```
#include <stdio.h>
#include <math.h>

#define ROWS    128
#define COLS    128

/*
**      This routine converts the data in an Odetics range image into 3D
**      cartesian coordinate data.  The range image is 8-bit, and comes
**      already separated from the intensity image.
*/


main(argc,argv)

int     argc;
char    *argv[];

{
int     r,c;
double  cp[7];
double  xangle,yangle,dist;
double  ScanDirectionFlag,SlantCorrection;
unsigned char   RangeImage[128*128];
double          P[3][128*128];
int             ImageTypeFlag;
char    Filename[160],Outfile[160];
FILE    *fpt;

printf("Enter range image file name:");
scanf("%s",Filename);
if ((fpt=fopen(Filename,"r")) == NULL)
  {
  printf("Couldn't open %s\n",Filename);
  exit();
  }
fread(RangeImage,1,128*128,fpt);
fclose(fpt);

printf("Up(-1), Down(1) or Neither(0)? ");
scanf("%d",&ImageTypeFlag);


cp[0]=1220.7;           /* horizontal mirror angular velocity in rpm */
cp[1]=32.0;             /* scan time per single pixel in microseconds */
cp[2]=(COLS/2)-0.5;             /* middle value of columns */
cp[3]=1220.7/192.0;     /* vertical mirror angular velocity in rpm */
cp[4]=6.14;             /* scan time (with retrace) per line in milliseconds */
cp[5]=(ROWS/2)-0.5;             /* middle value of rows */
cp[6]=10.0;             /* standoff distance in range units (3.66cm per r.u.) */

cp[0]=cp[0]*3.1415927/30.0;     /* convert rpm to rad/sec */
cp[3]=cp[3]*3.1415927/30.0;     /* convert rpm to rad/sec */
cp[0]=2.0*cp[0];                /* beam ang. vel. is twice mirror ang. vel. */
cp[3]=2.0*cp[3];                /* beam ang. vel. is twice mirror ang. vel. */
cp[1]/=1000000.0;               /* units are microseconds : 10^-6 */
cp[4]/=1000.0;                  /* units are milliseconds : 10^-3 */

switch(ImageTypeFlag)
  {
  case 1:               /* Odetics image -- scan direction upward */
    ScanDirectionFlag=-1;
    break;
  case 0:               /* Odetics image -- scan direction downward */
    ScanDirectionFlag=1;
    break;
  default:              /* in case we want to do this on synthetic model */
    ScanDirectionFlag=0;
    break;
  }

        /* start with semi-spherical coordinates from laser-range-finder: */
        /*                      (r,c,RangeImage[r*COLS+c])                */
        /* convert those to axis-independant spherical coordinates:       */
        /*                      (xangle,yangle,dist)                      */
        /* then convert the spherical coordinates to cartesian:           */
        /*                      (P => X[] Y[] Z[])                        */

if (ImageTypeFlag != 3)
  {
  for (r=0; r<ROWS; r++)
    {
    for (c=0; c<COLS; c++)
      {
      SlantCorrection=cp[3]*cp[1]*((double)c-cp[2]);
      xangle=cp[0]*cp[1]*((double)c-cp[2]);
      yangle=(cp[3]*cp[4]*(cp[5]-(double)r))+   /* Standard Transform Part */
        SlantCorrection*ScanDirectionFlag;      /*  + slant correction */
      dist=(double)RangeImage[r*COLS+c]+cp[6];
      P[2][r*COLS+c]=sqrt((dist*dist)/(1.0+(tan(xangle)*tan(xangle))
        +(tan(yangle)*tan(yangle))));
      P[0][r*COLS+c]=tan(xangle)*P[2][r*COLS+c];
      P[1][r*COLS+c]=tan(yangle)*P[2][r*COLS+c];
      }
    }
  }

sprintf(Outfile,"%s.coords",Filename);
fpt=fopen(Outfile,"w");
fwrite(P[0],8,128*128,fpt);
fwrite(P[1],8,128*128,fpt);
fwrite(P[2],8,128*128,fpt);
fclose(fpt);
}
```