Bibliografía

1. Berry R. *et al.*, CCD Camera cookbook, Willmann-Bell, 1993.

2. Buil C., CCD Astronomy, Willmann-Bell, 1991.

3. Cruz-Gonzalez I., *et al.* Resumen Ejecutivo del proyecto de modificación del diseño del telescopio óptico/infrarrojo mexicano de nueva tecnología: TIM. IA UNAM, 1996.

4. Fisher M., A high resolution incremental tape encoder on the William Herschel Telescope, *SPIE* 2199, 1994.

5. Francon M., Laser speckle and applications in optics, Academic Press, 1979

6. Goodman J., Some fundamental properties of speckle, Journal of the Optical Society of América, vol. 66(11), 1976.

7. Hayashi A. *et al.*, High-resolution rotation-angle measurement of a cylinder using spekle displacement rotation, Applied Optics vol. 22 No. 22, 1983

8. Holst G., CCD Arrays, cameras and displays, *SPIE*, 1996.

9. Jähne B., Digital image processing, Springer-Verlag 1991.

10. Jähne B., Practical handbook on image processing for scientific applications, CRC Press, 1997.

11. Janesick J., CCD Camera design and application, *SPIE Symposium*, 1994

12. Lewis H. *et al.*, Pointing and tracking performance of the W.M. Keck Tel., *SPIE* 2199 1994.

13. Mancini D., The Galileo Italian National Telescope (TNG) drive system, *SPIE* 2199, 1994.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

14. Michel-Murillo R., Development of the big MIC image photon counting unit, PhD Thesis, UCL, 1997.

15. Nachtigal C., Instrumentation and control, Willey 1990.

16. Schier A. *et al.*, A new telescope encoder, *SPIE* 2199 (1994).

17. Sihori r. *et al.*, Optical components systems and measurement technics, Marcel Decker, 1991.

18. Ravensbergen M. *et al.*, Encoders for the altitude and azimuth axes of the VLT, *SPIE* 2479. 1995.

19. Robinson D. *et al.*, Interferogram Analysis, IOP Publishing, 1993.

20. Russ J., The image processing handbook, CRC Press, 1992

21. Saddlemeyer L., High Resolution encoding using redundant egde detection, *SPIE* 2479 (1995).

22. Taylor J., An introduction to error analysis, University science books, 1982.

23. Wells L., Rectifying and Registering images with IRAF, NOAO 1994.

24. Wilkes, J. *et al.*, Selection of a tape encoding system for the main axis of the Gemini Telescopes, *SPIE* 3112, 1997.

25. Yamaguchi I., Speckle displacement and decorrelation in the diffraction and image fields for small object deformation, Optica Acta vol. 28(10), 1981.

26. Yamaguchi I., Automatic measurement of in-plane translation by speckle correlation using a linear image sensor, Journal of Physics E: Scientific instruments. vol 19, 1986.

27. Yamaguchi I. *et al.*, Laser rotary encoder, Applied Optics, vol. 28 No. 20, 1989.

28. Yamaguchi I. *et al.*, Linear & rotary encoders using electronic speckle correlation, Optical Eng. Vol. 30 No. 12, 1991.

29. Catálogo: *Optical encoders, Computer Optical Products*, 1997.

30. Catálogo: *Heindenhain angle encoders*, 1997.

31. Catálogo: *Acrotech Positioning Systems*, 1997.

32. Catálogo: *Apogee Instruments Inc.*, 1997.

33. Catálogo: *Thomson-CFS Semiconducteurs specifiques*, 1996.

34. Catálogo: *Optical encoders, Dynamics Research Co.*, 1995.

35. Catálogo: *Product guide, Renco Encoders.*, 1996.

36. Catálogo: *Optical encoder design guide, BEI Motion Systems.*, 1996.

37. Hoja de datos RL1024G, EG&G Reticon, 1992.

38. Manual: Star I CCD Camera system, Photometrics Inc., 1995.

Apéndice A

Programa desarrollado

```
/*********** PROG. PRINCIPAL DE CODIFICADOR (CODISTAR.PRJ)*********

/*************************************************agosto'97 (para 2m)

Caracteristicas:

* correlacion binaria, (0--1)  con busqueda de maximo
* el umbral de binarizacion es la media
* usa 1 sola imagen (adquiere 1 y compara con la anterior)
* 1 de cada n pix   (ajustar umbral de binarizacion segun n)
* pecision de sub-pixel por ajuste de parabola
* obtiene valores reales: micras, segarc, vel (para 2metros)
* ultima mod 20/sep/97 durante pruebas en SPM */


/***************************** DEFINICIONES ************************

#define MAIN
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <ctype.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <time.h>
```

```
#include "codista.h"
#include "decl.h"
#include "gpguia.h"


#define RESULTADO       // despliega valores reales
#define ARCHIVO         // genera archivo de datos
#define VELOCIDAD       // calcula velocidad promedio en NVECES
#define NVECES 30       // numero de lecturas consecutivas
#define FACTSTAR 1.7908 // factor de pixeles star a micras:
                        // 1 micra = factstar* pixeles
                        // sigmaX = 23 / FACTSTAR

#define RADIO 0.892      // radio del punto de lectura
#define iteraciones 30   // limites de los corrimientos

int nit= 5;              // pixeles por corrimiento
                         // (1 iteracion cada nit)


int bufferI1[ANCHO * LARGO];    //buffers que contienen la imagen
int bufferI2[ANCHO * LARGO];
int bufferR2[ANCHO * LARGO];

int LArgo = ANCHO;      //  ANCHO = 2   (definido en gpguia.h)
int ANcho = LARGO;      // LARGO = 350
double sumaANG=0;



/***************************** PRINCIPAL ************************

void main(void)
{


#ifdef VELOCIDAD
       clock_t start, end;
#endif                          // umbral de seleccion de pico
       double umbral=0.7;
       if (nit==1 || nit==2) umbral=0.85;
       if(nit==3 || nit==4) umbral=0.8;
       if(nit<1 || nit>4 ) umbral=0.7;
       int media;
       int itr =nit;
```

```
        double valoresB[100];  // valores de la corr burda
        double valoresF[100];  // valores de la corr fina

        for (int h=1; h<=NVECES; h++){

          if (inicializa())

#ifdef VELOCIDAD                // inicializa reloj

          if (h==2) start = clock();
          if (h==NVECES) end = clock();
#endif
          toma_imagen();        // rutina de adquire imagen del
                                // programa de control de StarI
          cierra_gpib();
          media=Media(bufferI1);
          binaria(bufferI1, bufferI2, media);
          if (h==1) cambia_nom(bufferI2, bufferR2);
                                // solo para la primera
                                // bufferR2 = bufferI1


/********************** BUSQUEDA BURDA *********************

          int indice = 0;
          int kmax=0, pico=0;
          double X,XAnt=-1,max=0.0,umb=umbral;
          for(int k=-iteraciones;k<=iteraciones; k+=itr) {
             double X=pasa_y_corr(bufferI2, bufferR2,
                         k, ANcho-abs(k));
             valoresB[indice++] = X;


#ifdef ARCHIVO
             bitacora("Lagx =", k, "Xburdo =", X);
#endif

             if(X<XAnt && XAnt>=umb) {
                 pico=1;
                 break;         // pendiente positva
             }
             else {             // pendiente negativa

                 if(X > max && X>=umb) max=X;
                                // encontre un maximo
```

```
            }
            XAnt=X;
        }


/********************** BUSQUEDA FINA *********************

        XAnt=-1,max=0.0,pico=0;        // fino
        int INDICE=0;
        int ventanaI = (k-itr)-itr;
        int ventanaD = k + itr;
        for(int l=ventanaI;l<=ventanaD; l++) {
            double X=pasa_y_corr(bufferI2, bufferR2,
                        l, ANcho-abs(l));
            valoresF[INDICE++] = X;


#ifdef ARCHIVO
        bitacora("Lagx =", l, "Xfino =", X);
#endif

            if(X<XAnt && XAnt>=umb) {
                pico=1;
                break;          // pendiente negativa
            }
            else {              // pendiente positva

                if(X > max && X>=umb) {
                                // encontre un maximo
                    max=X;
                    kmax=INDICE-1;
                }
            }
            XAnt=X;
        }
        int kmaximo = kmax+ventanaI;


        if (pico==1) parabola(valoresF, kmax, kmaximo);

        else printf(" DESPLAZAMIENTO MAYOR QUE LOS LIMITES!! \n");

        cambia_nom(bufferI2, bufferR2);   // img = referencia
      }
```

```
/********************** CALCULA VELOCIDAD ************************

#ifdef VELOCIDAD
    printf("Tiempo : %f\n", (end - start) / CLK_TCK);
                        //clock_t define CLK_TCK
    double SUMAANG;
    if (sumaANG<0) SUMAANG= -sumaANG;
    else SUMAANG= sumaANG;
    printf("despl = %4.4f segarc\n",sumaANG);
    printf("Velocidad media:     W= %4.4f segarc/s\n",
                    SUMAANG/((end - start) / CLK_TCK));
    int minANG = (int)(SUMAANG/60);
    int minAR = (int) ((SUMAANG/15)/60);
    double segANG = SUMAANG -(double)(minANG*60);
    double segAR = (SUMAANG/15) -(double)(minAR*60);
    printf("= 00 %2d' %2.2f''           AR= 00h %2dm %2.2fs \n",
                        minANG,segANG,minAR,segAR);
#endif
#ifdef ARCHIVO
    bitacora("tiempo=",((end - start) / CLK_TCK) , "W =",
                    SUMAANG/((end - start) / CLK_TCK));
    bitacora("(min)=",minANG , "AR(seg)=", segAR);
#endif

    printf("Oprime una tecla para continuar...\n");
    getch();
}  // main



/****************************** BINARIZA ************************

/****** MEDIA ********* calcula la media de los valores de la imagen

int Media(int huge *BUFFER)
{

    int huge *p= BUFFER;
    long SUMA=0, j;
    long limite=LArgo*ANcho;
    for(j=0;j<limite;j++) {
         SUMA+=p[j];
    }
```

```
    return (int)(SUMA/limite);
} // Media


/******* BINARIA ********** hace la binarizacion

void binaria(int huge *buffer_dec, int huge *buffer_bin, int media)
{
    long j;
    int huge *ptr_dec = buffer_dec;
    int huge *ptr_bin = buffer_bin;
    long limite=LArgo*ANcho;
    for(j=0;j<limite;j++) {
        if (*ptr_dec >= media) *ptr_bin = 1;  //binariza (0 o 1)
        else *ptr_bin = 0;
        ptr_bin++;
        ptr_dec++;
    }

}// binaria



/****************** CORRIMIENTO Y CORRELACION **********************

double pasa_y_corr(int huge *II, int huge *IR, int lag, int ANcho1)
{
    int columnas, renglones;
    long npixs = LArgo * ANcho;
    long limite=LArgo*ANcho1;
    long x=0;
    int I=0, R=0,t=0;
    for(int i=0, r=0;i<npixs, r<npixs;i++, r++) {
        renglones = (int)(i/ANcho);
        columnas =(int)(i-(long)renglones*ANcho);
        if(renglones >= 0 && renglones < LArgo &&
                columnas >= 0 && columnas < ANcho1) {
            I=i;
            t++;
            R=r;
            if (lag>0) I=i+lag;
            else R=r-lag;
            if((II[I] ^ IR[R])==0) x++;

        }
```

```
    }
    return (double)x/limite;
} // pasa_y_corr


/************************ AJUSTE PARABOLICO ************************

double parabola ( double valcorr[100], int pos_max, int Kmaximo)
{       int pos_ant, pos_des;
    double Sm;
    double valA;                //valor antes del maximo
    double valM;                //valor del maximo
    double valD;                //valor despues del maximo
    valM = valcorr[pos_max];
    pos_ant = pos_max - 1;
    valA = valcorr[pos_ant];
    pos_des = pos_max + 1;
    valD = valcorr[pos_des];
    Sm = Kmaximo+ ((valA-valD)/(2*(valD-2*valM+valA)));


/************************* VALORES REALES ************************

    double lineal= Sm*FACTSTAR;
    double angular= (1.296*lineal)/(RADIO*6.2831853);
                //1.296 es  (360 *3600 / 1 millon)
    sumaANG+=angular;

#ifdef RESULTADO
    printf("Parabola: Sm = %+06.5f pixeles\n",Sm);
    if (lineal>=0){
      printf("Despl: Lineal= %+06.5f micras a la DERECHA  ",lineal);
      printf("Angular= %+06.5f segarc OESTE\n\n",angular);

    }
    else {
      printf("Despl: Lineal= %+06.5f micras a la IZQUIERDA ",-lineal);
      printf("Angular= %+06.5f segarc ESTE\n\n",-angular);

    }
#endif

#ifdef ARCHIVO
```

```
    bitacora("Sm=", Sm, "angular=",angular );
#endif

    return Sm;
} // parabola



/********************** IMAGEN = REFERENCIA ***************************

void cambia_nom(int huge *buffer_img, int huge *buffer_ref)
{
    long j;
    int huge *ptr_img = buffer_img;
    int huge *ptr_ref = buffer_ref;
    long limite=LArgo*ANcho;
    for(j=0;j<limite;j++) {
        *ptr_ref = *ptr_img;
        ptr_ref++;
        ptr_img++;
    }

}// cambia_nom



/************************ GERNERA ARCHIVO ***************************

#ifdef ARCHIVO
void bitacora(char *A, double Ai, char *B, double Bi)
{
    FILE *fp;
    char *archivo= "d:\\bitacora.txt";
    if ((fp = fopen (archivo, "a+")) != NULL)    {
        fprintf(fp,"%s %+4.4f, %s %+4.4f\n",A, Ai, B, Bi);
        fclose (fp);
    }
    else {
       fprintf (stderr, "Error al abrir la bitacora %s\n", archivo);
    }
} // archivo
#endif
```