dst->height;
                                        }
                                }
                                if (abs(pt1.x-pt3.x)<threshold && abs(pt1.y-pt3.y)<threshold && abs(pt2.x-
pt4.x)<threshold && abs(pt2.y-pt4.y)<threshold && fabs(theta-theta2)<CV_PI*5/180 && fabs(rho-rho2)<threshold)
                                {
                                        rho_g=rho_g+rho2;
                                        theta_g=theta_g+theta2;
                                        num_g++;
                                        Index[j]=1;
                                        //printf("%d,",j);
                                }
                                else if (abs(pt1.x-pt4.x)<threshold && abs(pt1.y-pt4.y)<threshold &&
abs(pt2.x-pt3.x)<threshold && abs(pt2.y-pt3.y)<threshold && fabs(theta-theta2)<CV_PI*5/180 && fabs(rho-rho2)<threshold)
                                {
                                        rho_g=rho_g+rho2;
                                        theta_g=theta_g+theta2;
                                        num_g++;
                                        Index[j]=1;
                                        //printf("%d,",j);
                                }
                                else if (abs(pt1.x-pt3.x)<threshold && abs(pt1.y-pt3.y)<threshold &&
abs(pt2.x-pt4.x)<threshold && abs(pt2.y-pt4.y)<threshold && fabs(theta-theta2)>CV_PI*175/180 && fabs(rho+rho2)<threshold)
                                {
                                        if(rho<0 && theta>CV_PI*175/180)
                                        {
                                                rho_g=rho_g-rho2;
                                                theta_g=theta_g+CV_PI-theta2;
                                        }
                                        if(rho>0 && theta<CV_PI*5/180)
                                        {
                                                rho_g=rho_g-rho2;
                                                theta_g=theta_g+theta2-CV_PI;
                                        }
                                        num_g++;
                                        Index[j]=1;
                                        //printf("%d,",j);
                                }
                                else if (abs(pt1.x-pt4.x)<threshold && abs(pt1.y-pt4.y)<threshold &&
abs(pt2.x-pt3.x)<threshold && abs(pt2.y-pt3.y)<threshold && fabs(theta-theta2)>CV_PI*175/180 && fabs(rho+rho2)<threshold)
                                {
                                        if(rho<0 && theta>CV_PI*175/180)
                                        {
                                                rho_g=rho_g-rho2;
                                                theta_g=theta_g+CV_PI-theta2;
                                        }
                                        if(rho>0 && theta<CV_PI*5/180)
                                        {
                                                rho_g=rho_g-rho2;
                                                theta_g=theta_g+theta2-CV_PI;
                                        }
                                        num_g++;
                                        Index[j]=1;
                                        //printf("%d,",j);
                                }
                        }
                        //printf("\n");
```

```
                                        if(num_g>0){
                                                insert(rho_g/num_g,theta_g/num_g,0,0);
                                                Ngroup++;
                                        }
                                }
                        }
                        N=0; for( j = 0; j < lines->total; j++ ) N=N+Index[j];
                }
                ptr=head->next;
                N=0;
                while(N<Ngroup)
                {
                        double rho = ptr->x1;
                        double theta = ptr->y1;
                        double a = cos(theta), b = sin(theta), c = tan(theta);
                        if( fabs(b) < 0.001 )
                        {
                                pt1.x = pt2.x = cvRound(rho);
                                pt1.y = 0;
                                pt2.y = color_dst->height;
                        }
                        else if( fabs(a) < 0.001 )
                        {
                                pt1.y = pt2.y = cvRound(rho);
                                pt1.x = 0;
                                pt2.x = color_dst->width;
                        }
                        else
                        {
                                pt1.x = 0;
                                pt1.y = cvRound(rho/b);

                                if(pt1.y < 0)
                                {
                                        pt1.x = cvRound(rho / a);
                                        pt1.y = 0;
                                }

                                if(pt1.y > color_dst->height)
                                {
                                        pt1.x = cvRound((pt1.y - color_dst->height)*c);
                                        pt1.y = color_dst->height;
                                }
                                pt2.x = color_dst->width;
                                pt2.y = cvRound(rho/b - color_dst->width/c);
                                if(pt2.y < 0)
                                {
                                        pt2.x = cvRound(rho/a);
                                        pt2.y = 0;
                                }
                                if(pt2.y > color_dst->height)
                                {
                                        pt2.x = cvRound(-1.0 * ((color_dst->height - rho/b) * c));
                                        pt2.y = color_dst->height;
                                }
                        }
                        //printf("j=%2d] rho=%10.3f, theta=%10.3f, x=%3d,y=%3d <-->
x=%3d,y=%3d\n",N,rho,theta,pt1.x,pt1.y,pt2.x,pt2.y);
                        cvLine( color_dst2, pt1, pt2, CV_RGB(255,0,0), 1, 8, 0 );
                        ptr=ptr->next;
                        N++;
                }
                printf("initial=%d, final=%d\n",lines->total,N);
                if(viewRefiedHoughFlag==1)
                {
                        cvNamedWindow( "Hough_refined", 1 );
                        cvShowImage( "Hough_refined", color_dst2 );
                }
                // Grouping Vertical lines and Horizontal lines
                threshold=15;
```

```
ptr=head->next;
double init_rho=ptr->x1;
double init_theta=ptr->y1;
Ngroup=1;
for(i=1;i<N;i++)
{
	ptr=ptr->next;
	if(fabs(ptr->y1-init_theta)<CV_PI*threshold/180 || fabs(ptr->y1-init_theta)>CV_PI*(180-threshold)/180)
	{
		Ngroup++;
	}
}
double Vertical[8][2];
double Horizontal[10][2];
if(Ngroup==8) // vertical lines
{
	ptr=head->next;
	Vertical[0][0]=ptr->x1;
	Vertical[0][1]=ptr->y1;
	Ngroup=1;
	for(i=1;i<N;i++)
	{
		ptr=ptr->next;
		if(fabs(ptr->y1-init_theta)<CV_PI*threshold/180 || fabs(ptr->y1-init_theta)>CV_PI*(180-
threshold)/180)
		{
			Vertical[Ngroup][0]=ptr->x1;
			Vertical[Ngroup][1]=ptr->y1;
			Ngroup++;
		}
		else
		{
			Horizontal[i-Ngroup][0]=ptr->x1;
			Horizontal[i-Ngroup][1]=ptr->y1;
		}
	}
}
else if (Ngroup==10)
{
	ptr=head->next;
	Horizontal[0][0]=ptr->x1;
	Horizontal[0][1]=ptr->y1;
	Ngroup=1;
	for(i=1;i<N;i++)
	{
		ptr=ptr->next;
		if(fabs(ptr->y1-init_theta)<CV_PI*threshold/180 || fabs(ptr->y1-init_theta)>CV_PI*(180-
threshold)/180)
		{
			Horizontal[Ngroup][0]=ptr->x1;
			Horizontal[Ngroup][1]=ptr->y1;
			Ngroup++;
		}
		else
		{
			Vertical[i-Ngroup][0]=ptr->x1;
			Vertical[i-Ngroup][1]=ptr->y1;
		}
	}
}
else
{
	printf("Line grouping failed.\n");
	printf("re-assign thereshold values.\n");
	exit;
}
// sorting Vertical lines wrt rho values
double temp[10][2],rho,rho2;
int sort;
for(i=0;i<8;i++)
```

```
{
		temp[i][0]=Vertical[i][0];
		temp[i][1]=Vertical[i][1];
}
for(i=0;i<8;i++)
{
		sort=0;
		rho=fabs(temp[i][0]);
		for(j=0;j<8;j++)
		{
			rho2=fabs(temp[j][0]);
			if(rho>rho2)
			{
				sort++;
			}
		}
		Vertical[sort][0]=temp[i][0];
		Vertical[sort][1]=temp[i][1];
}
// sorting Horizontal lines wrt rho values
for(i=0;i<10;i++)
{
		temp[i][0]=Horizontal[i][0];
		temp[i][1]=Horizontal[i][1];
}
for(i=0;i<10;i++)
{
		sort=0;
		rho=fabs(temp[i][0]);
		for(j=0;j<10;j++)
		{
			rho2=fabs(temp[j][0]);
			if(rho>rho2)
			{
				sort++;
			}
		}
		Horizontal[sort][0]=temp[i][0];
		Horizontal[sort][1]=temp[i][1];
}
// intersect lines
CvMat* A  = cvCreateMat(2,2,CV_64FC1);
CvMat* Ainv  = cvCreateMat(2,2,CV_64FC1);
CvMat* x  = cvCreateMat(2,1,CV_64FC1);
CvMat* f  = cvCreateMat(2,1,CV_64FC1);
for(i=0;i<10;i++) //horizontal lines
{
		for(j=0;j<8;j++) // vertical lines
		{
			//setup Ax=f
			cvmSet(A,0,0,cos(Horizontal[i][1]));
			cvmSet(A,0,1,sin(Horizontal[i][1]));
			cvmSet(A,1,0,cos(Vertical[j][1]));
			cvmSet(A,1,1,sin(Vertical[j][1]));
			cvmSet(f,0,0,Horizontal[i][0]);
			cvmSet(f,1,0,Vertical[j][0]);
			// x=Ainv*f
			cvInvert(A,Ainv);
			cvMatMul(Ainv,f,x);
			Corners[i*8+j][0]=cvmGet(x,0,0)+.5;
			Corners[i*8+j][1]=cvmGet(x,1,0)+.5;
		}
}
// initial values from the openCV Hough
for( i = 0; i < lines->total; i++ )
{
		float* line = (float*)cvGetSeqElem(lines,i);
		float rho = line[0];
		float theta = line[1];
		double a = cos(theta), b = sin(theta), c = tan(theta);
```

```
                        if( fabs(b) < 0.001 )
                        {
                                pt1.x = pt2.x = cvRound(rho);
                                pt1.y = 0;
                                pt2.y = color_dst->height;
                        }
                        else if( fabs(a) < 0.001 )
                        {
                                pt1.y = pt2.y = cvRound(rho);
                                pt1.x = 0;
                                pt2.x = color_dst->width;
                        }
                        else
                        {
                                pt1.x = 0;
                                pt1.y = cvRound(rho/b);

                                if(pt1.y < 0)
                                {
                                        pt1.x = cvRound(rho / a);
                                        pt1.y = 0;
                                }

                                if(pt1.y > color_dst->height)
                                {
                                        pt1.x = cvRound((pt1.y - color_dst->height)*c);
                                        pt1.y = color_dst->height;
                                }

                                pt2.x = color_dst->width;
                                pt2.y = cvRound(rho/b - color_dst->width/c);
                                if(pt2.y < 0)
                                {
                                        pt2.x = cvRound(rho/a);
                                        pt2.y = 0;
                                }
                                if(pt2.y > color_dst->height)
                                {
                                        pt2.x = cvRound(-1.0 * ((color_dst->height - rho/b) * c));
                                        pt2.y = color_dst->height;
                                }
                        }
                        cvLine( color_dst, pt1, pt2, CV_RGB(255,0,0), 1, 8, 0 );
                }
                if(viewHoughFlag==1)
                {
                        cvNamedWindow( "Hough_initial", 1 );
                        cvShowImage( "Hough_initial", color_dst );
                }
        #endif
        cvReleaseImage(&dst);
        cvReleaseImage(&color_dst);
        cvReleaseImage(&color_dst2);

}
```

<Function SetV>

Set up V matrix to estimate the absolute conic B

H is the homography of the *i*-th image

```
void SetV(CvMat*H, int i, CvMat *V){
        double h11,h12,h13,h21,h22,h23,h31,h32,h33;
        h11=cvmGet(H,0,0);
        h12=cvmGet(H,1,0);
        h13=cvmGet(H,2,0);
        h21=cvmGet(H,0,1);
        h22=cvmGet(H,1,1);
        h23=cvmGet(H,2,1);
```

```
		h31=cvmGet(H,0,2);
		h32=cvmGet(H,1,2);
		h33=cvmGet(H,2,2);
		cvmSet(V,2*i,0,h11*h21);
		cvmSet(V,2*i,1,h11*h22+h12*h21);
		cvmSet(V,2*i,2,h12*h22);
		cvmSet(V,2*i,3,h13*h21+h11*h23);
		cvmSet(V,2*i,4,h13*h22+h12*h23);
		cvmSet(V,2*i,5,h13*h23);
		cvmSet(V,2*i+1,0,h11*h11-h21*h21);
		cvmSet(V,2*i+1,1,h11*h12+h12*h11-h21*h22-h22*h21);
		cvmSet(V,2*i+1,2,h12*h12-h22*h22);
		cvmSet(V,2*i+1,3,h13*h11+h11*h13-h23*h21-h21*h23);
		cvmSet(V,2*i+1,4,h13*h12+h12*h13-h23*h22-h22*h23);
		cvmSet(V,2*i+1,5,h13*h13-h23*h23);
}
```

<function R2Rodrigues & Rodrigues2R>

Convert rotation matrix into Rodrigues representation and its inverse function

```
void R2Rodrigues(CvMat *R, CvMat *W){
		double norm;
		norm=acos((cvmGet(R,0,0)+cvmGet(R,1,1)+cvmGet(R,2,2)-1)/2);
		cvmSet(W,0,0,norm/(2*sin(norm))*(cvmGet(R,2,1)-cvmGet(R,1,2)));
		cvmSet(W,1,0,norm/(2*sin(norm))*(cvmGet(R,0,2)-cvmGet(R,2,0)));
		cvmSet(W,2,0,norm/(2*sin(norm))*(cvmGet(R,1,0)-cvmGet(R,0,1)));
}
void Rodrigues2R(CvMat *W, CvMat *R){
		double norm;
		CvMat* Wx   = cvCreateMat(3,3,CV_64FC1);
		CvMat* Wx2  = cvCreateMat(3,3,CV_64FC1);
		CvMat* I    = cvCreateMat(3,3,CV_64FC1);
		CvMat* temp = cvCreateMat(3,3,CV_64FC1);
		cvmSet(I,0,0,1);
		cvmSet(I,1,0,0);
		cvmSet(I,2,0,0);
		cvmSet(I,0,1,0);
		cvmSet(I,1,1,1);
		cvmSet(I,2,1,0);
		cvmSet(I,0,2,0);
		cvmSet(I,1,2,0);
		cvmSet(I,2,2,1);
		// ||W||
		norm=sqrt(cvmGet(W,0,0)*cvmGet(W,0,0)+cvmGet(W,1,0)*cvmGet(W,1,0)+cvmGet(W,2,0)*cvmGet(W,2,0));
		// R=I+sin(||W||)/||W||*Wx+(1-cos(||W||)/(||W||^2)*Wx2
		cvmSet(Wx,0,0,0);
		cvmSet(Wx,1,0,cvmGet(W,2,0));
		cvmSet(Wx,2,0,-cvmGet(W,1,0));
		cvmSet(Wx,0,1,-cvmGet(W,2,0));
		cvmSet(Wx,1,1,0);
		cvmSet(Wx,2,1,cvmGet(W,0,0));
		cvmSet(Wx,0,2,cvmGet(W,1,0));
		cvmSet(Wx,1,2,-cvmGet(W,0,0));
		cvmSet(Wx,2,2,0);
		cvMatMul(Wx,Wx,Wx2);
		//CvScalar CoeffWx,CoeffWx2;
		cvScaleAdd(Wx,cvRealScalar(sin(norm)/norm),I,temp);
		cvScaleAdd(Wx2,cvRealScalar((1-cos(norm))/(norm*norm)),temp,R);
}
```

< Function setJall >

Construct Jacobian matrix for the image w.r.t parameter vector P and ObjectPoints (for the entire images)

```
void setJall(CvMat *P,CvMat *J, int Nfiles, CvMat *ObjectPoints){
		// P : 5+6*Nfiles
		int i,j;
```

```
	double u0, v0, au, av, sk;
	double tx, ty, tz, wx, wy, wz;
	double X, Y;
	// dummy variables
	double
MapleGenVar1,MapleGenVar2,MapleGenVar3,MapleGenVar4,MapleGenVar5,MapleGenVar6,MapleGenVar7,MapleGenVar8,MapleGenVar9,
MapleGenVar10;
	au=cvmGet(P,0,0);
	av=cvmGet(P,1,0);
	u0=cvmGet(P,2,0);
	v0=cvmGet(P,3,0);
	sk=cvmGet(P,4,0);
	for(i=0;i<160*Nfiles;i++) for(j=0;j<5+6*Nfiles;j++) cvmSet(J,i,j,0);
	for(j=0;j<Nfiles;j++)
	{
		wx=cvmGet(P,5+6*j,0);
		wy=cvmGet(P,6+6*j,0);
		wz=cvmGet(P,7+6*j,0);
		tx=cvmGet(P,8+6*j,0);
		ty=cvmGet(P,9+6*j,0);
		tz=cvmGet(P,10+6*j,0);
		for(i=0;i<80;i++)
		{
			X=cvmGet(ObjectPoints,0,i);
			Y=cvmGet(ObjectPoints,1,i);
			// Setup Jacobian matrix
			// intrinsic parameters
			cvmSet(J,160*j+2*i   , 0+j*6 , -((1.0+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-
wz*wz-wy*wy))*X+(-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)*Y+tx)/((-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));
			cvmSet(J,160*j+2*i+1 , 0+j*6 , 0 );
			cvmSet(J,160*j+2*i   , 1+j*6 , 0 );
			cvmSet(J,160*j+2*i+1 , 1+j*6 , -((sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)*X+(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-wx*wx))*Y+ty)/((-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));
			cvmSet(J,160*j+2*i   , 2+j*6 , -1);
			cvmSet(J,160*j+2*i+1 , 2+j*6 , 0);
			cvmSet(J,160*j+2*i   , 3+j*6 , 0);
			cvmSet(J,160*j+2*i+1 , 3+j*6 , -1);
			cvmSet(J,160*j+2*i   , 4+j*6 , -((sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)*X+(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-wx*wx))*Y+ty)/((-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));
			cvmSet(J,160*j+2*i+1 , 4+j*6 , 0);
			// extrinsic parameters
			MapleGenVar2 = -1.0;
			MapleGenVar7 = au*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*(-
wz*wz-wy*wy)-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wy*wy)*wx);
			MapleGenVar9 = sk*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wx*wx*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy);
			MapleGenVar10 = u0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz);
			MapleGenVar8 = MapleGenVar9+MapleGenVar10;
			MapleGenVar6 = MapleGenVar7+MapleGenVar8;
			MapleGenVar7 = X;
			MapleGenVar5 = MapleGenVar6*MapleGenVar7;
```

```
                                MapleGenVar8 = au*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy);
                                MapleGenVar9 = sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*(-
wz*wz-wx*wx)-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wx*wx)*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx)+u0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wx-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
                                MapleGenVar7 = MapleGenVar8+MapleGenVar9;
                                MapleGenVar8 = Y;
                                MapleGenVar6 = MapleGenVar7*MapleGenVar8;
                                MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                                MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                                MapleGenVar3 = MapleGenVar4*MapleGenVar5;
                                MapleGenVar1 = MapleGenVar2*MapleGenVar3;
                                MapleGenVar5 = (au*(1.0+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wy*wy))+sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+u0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X;
                                MapleGenVar6 = (au*(-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+sk*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+u0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y;
                                MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                                MapleGenVar3 = MapleGenVar4+au*tx+sk*ty+u0*tz;
                                MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                                MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)*X;
                                MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wx-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx)*Y;
                                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                                MapleGenVar4 = MapleGenVar5*MapleGenVar6;
                                MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                                cvmSet(J,160*j+2*i   , 5+j*6 , MapleGenVar1+MapleGenVar2);

                                MapleGenVar2 = -1.0;
                                MapleGenVar7 = av*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wx*wx*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy);
                                MapleGenVar8 = v0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz);
                                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                                MapleGenVar7 = X;
                                MapleGenVar5 = MapleGenVar6*MapleGenVar7;
                                MapleGenVar6 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*(-
wz*wz-wx*wx)-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wx*wx)*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx)+v0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wx-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx))*Y;
                                MapleGenVar4 = MapleGenVar5+MapleGenVar6;
```

```
                                MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                                MapleGenVar3 = MapleGenVar4*MapleGenVar5;
                                MapleGenVar1 = MapleGenVar2*MapleGenVar3;
                                MapleGenVar3 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+v0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X+(av*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+v0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y+av*ty+v0*tz;
                                MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                                MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)*X;
                                MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wx-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx)*Y;
                                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                                MapleGenVar4 = MapleGenVar5*MapleGenVar6;
                                MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                                cvmSet(J,160*j+2*i+1 , 5+j*6 , MapleGenVar1+MapleGenVar2);

                                MapleGenVar2 = -1.0;
                                MapleGenVar7 = au*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*(-
wz*wz-wy*wy)-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wy*wy)*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy);

                                MapleGenVar9 = sk*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wy*wy*wx-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wy*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx);
                                MapleGenVar10 = u0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx
*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
                                MapleGenVar8 = MapleGenVar9+MapleGenVar10;
                                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                                MapleGenVar7 = X;
                                MapleGenVar5 = MapleGenVar6*MapleGenVar7;
                                MapleGenVar8 = au*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*wy*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wy*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx);
                                MapleGenVar9 = sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*(-
wz*wz-wx*wx)-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-
wx*wx)*wy)+u0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*w
y+wz*wz,3.0))*wy*wy*wz-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz);
                                MapleGenVar7 = MapleGenVar8+MapleGenVar9;
                                MapleGenVar8 = Y;
                                MapleGenVar6 = MapleGenVar7*MapleGenVar8;
                                MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                                MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                                MapleGenVar3 = MapleGenVar4*MapleGenVar5;
                                MapleGenVar1 = MapleGenVar2*MapleGenVar3;
                                MapleGenVar5 = (au*(1.0+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wy*wy))+sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+u0*(-
```

```
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X;
                                MapleGenVar6 = (au*(-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+sk*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+u0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y;
                                MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                                MapleGenVar3 = MapleGenVar4+au*tx+sk*ty+u0*tz;
                                MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                                MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx
*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx)*X;
                                MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*w
y+wz*wz,3.0))*wy*wy*wz-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)*Y;
                                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                                MapleGenVar4 = MapleGenVar5*MapleGenVar6;
                                MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                                cvmSet(J,160*j+2*i   , 5+j*6+1 , MapleGenVar1+MapleGenVar2);

                                MapleGenVar2 = -1.0;
                                MapleGenVar7 = av*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wy*wy*wx-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wy*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx);
                                MapleGenVar8 = v0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx
*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
                                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                                MapleGenVar7 = X;
                                MapleGenVar5 = MapleGenVar6*MapleGenVar7;
                                MapleGenVar6 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*(-
wz*wz-wx*wx)-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-
wx*wx)*wy)+v0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*w
y+wz*wz,3.0))*wy*wy*wz-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz))*Y;
                                MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                                MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                                MapleGenVar3 = MapleGenVar4*MapleGenVar5;
                                MapleGenVar1 = MapleGenVar2*MapleGenVar3;
                                MapleGenVar3 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+v0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X+(av*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+v0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y+av*ty+v0*tz;
                                MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                                MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx
*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx)*X;
                                MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*w
y+wz*wz,3.0))*wy*wy*wz-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)*Y;
```

```
                                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                                MapleGenVar4 = MapleGenVar5*MapleGenVar6;
                                MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                                cvmSet(J,160*j+2*i+1 , 5+j*6+1 , MapleGenVar1+MapleGenVar2);

                                MapleGenVar2 = -1.0;
                                MapleGenVar7 = au*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*(-
wz*wz-wy*wy)-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wy*wy)*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz);

                                MapleGenVar9 = sk*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wz*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
                                MapleGenVar10 = u0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx);
                                MapleGenVar8 = MapleGenVar9+MapleGenVar10;
                                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                                MapleGenVar7 = X;
                                MapleGenVar5 = MapleGenVar6*MapleGenVar7;
                                MapleGenVar8 = au*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wz*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx
*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
                                MapleGenVar9 = sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*(-
wz*wz-wx*wx)-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wx*wx)*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)+u0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wz*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy);
                                MapleGenVar7 = MapleGenVar8+MapleGenVar9;
                                MapleGenVar8 = Y;
                                MapleGenVar6 = MapleGenVar7*MapleGenVar8;
                                MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                                MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                                MapleGenVar3 = MapleGenVar4*MapleGenVar5;
                                MapleGenVar1 = MapleGenVar2*MapleGenVar3;
                                MapleGenVar5 = (au*(1.0+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wy*wy))+sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+u0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X;
                                MapleGenVar6 = (au*(-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+sk*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+u0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y;
                                MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                                MapleGenVar3 = MapleGenVar4+au*tx+sk*ty+u0*tz;
                                MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                                MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx)*X;
                                MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wz*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy)*Y;
                                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                                MapleGenVar4 = MapleGenVar5*MapleGenVar6;
                                MapleGenVar2 = MapleGenVar3*MapleGenVar4;
```

```
cvmSet(J,160*j+2*i   , 5+j*6+2 , MapleGenVar1+MapleGenVar2);

MapleGenVar2 = -1.0;
MapleGenVar7 = av*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wz*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
MapleGenVar8 = v0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx);
MapleGenVar6 = MapleGenVar7+MapleGenVar8;
MapleGenVar7 = X;
MapleGenVar5 = MapleGenVar6*MapleGenVar7;
MapleGenVar6 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*(-
wz*wz-wx*wx)-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wx*wx)*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)+v0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wz*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy))*Y;
MapleGenVar4 = MapleGenVar5+MapleGenVar6;
MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
MapleGenVar3 = MapleGenVar4*MapleGenVar5;
MapleGenVar1 = MapleGenVar2*MapleGenVar3;
MapleGenVar3 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+v0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X+(av*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+v0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y+av*ty+v0*tz;
MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx)*X;
MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wz*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy)*Y;
MapleGenVar6 = MapleGenVar7+MapleGenVar8;
MapleGenVar4 = MapleGenVar5*MapleGenVar6;
MapleGenVar2 = MapleGenVar3*MapleGenVar4;
cvmSet(J,160*j+2*i+1 , 5+j*6+2 , MapleGenVar1+MapleGenVar2);

cvmSet(J,160*j+2*i   , 5+j*6+3 , -au/((-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));
cvmSet(J,160*j+2*i+1 , 5+j*6+3 , 0);
cvmSet(J,160*j+2*i   , 5+j*6+4 , -sk/((-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));
cvmSet(J,160*j+2*i+1 , 5+j*6+4 , -av/((-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));

MapleGenVar1 = -u0/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
MapleGenVar5 = (au*(1.0+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wy*wy))+sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
```

```
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+u0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X;
                                MapleGenVar6 = (au*(-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+sk*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+u0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y;
                                MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                                MapleGenVar3 = MapleGenVar4+au*tx+sk*ty+u0*tz;
                                MapleGenVar4 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                                MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                                cvmSet(J,160*j+2*i   , 5+j*6+5, MapleGenVar1+MapleGenVar2);
                                MapleGenVar1 = -v0/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                                MapleGenVar3 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+v0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X+(av*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+v0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y+av*ty+v0*tz;
                                MapleGenVar4 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                                MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                                cvmSet(J,160*j+2*i+1 , 5+j*6+5, MapleGenVar1+MapleGenVar2);
                        }
                }
}
```

< Function setJ >

Construct Jacobian matrix for the image w.r.t parameter vector P and ObjectPoints (for a single image)

```
void setJ(CvMat *P,CvMat *J, CvMat *ObjectPoints){
        int i;
        double u0, v0, au, av, sk;
        double tx, ty, tz, wx, wy, wz;
        double X, Y;
        // dummy variables
        double
MapleGenVar1,MapleGenVar2,MapleGenVar3,MapleGenVar4,MapleGenVar5,MapleGenVar6,MapleGenVar7,MapleGenVar8,MapleGenVar9,
MapleGenVar10;
        au=cvmGet(P,0,0);
        av=cvmGet(P,1,0);
        u0=cvmGet(P,2,0);
        v0=cvmGet(P,3,0);
        sk=cvmGet(P,4,0);
        wx=cvmGet(P,5,0);
        wy=cvmGet(P,6,0);
        wz=cvmGet(P,7,0);
        tx=cvmGet(P,8,0);
        ty=cvmGet(P,9,0);
        tz=cvmGet(P,10,0);
        for(i=0;i<80;i++)
        {
                X=cvmGet(ObjectPoints,0,i);
                Y=cvmGet(ObjectPoints,1,i);
                // Setup Jacobian matrix
                cvmSet(J,2*i   , 0 , -((1.0+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-wy*wy))*X+(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)*Y+tx)/((-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
```

```
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));
                    cvmSet(J,2*i+1 , 0 , 0 );
                    cvmSet(J,2*i   , 1 , 0 );
                    cvmSet(J,2*i+1 , 1 , -((sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)*X+(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-wx*wx))*Y+ty)/((-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));
                    cvmSet(J,2*i   , 2 , -1);
                    cvmSet(J,2*i+1 , 2 , 0);
                    cvmSet(J,2*i   , 3 , 0);
                    cvmSet(J,2*i+1 , 3 , -1);
                    cvmSet(J,2*i   , 4 , -((sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)*X+(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-wx*wx))*Y+ty)/((-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));
                    cvmSet(J,2*i+1 , 4 , 0);
                    MapleGenVar2 = -1.0;
                    MapleGenVar7 = au*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*(-wz*wz-wy*wy)-
2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wy*wy)*wx);
                    MapleGenVar9 = sk*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wx*wx*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy);
                    MapleGenVar10 = u0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz);
                    MapleGenVar8 = MapleGenVar9+MapleGenVar10;
                    MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                    MapleGenVar7 = X;
                    MapleGenVar5 = MapleGenVar6*MapleGenVar7;
                    MapleGenVar8 = au*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy);
                    MapleGenVar9 = sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*(-wz*wz-wx*wx)-
2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wx*wx)*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx)+u0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wx-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
                    MapleGenVar7 = MapleGenVar8+MapleGenVar9;
                    MapleGenVar8 = Y;
                    MapleGenVar6 = MapleGenVar7*MapleGenVar8;
                    MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                    MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                    MapleGenVar3 = MapleGenVar4*MapleGenVar5;
                    MapleGenVar1 = MapleGenVar2*MapleGenVar3;
                    MapleGenVar5 = (au*(1.0+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wy*wy))+sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+u0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X;
                    MapleGenVar6 = (au*(-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+sk*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+u0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y;
                    MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                    MapleGenVar3 = MapleGenVar4+au*tx+sk*ty+u0*tz;
```

```
                MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)*X;
                MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wx-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx)*Y;
                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                MapleGenVar4 = MapleGenVar5*MapleGenVar6;
                MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                cvmSet(J,2*i   , 5 , MapleGenVar1+MapleGenVar2);

                MapleGenVar2 = -1.0;
                MapleGenVar7 = av*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wx*wx*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy);
                MapleGenVar8 = v0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz);
                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                MapleGenVar7 = X;
                MapleGenVar5 = MapleGenVar6*MapleGenVar7;
                MapleGenVar6 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*(-wz*wz-wx*wx)-
2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wx*wx)*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx)+v0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wx-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx))*Y;
                MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                MapleGenVar3 = MapleGenVar4*MapleGenVar5;
                MapleGenVar1 = MapleGenVar2*MapleGenVar3;
                MapleGenVar3 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+v0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X+(av*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+v0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y+av*ty+v0*tz;
                MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wx*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)*X;
                MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wx-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx)*Y;
                MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                MapleGenVar4 = MapleGenVar5*MapleGenVar6;
                MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                cvmSet(J,2*i+1 , 5 , MapleGenVar1+MapleGenVar2);

                MapleGenVar2 = -1.0;
                MapleGenVar7 = au*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*(-wz*wz-wy*wy)-
2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wy*wy)*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy);
```

```
                    MapleGenVar9 = sk*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wy*wy*wx-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wy*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx);
                    MapleGenVar10 = u0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx
*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
                    MapleGenVar8 = MapleGenVar9+MapleGenVar10;
                    MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                    MapleGenVar7 = X;
                    MapleGenVar5 = MapleGenVar6*MapleGenVar7;
                    MapleGenVar8 = au*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*wy*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wy*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx);
                    MapleGenVar9 = sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*(-wz*wz-wx*wx)-
2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-
wx*wx)*wy)+u0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*w
y+wz*wz,3.0))*wy*wy*wz-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz);
                    MapleGenVar7 = MapleGenVar8+MapleGenVar9;
                    MapleGenVar8 = Y;
                    MapleGenVar6 = MapleGenVar7*MapleGenVar8;
                    MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                    MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                    MapleGenVar3 = MapleGenVar4*MapleGenVar5;
                    MapleGenVar1 = MapleGenVar2*MapleGenVar3;
                    MapleGenVar5 = (au*(1.0+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wy*wy))+sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+u0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X;
                    MapleGenVar6 = (au*(-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+sk*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+u0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y;
                    MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                    MapleGenVar3 = MapleGenVar4+au*tx+sk*ty+u0*tz;
                    MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                    MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx
*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx)*X;
                    MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*w
y+wz*wz,3.0))*wy*wy*wz-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)*Y;
                    MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                    MapleGenVar4 = MapleGenVar5*MapleGenVar6;
                    MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                    cvmSet(J,2*i   , 6 , MapleGenVar1+MapleGenVar2);

                    MapleGenVar2 = -1.0;
                    MapleGenVar7 = av*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wy*wy*wx-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wy*wy*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx);
                    MapleGenVar8 = v0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wy-
```

```
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx
*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
                    MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                    MapleGenVar7 = X;
                    MapleGenVar5 = MapleGenVar6*MapleGenVar7;
                    MapleGenVar6 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*(-wz*wz-wx*wx)-
2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-
wx*wx)*wy)+v0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*w
y+wz*wz,3.0))*wy*wy*wz-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz))*Y;
                    MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                    MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                    MapleGenVar3 = MapleGenVar4*MapleGenVar5;
                    MapleGenVar1 = MapleGenVar2*MapleGenVar3;
                    MapleGenVar3 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+v0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X+(av*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+v0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y+av*ty+v0*tz;
                    MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                    MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0
))*wy*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx
*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx)*X;
                    MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wy-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wy*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*w
y+wz*wz,3.0))*wy*wy*wz-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)*Y;
                    MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                    MapleGenVar4 = MapleGenVar5*MapleGenVar6;
                    MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                    cvmSet(J,2*i+1 , 6 , MapleGenVar1+MapleGenVar2);

                    MapleGenVar2 = -1.0;
                    MapleGenVar7 = au*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*(-wz*wz-wy*wy)-
2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wy*wy)*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz);
                    MapleGenVar9 = sk*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wz*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
                    MapleGenVar10 = u0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx);
                    MapleGenVar8 = MapleGenVar9+MapleGenVar10;
                    MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                    MapleGenVar7 = X;
                    MapleGenVar5 = MapleGenVar6*MapleGenVar7;
                    MapleGenVar8 = au*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wz*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx
*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
                    MapleGenVar9 = sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*(-wz*wz-wx*wx)-
2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wx*wx)*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)+u0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wz*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy);
                    MapleGenVar7 = MapleGenVar8+MapleGenVar9;
```

```
                        MapleGenVar8 = Y;
                        MapleGenVar6 = MapleGenVar7*MapleGenVar8;
                        MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                        MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                        MapleGenVar3 = MapleGenVar4*MapleGenVar5;
                        MapleGenVar1 = MapleGenVar2*MapleGenVar3;
                        MapleGenVar5 = (au*(1.0+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wy*wy))+sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+u0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X;
                        MapleGenVar6 = (au*(-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+sk*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+u0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y;
                        MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                        MapleGenVar3 = MapleGenVar4+au*tx+sk*ty+u0*tz;
                        MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                        MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx)*X;
                        MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wz*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy)*Y;
                        MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                        MapleGenVar4 = MapleGenVar5*MapleGenVar6;
                        MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                        cvmSet(J,2*i   , 7 , MapleGenVar1+MapleGenVar2);

                        MapleGenVar2 = -1.0;
                        MapleGenVar7 = av*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wz*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz
*wz)+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wx*wz*wy-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wy*wx);
                        MapleGenVar8 = v0*(-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx);
                        MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                        MapleGenVar7 = X;
                        MapleGenVar5 = MapleGenVar6*MapleGenVar7;
                        MapleGenVar6 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*(-wz*wz-wx*wx)-
2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*(-wz*wz-wx*wx)*wz-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz)+v0*(cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wz*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy))*Y;
                        MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                        MapleGenVar5 = 1/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                        MapleGenVar3 = MapleGenVar4*MapleGenVar5;
                        MapleGenVar1 = MapleGenVar2*MapleGenVar3;
                        MapleGenVar3 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+v0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X+(av*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+v0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y+av*ty+v0*tz;
```

```
                        MapleGenVar5 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                        MapleGenVar7 = (-
cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wy*wz+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0)
)*wz*wy+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wz*wx-2.0*(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wx)*X;
                        MapleGenVar8 = (cos(sqrt(wx*wx+wy*wy+wz*wz))/(wx*wx+wy*wy+wz*wz)*wx*wz-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy+wz*wz,3.0))*wz*wx+sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(pow(wx*wx+wy*wy
+wz*wz,3.0))*wz*wz*wy-2.0*(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/pow(wx*wx+wy*wy+wz*wz,2.0)*wz*wz*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy)*Y;
                        MapleGenVar6 = MapleGenVar7+MapleGenVar8;
                        MapleGenVar4 = MapleGenVar5*MapleGenVar6;
                        MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                        cvmSet(J,2*i+1 , 7 , MapleGenVar1+MapleGenVar2);

                        cvmSet(J,2*i   , 8 , -au/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));
                        cvmSet(J,2*i+1 , 8 , 0);
                        cvmSet(J,2*i   , 9 , -sk/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));
                        cvmSet(J,2*i+1 , 9 , -av/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz));

                        MapleGenVar1 = -u0/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                        MapleGenVar5 = (au*(1.0+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wy*wy))+sk*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+u0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X;
                        MapleGenVar6 = (au*(-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+sk*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+u0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y;
                        MapleGenVar4 = MapleGenVar5+MapleGenVar6;
                        MapleGenVar3 = MapleGenVar4+au*tx+sk*ty+u0*tz;
                        MapleGenVar4 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                        MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                        cvmSet(J,2*i   , 10, MapleGenVar1+MapleGenVar2);
                        MapleGenVar1 = -v0/((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz);
                        MapleGenVar3 = (av*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wz+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wy*wx)+v0*(-
sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx))*X+(av*(1.0+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*(-wz*wz-
wx*wx))+v0*(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wx+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy))*Y+av*ty+v0*tz;
                        MapleGenVar4 = 1/(pow((-sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*wy+(1.0-
cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wx)*X+(sin(sqrt(wx*wx+wy*wy+wz*wz))/sqrt(wx*wx+wy*wy+wz*wz)*w
x+(1.0-cos(sqrt(wx*wx+wy*wy+wz*wz)))/(wx*wx+wy*wy+wz*wz)*wz*wy)*Y+tz,2.0));
                        MapleGenVar2 = MapleGenVar3*MapleGenVar4;
                        cvmSet(J,2*i+1 , 10, MapleGenVar1+MapleGenVar2);
                }
}
```

< Function ErrorsGD>

Return errors w.r.t parameter vector P and display corners on the image

```
double ErrorsGD(CvMat* P,CvMat* ObjectPoints,CvMat* ImgPoints,int N,CvMat* d,IplImage *img,int Viewflag){
```

```
	CvMat *R   = cvCreateMat(3,3,CV_64FC1);
	CvMat *K   = cvCreateMat(3,3,CV_64FC1);
	CvMat *W   = cvCreateMat(3,1,CV_64FC1);
	CvMat *Rt  = cvCreateMat(3,4,CV_64FC1);
	// set K
	cvmSet(K,0,0,cvmGet(P,0,0));
	cvmSet(K,1,0,0);
	cvmSet(K,2,0,0);
	cvmSet(K,0,1,cvmGet(P,4,0));
	cvmSet(K,1,1,cvmGet(P,1,0));
	cvmSet(K,2,1,0);
	cvmSet(K,0,2,cvmGet(P,2,0));
	cvmSet(K,1,2,cvmGet(P,3,0));
	cvmSet(K,2,2,1);
	// set R
	cvmSet(W,0,0,cvmGet(P,5,0));
	cvmSet(W,1,0,cvmGet(P,6,0));
	cvmSet(W,2,0,cvmGet(P,7,0));
	Rodrigues2R(W, R);
	// set homography    Homography=K*[R(1),R(2),t]
	cvmSet(Rt,0,0,cvmGet(R,0,0));
	cvmSet(Rt,1,0,cvmGet(R,1,0));
	cvmSet(Rt,2,0,cvmGet(R,2,0));
	cvmSet(Rt,0,1,cvmGet(R,0,1));
	cvmSet(Rt,1,1,cvmGet(R,1,1));
	cvmSet(Rt,2,1,cvmGet(R,2,1));
	cvmSet(Rt,0,2,cvmGet(R,0,2));
	cvmSet(Rt,1,2,cvmGet(R,1,2));
	cvmSet(Rt,2,2,cvmGet(R,2,2));
	cvmSet(Rt,0,3,cvmGet(P,8,0));
	cvmSet(Rt,1,3,cvmGet(P,9,0));
	cvmSet(Rt,2,3,cvmGet(P,10,0));
	if(Viewflag==1)		CheckRtK(Rt,K,ObjectPoints,img);
	double RMSE=Residuals(Rt,K,ObjectPoints,ImgPoints,d);
	return RMSE;
}
```

< Function CheckRtK>
Check a Homography w.r.t Rt , K, and ObjectPoints

```
void CheckRtK(CvMat* Rt,CvMat* K,CvMat* ObjectPoints, IplImage *img){
	int i;
	CvMat* Homography=cvCreateMat(3,3,CV_64FC1);
	CvMat* temp=cvCreateMat(3,3,CV_64FC1);
	CvMat* estImgPoints=cvCreateMat(3,80,CV_64FC1);
	cvmSet(temp,0,0,cvmGet(Rt,0,0));
	cvmSet(temp,1,0,cvmGet(Rt,1,0));
	cvmSet(temp,2,0,cvmGet(Rt,2,0));
	cvmSet(temp,0,1,cvmGet(Rt,0,1));
	cvmSet(temp,1,1,cvmGet(Rt,1,1));
	cvmSet(temp,2,1,cvmGet(Rt,2,1));
	cvmSet(temp,0,2,cvmGet(Rt,0,3));
	cvmSet(temp,1,2,cvmGet(Rt,1,3));
	cvmSet(temp,2,2,cvmGet(Rt,2,3));
	cvMatMul(K,temp,Homography);
	cvMatMul(Homography,ObjectPoints,estImgPoints);
	CvPoint Draw;
	for(i=0;i<80;i++)
	{
		Draw.x=cvmGet(estImgPoints,0,i)/cvmGet(estImgPoints,2,i)+.5;
		Draw.y=cvmGet(estImgPoints,1,i)/cvmGet(estImgPoints,2,i)+.5;
		draw_cross(img,Draw,CV_RGB(255,255,255),3);
	}
	cvNamedWindow( "Check Points", 1 );
	cvShowImage( "Check Points", img );
	cvWaitKey(0);
}
```

< Function Residuals>
Return errors w.r.t the Rt, K , ObjectPoints and ImgPoints

```
double Residuals(CvMat* Rt,CvMat* K,CvMat* ObjectPoints,CvMat* ImgPoints){
	int i;
	CvMat* Homography=cvCreateMat(3,3,CV_64FC1);
	CvMat* temp=cvCreateMat(3,3,CV_64FC1);
	CvMat* estImgPoints=cvCreateMat(3,80,CV_64FC1);
	cvmSet(temp,0,0,cvmGet(Rt,0,0));
	cvmSet(temp,1,0,cvmGet(Rt,1,0));
	cvmSet(temp,2,0,cvmGet(Rt,2,0));
	cvmSet(temp,0,1,cvmGet(Rt,0,1));
	cvmSet(temp,1,1,cvmGet(Rt,1,1));
	cvmSet(temp,2,1,cvmGet(Rt,2,1));
	cvmSet(temp,0,2,cvmGet(Rt,0,3));
	cvmSet(temp,1,2,cvmGet(Rt,1,3));
	cvmSet(temp,2,2,cvmGet(Rt,2,3));
	cvMatMul(K,temp,Homography);
	cvMatMul(Homography,ObjectPoints,estImgPoints);
	double dtd=0;
	for(i=0;i<80;i++)
	{
		dtd=dtd+pow((cvmGet(estImgPoints,0,i)/cvmGet(estImgPoints,2,i))-cvmGet(ImgPoints,0,i),2);
		dtd=dtd+pow((cvmGet(estImgPoints,1,i)/cvmGet(estImgPoints,2,i))-cvmGet(ImgPoints,1,i),2);
	};
	return dtd;
}
```

< Function Residuals>
Return errors & d vector w.r.t the Rt, K , ObjectPoints and ImgPoints

```
double Residuals(CvMat* Rt,CvMat* K,CvMat* ObjectPoints,CvMat* ImgPoints,CvMat* d){
	int i;
	CvMat* Homography=cvCreateMat(3,3,CV_64FC1);
	CvMat* temp=cvCreateMat(3,3,CV_64FC1);
	CvMat* estImgPoints=cvCreateMat(3,80,CV_64FC1);
	cvmSet(temp,0,0,cvmGet(Rt,0,0));
	cvmSet(temp,1,0,cvmGet(Rt,1,0));
	cvmSet(temp,2,0,cvmGet(Rt,2,0));
	cvmSet(temp,0,1,cvmGet(Rt,0,1));
	cvmSet(temp,1,1,cvmGet(Rt,1,1));
	cvmSet(temp,2,1,cvmGet(Rt,2,1));
	cvmSet(temp,0,2,cvmGet(Rt,0,3));
	cvmSet(temp,1,2,cvmGet(Rt,1,3));
	cvmSet(temp,2,2,cvmGet(Rt,2,3));
	cvMatMul(K,temp,Homography);
	cvMatMul(Homography,ObjectPoints,estImgPoints);
	for(i=0;i<80;i++)
	{
		// d : input-estimated
		cvmSet(d,i*2,0,cvmGet(ImgPoints,0,i)-cvmGet(estImgPoints,0,i)/cvmGet(estImgPoints,2,i));
		cvmSet(d,i*2+1,0,cvmGet(ImgPoints,1,i)-cvmGet(estImgPoints,1,i)/cvmGet(estImgPoints,2,i));
	}
	//return d'*d
	double dtd=0;
	for(i=0;i<160;i++) dtd=dtd+pow(cvmGet(d,i,0),2);
	return dtd;
}
```

<Function Homography>

- Estimate H , datapoint1 is in the domain and datapoint2 is in the range.
- N is the number of points

```
void Homography(CvMat* datapoints1, CvMat* datapoints2, int N, CvMat* H){
	int j;
```

```
	CvMat* A  = cvCreateMat(N*2,9,CV_64FC1);
	CvMat* U  = cvCreateMat(N*2,N*2,CV_64FC1);
	CvMat* D  = cvCreateMat(N*2,9,CV_64FC1);
	CvMat* V  = cvCreateMat(9,9,CV_64FC1);
	for(j=0;j<N;j++)
	{
		cvmSet(A,2*j,0,0);
		cvmSet(A,2*j,1,0);
		cvmSet(A,2*j,2,0);
		cvmSet(A,2*j,3,-cvmGet(datapoints1,0,j));
		cvmSet(A,2*j,4,-cvmGet(datapoints1,1,j));
		cvmSet(A,2*j,5,-cvmGet(datapoints1,2,j));
		cvmSet(A,2*j,6,cvmGet(datapoints2,1,j)*cvmGet(datapoints1,0,j));
		cvmSet(A,2*j,7,cvmGet(datapoints2,1,j)*cvmGet(datapoints1,1,j));
		cvmSet(A,2*j,8,cvmGet(datapoints2,1,j)*cvmGet(datapoints1,2,j));
		cvmSet(A,2*j+1,0,cvmGet(datapoints1,0,j));
		cvmSet(A,2*j+1,1,cvmGet(datapoints1,1,j));
		cvmSet(A,2*j+1,2,cvmGet(datapoints1,2,j));
		cvmSet(A,2*j+1,3,0);
		cvmSet(A,2*j+1,4,0);
		cvmSet(A,2*j+1,5,0);
		cvmSet(A,2*j+1,6,-cvmGet(datapoints2,0,j)*cvmGet(datapoints1,0,j));
		cvmSet(A,2*j+1,7,-cvmGet(datapoints2,0,j)*cvmGet(datapoints1,1,j));
		cvmSet(A,2*j+1,8,-cvmGet(datapoints2,0,j)*cvmGet(datapoints1,2,j));
	}
	// estimate H
	cvSVD(A, D, U, V, CV_SVD_V_T);
	cvmSet(H,0,0,cvmGet(V,8,0));
	cvmSet(H,0,1,cvmGet(V,8,1));
	cvmSet(H,0,2,cvmGet(V,8,2));
	cvmSet(H,1,0,cvmGet(V,8,3));
	cvmSet(H,1,1,cvmGet(V,8,4));
	cvmSet(H,1,2,cvmGet(V,8,5));
	cvmSet(H,2,0,cvmGet(V,8,6));
	cvmSet(H,2,1,cvmGet(V,8,7));
	cvmSet(H,2,2,cvmGet(V,8,8));
}
```

<function NormalizationMatrixImg>
- return matrix for normalizing Corners1

```
double NormalizationMatrixImg(CvMat *Corners1,int N,CvMat *MT){
	int i;
	double scale,Cx,Cy,AvgDist;
	Cx=0;Cy=0;AvgDist=0;
	for(i=0;i<N;i++)
	{
		Cx=Cx+cvmGet(Corners1,0,i);
		Cy=Cy+cvmGet(Corners1,1,i);
	}
	Cx=Cx/N;
	Cy=Cy/N;
	for(i=0;i<N;i++) AvgDist=AvgDist+sqrt(pow(cvmGet(Corners1,0,i)-Cx,2)+pow(cvmGet(Corners1,1,i)-Cy,2));
	AvgDist=AvgDist/N;
	scale=sqrt(2)/AvgDist;

	cvmSet(MT,0,0,scale);
	cvmSet(MT,0,1,0);
	cvmSet(MT,0,2,-scale*Cx);
	cvmSet(MT,1,0,0);
	cvmSet(MT,1,1,scale);
	cvmSet(MT,1,2,-scale*Cy);
	cvmSet(MT,2,0,0);
	cvmSet(MT,2,1,0);
	cvmSet(MT,2,2,1);
	return scale;
}
```

```
================= Camera Calibration Report ====================
Number of images : 35
Image size : 640 X 480 pixel
Line detection method : the Hough transformation
Point detection method : intersecting lines acquired by the Hough transformation


================= Estimated absolute Conic ====================
|      -0.00000146        0.00000000        0.00047556 |
|       0.00000000       -0.00000147        0.00035138 |
|       0.00047556        0.00035138       -0.99999983 |

================= Estimated Instrinsic Parameters ====================
vo=240.060496
lamda=-0.760411
alpha=722.089891
beta=720.074380
gamma=1.032416
uo=326.433009
|           |   |      722.08989075        1.03241613      326.43300873  |
|     K     | = |        0.00000000      720.07437981      240.06049569  |
|           |   |        0.00000000        0.00000000        1.00000000  |


================= Estimated Extrinsic Parameters ====================
P1010001s.jpg
|           |   |       -0.99957794        0.00970608        0.02738144        8.00519800  |
|   R   | t | = |       -0.01102359       -0.99876798       -0.04838386        9.66484894  |
|           |   |        0.02687809       -0.04866528        0.99845343      -49.60822938  |
errors(dtd) = 157194.426646 pixels

P1010002s.jpg
|           |   |       -0.96896583        0.00306952        0.24717564        7.94090494  |
|   R   | t | = |       -0.01302612       -0.99916766       -0.03865630        9.67477209  |
|           |   |        0.24685125       -0.04067638        0.96819930      -48.68088530  |
errors(dtd) = 147479.834817 pixels

P1010003s.jpg
|           |   |       -0.97910141        0.17033140       -0.11111998        5.51075749  |
|   R   | t | = |       -0.16213021       -0.98359042       -0.07914337       10.13206785  |
|           |   |       -0.12277715       -0.05947348        0.99065063      -52.25856603  |
errors(dtd) = 249150.793581 pixels

P1010004s.jpg
|           |   |       -0.94149662        0.08354707       -0.32650269        6.49487840  |
|   R   | t | = |       -0.05829528       -0.99455420       -0.08639218        9.38767200  |
|           |   |       -0.33194244       -0.06230438        0.94123981      -50.05679085  |
errors(dtd) = 225959.588606 pixels

P1010005s.jpg
|           |   |        0.98594577        0.08512974       -0.14374931       -7.46600082  |
|   R   | t | = |       -0.09294743        0.99448604       -0.04856225       -9.40268853  |
|           |   |        0.13882259        0.06124088        0.98842190       47.63608488  |
errors(dtd) = 270008.263600 pixels

P1010006s.jpg
|           |   |        0.99258281       -0.00071961       -0.12156824       -6.23496600  |
|   R   | t | = |       -0.00667798        0.99814992       -0.06043293       -8.80354139  |
|           |   |        0.12138682        0.06079652        0.99074165       53.29208353  |
errors(dtd) = 371963.162882 pixels

P1010007s.jpg
|           |   |        0.93178500       -0.10211753        0.34835144       -7.05645857  |
|   R   | t | = |        0.11646821        0.99298395       -0.02044563      -11.81426550  |
|           |   |       -0.34381953        0.05962280        0.93714100       55.45426439  |
errors(dtd) = 132643.332869 pixels

P1010008s.jpg
|           |   |       -0.89941956        0.01431259        0.43685192        4.94828141  |
|   R   | t | = |       -0.10231919       -0.97858715       -0.17860002       10.69248954  |
|           |   |        0.42494145       -0.20533469        0.88162488      -49.58032013  |
```

errors(dtd) = 365301.672289 pixels

P1010009s.jpg

			0.83730779	-0.01018603	0.54663690	-5.38426615
R	t	=	0.04771830	0.99737251	-0.05450719	-10.65317386
			-0.54464541	0.07172387	0.83559384	53.98362338

errors(dtd) = 282850.711621 pixels

P1010010s.jpg

			0.79226369	-0.18178532	0.58247089	-4.83791038
R	t	=	0.20343697	0.97866658	0.02872503	-12.17979117
			-0.57526658	0.09573832	0.81234385	52.40227804

errors(dtd) = 147496.727505 pixels

P1010011s.jpg

			-0.82558324	0.04490821	-0.56249050	3.99101251
R	t	=	0.00653777	-0.99599993	-0.08911448	9.51933422
			-0.56424248	-0.07724886	0.82198725	-46.94177872

errors(dtd) = 372541.458834 pixels

P1010012s.jpg

			0.87299584	0.16349188	-0.45950917	-6.36867916
R	t	=	-0.20059984	0.97912609	-0.03273854	-8.25878032
			0.44456493	0.12075807	0.88756944	47.55969829

errors(dtd) = 473366.215305 pixels

P1010013s.jpg

			0.81758699	0.42725540	-0.38601080	-8.29572671
R	t	=	-0.41722589	0.90159558	0.11422768	-5.20255596
			0.39683002	0.06766263	0.91539483	51.72087108

errors(dtd) = 857248.199249 pixels

P1010014s.jpg

			-0.98743123	0.15752851	0.01281913	5.44942031
R	t	=	-0.15804283	-0.98486835	-0.07111113	11.55782683
			0.00142313	-0.07224333	0.99738602	-49.75817714

errors(dtd) = 170982.728783 pixels

P1010015s.jpg

			0.98831636	0.13660094	-0.06760884	-8.79088797
R	t	=	-0.13980847	0.98914582	-0.04521218	-8.32442278
			0.06069898	0.05413623	0.99668696	60.74275612

errors(dtd) = 379171.972868 pixels

P1010016s.jpg

			-0.99945475	-0.01200895	0.03075679	8.58678093
R	t	=	0.02634065	-0.85166952	0.52341686	11.47489779
			0.01990894	0.52394162	0.85152147	-54.08855462

errors(dtd) = 212294.229051 pixels

P1010017s.jpg

			-0.92265860	-0.06452707	0.38018070	9.26950581
R	t	=	0.25025059	-0.85028949	0.46301449	9.22409427
			0.29338669	0.52234475	0.80067485	-55.75209303

errors(dtd) = 263977.159854 pixels

P1010018s.jpg

			0.89052416	-0.08649920	0.44663699	-5.52844236
R	t	=	-0.17142672	0.84558890	0.50556136	-9.12691483
			-0.42140194	-0.52678012	0.73818908	56.95139217

errors(dtd) = 285627.982604 pixels

P1010019s.jpg

			0.74869684	0.18755929	-0.63582588	-7.30975999
R	t	=	0.13768668	0.89422652	0.42591232	-11.34377040
			0.64845618	-0.40642396	0.64368016	51.05076936

errors(dtd) = 351599.950780 pixels

P1010020s.jpg

			0.88023361	0.22378584	-0.41845991	-6.94032069
R	t	=	-0.01751128	0.89653844	0.44261967	-11.46995093

```
|           |   |        0.47421741       -0.38228094        0.79308205       50.89648634  |
errors(dtd) = 538854.453151 pixels

P1010048s.jpg
|           |   |       -0.94992655        0.13878873       -0.27995934        6.68035166  |
|   R   | t | = |       -0.11439521       -0.98821098       -0.10174868       11.90858651  |
|           |   |       -0.29078046       -0.06462777        0.95460462      -39.55094875  |
errors(dtd) = 13892.101855 pixels

P1010049s.jpg
|           |   |       -0.91479249       -0.17324228        0.36488603       11.05518106  |
|   R   | t | = |        0.15752225       -0.98483813       -0.07266771        9.20992297  |
|           |   |        0.37194280       -0.00899821        0.92821204      -45.45418191  |
errors(dtd) = 169300.819533 pixels

P1010050s.jpg
|           |   |       -0.86570688        0.01110640        0.50042807        9.52288084  |
|   R   | t | = |       -0.02640950       -0.99937479       -0.02350674       11.07675896  |
|           |   |        0.49985412       -0.03356600        0.86545894      -44.65927821  |
errors(dtd) = 76666.126766 pixels

P1010051s.jpg
|           |   |        0.87936855        0.18913257       -0.43696662       -9.53100624  |
|   R   | t | = |       -0.20190354        0.97924851        0.01753032      -10.50682976  |
|           |   |        0.43121447        0.07280950        0.89930688       42.06122574  |
errors(dtd) = 223728.547618 pixels

P1010052s.jpg
|           |   |        0.91008656        0.02118863       -0.41387619       -5.57482899  |
|   R   | t | = |       -0.04500463        0.99783884       -0.04787728      -10.69271650  |
|           |   |        0.41196728        0.06219881        0.90907330       38.47909907  |
errors(dtd) = 261886.597373 pixels

P1010053s.jpg
|           |   |       -0.86939246       -0.10634217        0.48254336       10.80628786  |
|   R   | t | = |        0.06911652       -0.99313791       -0.09433984        9.87995146  |
|           |   |        0.48926441       -0.04866663        0.87077661      -43.76637292  |
errors(dtd) = 147928.045496 pixels

P1010054s.jpg
|           |   |        0.80045422        0.00082186        0.59939333       -9.69600482  |
|   R   | t | = |        0.01729882        0.99955083       -0.02447208      -11.10512841  |
|           |   |       -0.59914422        0.02995757        0.80008046       46.36652701  |
errors(dtd) = 136899.767885 pixels

P1010055s.jpg
|           |   |        0.78790151       -0.18351208        0.58782184       -9.60407529  |
|   R   | t | = |        0.16534144        0.98255586        0.08512460      -13.16041316  |
|           |   |       -0.59318919        0.03012151        0.80449940       46.80903115  |
errors(dtd) = 388022.439837 pixels

P1010056s.jpg
|           |   |        0.97300257       -0.17163322       -0.15429854       -6.26421796  |
|   R   | t | = |        0.15735556        0.98241448       -0.10050386      -11.83187160  |
|           |   |        0.16883492        0.07351078        0.98289925       39.58534013  |
errors(dtd) = 34819.799700 pixels

P1010059s.jpg
|           |   |       -0.99985925        0.01637012        0.00367321        9.02599902  |
|   R   | t | = |       -0.01383363       -0.92830695        0.37155732       11.56240399  |
|           |   |        0.00949230        0.37145421        0.92840275      -44.65048994  |
errors(dtd) = 47581.930312 pixels

P1010060s.jpg
|           |   |        0.97814802        0.03605763        0.20475912       -9.57649858  |
|   R   | t | = |       -0.09904678        0.94672385        0.30643710      -10.03322213  |
|           |   |       -0.18280095       -0.32002157        0.92960745       43.47322156  |
errors(dtd) = 81441.286051 pixels

P1010061s.jpg
|           |   |       -0.99965684        0.00649643       -0.02537704        8.10597922  |
```

```
|   R   | t | = |       -0.00472130       -0.99757815       -0.06939410       10.25122815  |
|       |   |   |       -0.02576640       -0.06925047        0.99726650      -36.85364813  |
errors(dtd) = 105341.396343 pixels

P1010063s.jpg
|       |   |   |        0.98994396        0.10992323        0.08903839       -9.34479684  |
|   R   | t | = |       -0.10316929        0.99166237       -0.07721292       -9.56398482  |
|       |   |   |       -0.09678352        0.06725043        0.99303088       42.28540470  |
errors(dtd) = 141435.666304 pixels

P1010064s.jpg
|       |   |   |       -0.98125112       -0.15190501        0.11862171        9.84327672  |
|   R   | t | = |        0.14390375       -0.98686736       -0.07337931        8.66437784  |
|       |   |   |        0.12821058       -0.05493342        0.99022440      -43.40775298  |
errors(dtd) = 192175.291589 pixels

P1010066s.jpg
|       |   |   |        0.93527518       -0.12581450       -0.33080364       -7.29320986  |
|   R   | t | = |        0.08610111        0.98748478       -0.13213783      -11.71813191  |
|       |   |   |        0.34328842        0.09510267        0.93440277       37.16934517  |
errors(dtd) = 140.580377 pixels


Refined Instrinsic Parameters
|       |   |   |      721.00540848        0.51099301      326.74810283  |
|   K   | = |   |        0.00000000      720.47787231      239.98309735  |
|       |   |   |        0.00000000        0.00000000        1.00000000  |

Refined Extrinsic Parameters (P1010001s.jpg)
|       |   |   |       -0.99959071        0.00935609        0.02703459        8.03363789  |
|   R   | t | = |       -0.01066533       -0.99875665       -0.04869717        9.63931954  |
|       |   |   |        0.02654536       -0.04896557        0.99844765      -49.52702209  |
errors(dtd) = 157346.754218 pixels (100.10%)


Refined Extrinsic Parameters (P1010002s.jpg)
|       |   |   |       -0.96761143        0.00197640        0.25243654        7.95755596  |
|   R   | t | = |       -0.01283869       -0.99906058       -0.04138990        9.61489716  |
|       |   |   |        0.25211759       -0.04329030        0.96672782      -48.52720609  |
errors(dtd) = 150487.494616 pixels (102.04%)


Refined Extrinsic Parameters (P1010003s.jpg)
|       |   |   |       -0.97941728        0.16908003       -0.11024401        5.53127917  |
|   R   | t | = |       -0.16212652       -0.98433254       -0.06931413       10.08099672  |
|       |   |   |       -0.12023641       -0.05001397        0.99148465      -52.28441740  |
errors(dtd) = 253202.808476 pixels (101.63%)


Refined Extrinsic Parameters (P1010004s.jpg)
|       |   |   |       -0.93933980        0.08013535       -0.33349522        6.49553836  |
|   R   | t | = |       -0.06084436       -0.99583435       -0.06791102        9.32609416  |
|       |   |   |       -0.33754807       -0.04350023        0.94030263      -50.00111044  |
errors(dtd) = 233250.239927 pixels (103.23%)


Refined Extrinsic Parameters (P1010005s.jpg)
|       |   |   |        0.98704098        0.08613354       -0.13539247       -7.49905903  |
|   R   | t | = |       -0.09313558        0.99457860       -0.04625120       -9.40058536  |
|       |   |   |        0.13067468        0.05826169        0.98971193       47.78104092  |
errors(dtd) = 271281.490526 pixels (100.47%)


Refined Extrinsic Parameters (P1010006s.jpg)
|       |   |   |        0.99259779        0.00019995       -0.12144792       -6.25397722  |
|   R   | t | = |       -0.00661259        0.99860424       -0.05240080       -8.76143916  |
|       |   |   |        0.12126793        0.05281601        0.99121368       53.26338190  |
errors(dtd) = 375492.805617 pixels (100.95%)


Refined Extrinsic Parameters (P1010007s.jpg)
```

```
|           |   |        0.93338887       -0.10203149        0.34405637       -7.09630478  |
|   R   | t | = |        0.11605518        0.99303414       -0.02035672      -11.80798015  |
|           |   |       -0.33958270        0.05893026        0.93872830       55.43966289  |
errors(dtd) = 132321.100590 pixels (99.76%)


Refined Extrinsic Parameters (P1010008s.jpg)
|           |   |       -0.88948102        0.01210113        0.45681186        4.93283943  |
|   R   | t | = |       -0.10260728       -0.97941262       -0.17384669       10.53519694  |
|           |   |        0.44530355       -0.20150556        0.87241060      -49.22233511  |
errors(dtd) = 371703.191986 pixels (101.75%)


Refined Extrinsic Parameters (P1010009s.jpg)
|           |   |        0.82866096       -0.00621730        0.55971632       -5.37004644  |
|   R   | t | = |        0.04982241        0.99678870       -0.06268989      -10.49528571  |
|           |   |       -0.55752914        0.07983507        0.82630964       53.42111967  |
errors(dtd) = 288717.793723 pixels (102.07%)


Refined Extrinsic Parameters (P1010010s.jpg)
|           |   |        0.78947373       -0.18265586        0.58597616       -4.85204262  |
|   R   | t | = |        0.20110108        0.97897266        0.03421823      -12.09928874  |
|           |   |       -0.57990480        0.09082605        0.80960549       52.21879992  |
errors(dtd) = 146161.433410 pixels (99.09%)


Refined Extrinsic Parameters (P1010011s.jpg)
|           |   |       -0.82731087        0.04101816       -0.56024480        4.02002178  |
|   R   | t | = |        0.00212197       -0.99709522       -0.07613555        9.52510952  |
|           |   |       -0.56174035       -0.06417659        0.82482068      -47.19852516  |
errors(dtd) = 380003.638912 pixels (102.00%)


Refined Extrinsic Parameters (P1010012s.jpg)
|           |   |        0.87807145        0.16638862       -0.44867065       -6.43099967  |
|   R   | t | = |       -0.19940900        0.97954454       -0.02699147       -8.29533520  |
|           |   |        0.43500182        0.11316940        0.89328949       47.99901500  |
errors(dtd) = 476941.701305 pixels (100.76%)


Refined Extrinsic Parameters (P1010013s.jpg)
|           |   |        0.82476068        0.43219958       -0.36465510       -8.37572061  |
|   R   | t | = |       -0.41626938        0.90049727        0.12579537       -5.23414745  |
|           |   |        0.38273963        0.04804368        0.92260619       52.46390465  |
errors(dtd) = 863526.279788 pixels (100.73%)


Refined Extrinsic Parameters (P1010014s.jpg)
|           |   |       -0.98749845        0.15679112        0.01622806        5.47260588  |
|   R   | t | = |       -0.15755043       -0.98500747       -0.07027192       11.51635290  |
|           |   |        0.00496674       -0.07195015        0.99739586      -49.72031375  |
errors(dtd) = 171979.556680 pixels (100.58%)


Refined Extrinsic Parameters (P1010015s.jpg)
|           |   |        0.98852493        0.13746692       -0.06262032       -8.81933171  |
|   R   | t | = |       -0.14016481        0.98928197       -0.04092699       -8.29178633  |
|           |   |        0.05632305        0.04923451        0.99719791       60.74982899  |
errors(dtd) = 380036.457285 pixels (100.23%)


Refined Extrinsic Parameters (P1010016s.jpg)
|           |   |       -0.99976588       -0.01959047        0.00918645        8.59752172  |
|   R   | t | = |        0.02150049       -0.85177165        0.52347186       11.43983451  |
|           |   |       -0.00243030        0.52354682        0.85199344      -53.80209420  |
errors(dtd) = 211431.473535 pixels (99.59%)


Refined Extrinsic Parameters (P1010017s.jpg)
|           |   |       -0.92227438       -0.06474698        0.38107452        9.29773681  |
```

```
|   R   | t | = |        0.25008163       -0.85168194        0.46053996        9.19760596  |
|           |   |        0.29473572        0.52004395        0.80167646      -55.64057982  |
errors(dtd) = 260033.477352 pixels (98.51%)


Refined Extrinsic Parameters (P1010018s.jpg)
|           |   |        0.89102800       -0.08509352        0.44590155       -5.56373336  |
|   R   | t | = |       -0.17043846        0.84769928        0.50235114       -9.09911947  |
|           |   |       -0.42073725       -0.52360771        0.74082059       56.82154897  |
errors(dtd) = 282237.887823 pixels (98.81%)


Refined Extrinsic Parameters (P1010019s.jpg)
|           |   |        0.74698407        0.18855303       -0.63754416       -7.32698379  |
|   R   | t | = |        0.13658633        0.89496346        0.42471706      -11.29073913  |
|           |   |        0.65066042       -0.40433669        0.64276968       50.80788462  |
errors(dtd) = 349684.246059 pixels (99.46%)


Refined Extrinsic Parameters (P1010020s.jpg)
|           |   |        0.88274946        0.22318425       -0.41345155       -7.00207407  |
|   R   | t | = |       -0.01915496        0.89633984        0.44295370      -11.49386186  |
|           |   |        0.46945339       -0.38309749        0.79551859       51.08945442  |
errors(dtd) = 536150.178926 pixels (99.50%)


Refined Extrinsic Parameters (P1010048s.jpg)
|           |   |       -0.94897994        0.13783116       -0.28361885        6.69564593  |
|   R   | t | = |       -0.11426765       -0.98859513       -0.09809470       11.85606313  |
|           |   |       -0.29390472       -0.06068144        0.95390659      -39.44539095  |
errors(dtd) = 13781.323421 pixels (99.20%)


Refined Extrinsic Parameters (P1010049s.jpg)
|           |   |       -0.91840015       -0.17409695        0.35529061       11.11990922  |
|   R   | t | = |        0.15813553       -0.98466104       -0.07372778        9.23774270  |
|           |   |        0.36267660       -0.01152753        0.93184376      -45.49764423  |
errors(dtd) = 168164.866189 pixels (99.33%)


Refined Extrinsic Parameters (P1010050s.jpg)
|           |   |       -0.86492262        0.00681195        0.50185900        9.52534402  |
|   R   | t | = |       -0.03045772       -0.99877744       -0.03893514       10.99967609  |
|           |   |        0.50098023       -0.04896136        0.86407268      -44.37458145  |
errors(dtd) = 76161.475767 pixels (99.34%)


Refined Extrinsic Parameters (P1010051s.jpg)
|           |   |        0.87913193        0.18665775       -0.43850420       -9.57105761  |
|   R   | t | = |       -0.20474748        0.97879547        0.00615662      -10.48325000  |
|           |   |        0.43035510        0.08437015        0.89870805       41.90092642  |
errors(dtd) = 222173.398270 pixels (99.30%)


Refined Extrinsic Parameters (P1010052s.jpg)
|           |   |        0.91792673        0.01849018       -0.39631886       -5.67938421  |
|   R   | t | = |       -0.04874169        0.99660214       -0.06639591      -10.80694282  |
|           |   |        0.39374456        0.08026383        0.91570898       38.85065323  |
errors(dtd) = 253495.255917 pixels (96.80%)


Refined Extrinsic Parameters (P1010053s.jpg)
|           |   |       -0.87631285       -0.10716716        0.46967115       10.90942875  |
|   R   | t | = |        0.06961755       -0.99287999       -0.09665779        9.95730081  |
|           |   |        0.47668563       -0.05200510        0.87753420      -43.94604834  |
errors(dtd) = 144413.240525 pixels (97.62%)


Refined Extrinsic Parameters (P1010054s.jpg)
|           |   |        0.80149311        0.00183597        0.59800119       -9.73117878  |
|   R   | t | = |        0.01775701        0.99948126       -0.02686808      -11.09075828  |
```

```
|           |   |       -0.59774032        0.03215330        0.80104474       46.32867420  |
errors(dtd) = 136571.441368 pixels (99.76%)


Refined Extrinsic Parameters (P1010055s.jpg)
|           |   |        0.79370760       -0.18105739        0.58072925       -9.69584636  |
|   R   | t | = |        0.16969664        0.98268056        0.07444443      -13.24204900  |
|           |   |       -0.58415006        0.03946069        0.81068586       46.97383913  |
errors(dtd) = 381720.525481 pixels (98.38%)


Refined Extrinsic Parameters (P1010056s.jpg)
|           |   |        0.97369364       -0.17054570       -0.15111207       -6.28864617  |
|   R   | t | = |        0.15728616        0.98289338       -0.09582096      -11.80496082  |
|           |   |        0.16486891        0.06953242        0.98386152       39.63833111  |
errors(dtd) = 35968.971011 pixels (103.30%)


Refined Extrinsic Parameters (P1010059s.jpg)
|           |   |       -0.99989051        0.01476323       -0.00101149        9.05064861  |
|   R   | t | = |       -0.01404335       -0.92514329        0.37935824       11.57816294  |
|           |   |        0.00466478        0.37933091        0.92524932      -44.74198993  |
errors(dtd) = 49843.833530 pixels (104.75%)


Refined Extrinsic Parameters (P1010060s.jpg)
|           |   |        0.97994895        0.03721173        0.19574304       -9.61551769  |
|   R   | t | = |       -0.09878776        0.94389564        0.31512221      -10.08326423  |
|           |   |       -0.17303476       -0.32814070        0.92864560       43.62252604  |
errors(dtd) = 80308.200822 pixels (98.61%)


Refined Extrinsic Parameters (P1010061s.jpg)
|           |   |       -0.99969786        0.00597065       -0.02384428        8.12611524  |
|   R   | t | = |       -0.00430017       -0.99757236       -0.06950463       10.22950601  |
|           |   |       -0.02420138       -0.06938109        0.99729662      -36.83229338  |
errors(dtd) = 106484.027690 pixels (101.08%)


Refined Extrinsic Parameters (P1010063s.jpg)
|           |   |        0.99011071        0.11041419        0.08654186       -9.36854771  |
|   R   | t | = |       -0.10361177        0.99143436       -0.07951417       -9.55894438  |
|           |   |       -0.09458006        0.06976108        0.99306999       42.24655689  |
errors(dtd) = 140980.751833 pixels (99.68%)


Refined Extrinsic Parameters (P1010064s.jpg)
|           |   |       -0.98168190       -0.15227872        0.11450697        9.87273029  |
|   R   | t | = |        0.14430884       -0.98668759       -0.07498368        8.66470310  |
|           |   |        0.12440102       -0.05708576        0.99058851      -43.36537080  |
errors(dtd) = 190935.516586 pixels (99.35%)


Refined Extrinsic Parameters (P1010066s.jpg)
|           |   |        0.93382148       -0.12479166       -0.33526777       -7.30346127  |
|   R   | t | = |        0.08599478        0.98800904       -0.12823037      -11.66084018  |
|           |   |        0.34724967        0.09091300        0.93335550       37.04521343  |
errors(dtd) = 27.225995 pixels (19.37%)

Improvement = sqrt( sum of refined errors ) / sqrt(sum of initial errors) = 0.427255.
```