1. Introducció.

El paquet Matrox Imaging Library (MIL) són unes llibreries de processament d'imatges dividit en diferents mòduls basats en la seva funcionalitat (Figura 1). Està format per un extens conjunt de funcions orientades al processament d'imatges i operacions especialitzades, com l'anàlisi d'objectes, el càlcul de les seves característiques i el reconeixement de patrons. També inclou un conjunt bàsic de gràfics. En general, les MIL són capaces de manipular imatges en color i en nivells de gris, malgrat que certes operacions només són possibles amb imatges en nivells de gris.

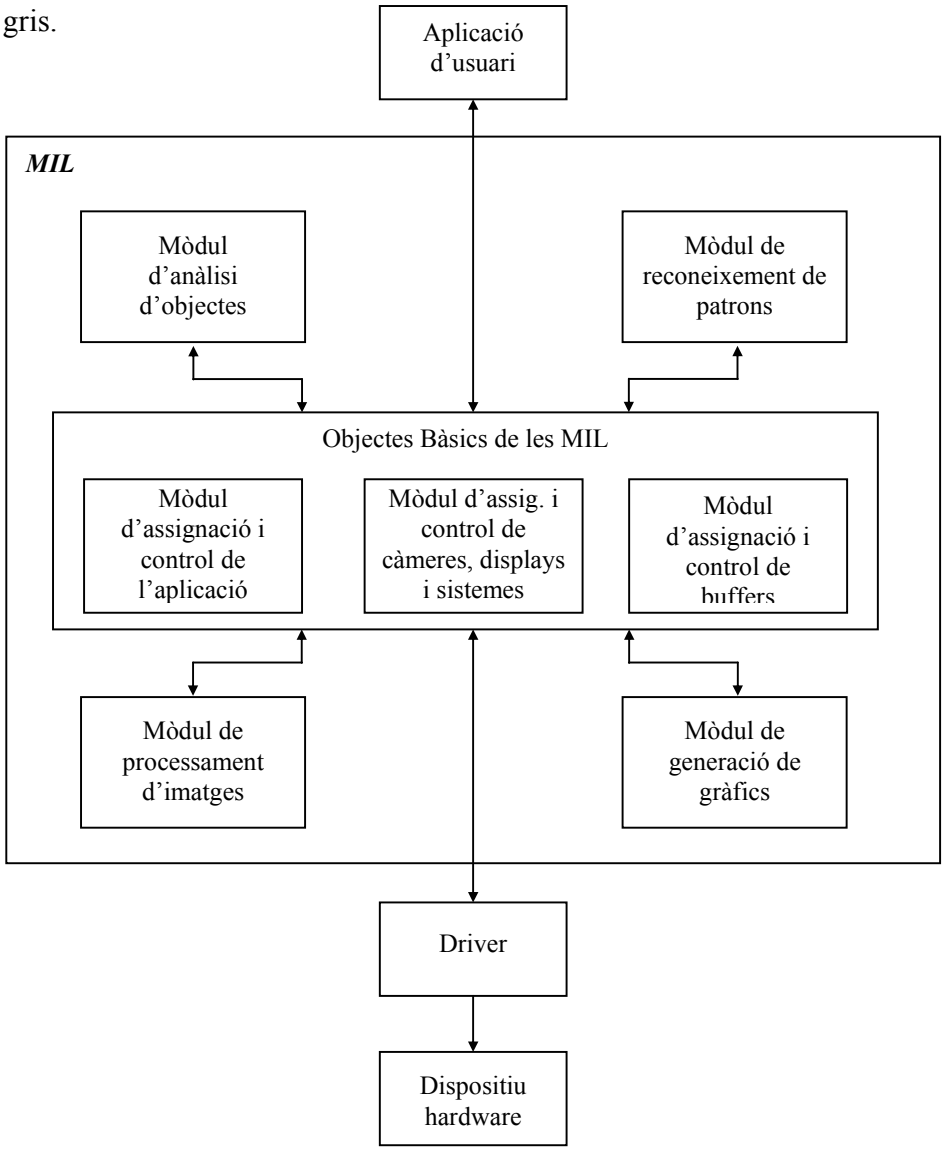

Figura 1. Matrox Imaging Library.

- **Mòdul d'assignació i control de l'aplicació:** inclou les funcions necessàries per fer les inicialitzacions de les MIL, el control d'errors i el seguiment de l'aplicació.

  Exemples: MapAllocDefault(), MappControl, MappGetError(), MappTimer()...

- **Mòdul d'assignació i control de càmeres, displays i sistemes:** aquest mòdul inclou les funcions per controlar càmeres, els displays (finestres de visualització d'imatges) i els sistemes (targes de captura). Podem controlar la gravació d'imatges des d'una càmera, associar displays a imatges per tal de veure-les a pantalla, controlar les targes de captura i moltes altres operacions.

  Exemples: MdigGrab(), MdigControl(), MdispSelect(), MsysControl()...

- **Mòdul d'assignació i control de buffers:** amb les funcions d'aquest mòdul podem realitzar totes les operacions bàsiques amb els buffers que contindran les imatges, com llegir i escriure imatges a un fitxer, copiar imatges, extreure els colors de les imatges, crear noves imatges a partir d'altres, etc.

  Exemples: MbufAllocColor(), MbufCopyColor(), MbufSave(), MbufGetColor()...

- **Mòdul d'anàlisis d'objectes:** les funcions d'aquest mòdul permeten realitzar totes les operacions relacionades amb la detecció i anàlisis de zones de la imatge que anomenem objectes.

  Exemples: MblobAllocResult(), MblobControl(), MblobGetResult(), MblobSelect()...

- **Mòdul de reconeixement de patrons:** inclou les funcions que ens permeten resoldre problemes com l'alineació, mesura i la inspecció d'objectes. Per exemple podríem buscar les coordenades d'un patró en una imatge o calcular el nombre de vegades que hi apareix.

  Exemples: MpatFindModel(), MpatGetNumber(), MpatSetAngle(), MpatInquire()...

- **Mòdul de processament d'imatges:** a aquest mòdul pertanyen totes les funcions que ens permeten la manipulació i processat d'imatges. Es possible obtenir càlculs estadístics (histograma, valors mínims i màxims...), aplicar filtres espacials, realitzar operacions morfològiques (erosió, dilatació...) i d'altres com l'accentuació de contorns, la combinació d'imatges, l'etiquetatge i pràcticament qualsevol operació que se'ns acudeixi.

  Exemples: MimHistogram(), MimResize(), MimConvolve(), MimDilate()...

- **Mòdul de generació de gràfics:** en aquest mòdul trobem les funcions que s'utilitzen per incloure dibuixos i anotacions a les imatges. En una imatge hi podrem escriure text i dibuixar-hi rectangles, arcs, línies i punts.

  Exemples: MgraText(), MgraRect(), MgraArc(), MgraLine(), MgraDot()...

Les MIL proporcionen una completa transparència sobre el maneig dels seus components i són independents del hardware utilitzat. Això fa possible que una aplicació MIL funcioni amb qualsevol tarja VGA compatible VESA o qualsevol tarja Matrox, sota diferents entorns, com DOS, Windows i Windows NT. Per tal d'obtenir un major rendiment de les MIL es aconsellable utilitzar targes acceleradores de la Matrox, com per exemple la Meteor RGB, que és la que s'ha utilitzat en aquest projecte.

2. Conceptes.

Les MIL utilitza cinc tipus d'objectes: aplicacions, sistemes, càmeres, displays i buffers d'imatges. Aquests objectes són mapejats com a objectes virtuals. D'aquesta forma es poden assignar, es a dir reservar espai per la seva utilització, quan s'han de manipular, i es poden alliberar quan ja no els necessitem.

Per tal de simplificar la construcció d'aplicacions MIL, aquesta assignació es fa normalment amb paràmetres per defecte, per exemple el format de gravació de la càmera o el tamany dels buffers, però el programador pot canviar aquest paràmetres quan cregui convenient, per exemple per utilitzar un tamany d'imatge concret.

En aplicació MIL podem assignar diversos sistemes, es a dir podem tenir diverses targes encara que normalment només disposarem d'una, i en cada sistema podem assignar varis buffers, displays i càmeres (Figura 2).

- **Aplicació:** conté els elements necessaris pel control i execució de l'entorn MIL.

- **Sistema:** les MIL utilitzen el concepte de sistema per referir-se a les targes de captura d'imatges.

- **Buffers:** són simplement espais per emmagatzemar les imatges. Les característiques principals d'un buffer són el seu tipus (color o gris) i les seves dimensions (x, y i numero de bandes de color RGB).

- **Displays:** un display defineix una finestra de visualització a la pantalla. Si volem que una imatge es vegi per pantalla haurem d'associar el buffer d'aquesta imatge a una display. Aleshores, totes les modificacions que realitzem a la imatge es veuran per pantalla. Quan ja no volem veure la imatge haurem de deseleccionar el buffer del display.

- **Càmeres:** mitjançant la càmera que connectem a la tarja, capturem les imatges que col·locarem en els buffers.

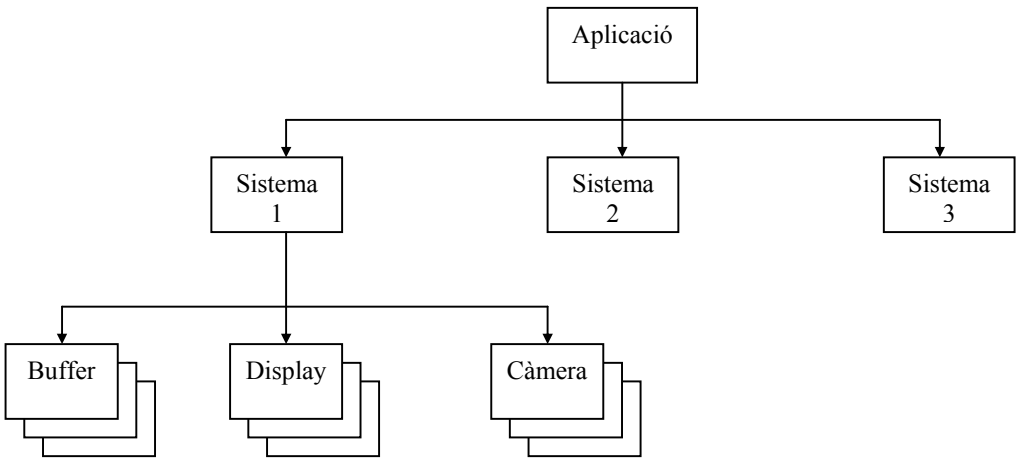

Figura 2. Objectes de les MIL.

3. Exemples d'aplicació.

La forma més útil per consolidar i aclarir els conceptes teòrics és la realització d'exemples, per tant a continuació construirem una aplicació que utilitza les llibreries MIL. Els següents exemples expliquen com fer servir les diferents funcions que es troben a la llibreria.

3.1. Carregar una imatge.

Aquesta aplicació serà molt senzilla: carregarà del disc dur una imatge de tamany 384 x 288 pixels, la visualitzarà per pantalla i la guardarà en format TIF en un fitxer a disc.

Pas 1. Capçalera

El primer pas és incloure el fitxer de capçalera **mil.h**[1], a més dels altres fitxer *.h que siguin necessaris. Aquest fitxer conté totes les definicions de constants, definicions de tipus i prototipus de les funcions pel funcionament de les MIL.

[1] Per tal de muntar (linkar) l'aplicació s'hauran d'incloure les llibreries (fitxers *.lib) situades en el directori on haguem instal·lat les MIL.

```
#include <conio.h>
#include <stdio.h>
#include <mil.h>
```

Pas 2. Declaració de variables

Haurem de declarar varies constants per guardar el tamany de la imatge i els noms dels fitxers, i cinc variables de tipus MIL_ID (el tipus utilitzant per les MIL) que seran l'aplicació, el sistema, el buffer per la imatge i el display per visualitzar-la.

```
#define FITXER_ENT "titanic.mim"
#define FITXER_SOR "titanic.tif"
#define AMPLADA 256
#define ALCADA 256

MIL_ID
   MilApplication,          // Identificador de l'aplicació
   MilSystem,               // Identificador del sistema
   MilDisplay,              // Display de la imatge
   MilImage;                // Imatge
```

Pas 3. Inicialitzacions

A continuació hem d'inicialitzar les variables anteriors. Aquí cal dir que normalment les funcions de les MIL tenen bastants paràmetres, però no tots ells són utilitzables o es pot indicar que els agafi per defecte. Aquestes circumstàncies s'indiquen amb les constants predefinides M_NULL i M_DEFAULT respectivament.

El primer que s'ha de fer és inicialitzar l'aplicació i el sistema. El tipus d'aquesta inicialització serà per defecte, cosa que li indiquem amb M_SETUP.

```
MappAllocDefault (M_SETUP, &MilApplication, &MilSystem, M_NULL, M_NULL,
                  M_NULL);
```

Ara iniciem el display: cal indicar el sistema, igual que en totes les funcions que utilitzin buffers o la càmera, i el número de display.

```
MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT, &MilDisplay);
```

Per inicialitzar la imatge hem de dir: sistema, número de bandes de color RGB, en aquest cas les tres, l'amplada i l'alçada de la imatge, els bits de color i el seu rang, i la utilitat que donarem al buffer: per una imatge que serà gravada i visualitzada.

```
MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                M_IMAGE+M_PROC+M_DISP, &MilImage);
```

Pas 4. Carreguem la imatge

Per carregar la imatge hem d'indicar el nom de l'arxiu i el buffer.

```
MbufLoad (FITXER_ENT, MilImage);
```

Pas 5. Visualització de la imatge

Per tal de visualitzar la imatge per pantalla hem d'associar la imatge al display.

```
MdispSelect (MilDisplay, MilImage);
```

Pas 6. Guardar la imatge en un fitxer

Indicarem el nom del fitxer i el format amb que volem guardar la imatge.

```
MbufExport (FITXER_SOR, M_TIFF, MilImage);
```

Pas 7. Alliberament de les variables

Finalment cal alliberar tots els dispositius utilitzats: aplicació, sistema, buffer, display i càmera.

```
MdispFree (MilDisplay);
MbufFree (MilImage);
MdigFree (MilCamera);
MappFreeDefault (MilApplication, MilSystem, M_NULL, M_NULL, M_NULL);
```

D'aquesta forma tan senzilla hem construït una aplicació que utilitza les MIL per gravar una imatge, visualitzar-la i guardar-la en un fitxer. El llistat complet el podem veure a continuació.

```
#include <stdio.h>
#include <conio.h>
#include <mil.h>

#define FITXER_ENT "titanic.mim"
#define FITXER_SOR "titanic.tif"
#define AMPLADA 256
#define ALCADA 256



void main(void) {
       /* Variables MIL */
        MIL_ID
      MilApplication,           // Identificador de l'aplicació
      MilSystem,                // Identificador del sistema
      MilDisplay,               // Display de la imatge
      MilCamera,                // Identificador de la camera
      MilImage;                 // Imatge

       /* Inicialització per treballar amb les MIL */
        MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                          M_NULL,M_NULL, M_NULL);

       /* Inicialitzar display */
        MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
            &MilDisplay);


       /* Inicialitzar buffer */
        MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                M_IMAGE+M_PROC+M_DISP, &MilImage);

       /* Carreguem la imatge */
        MbufLoad (FITXER_ENT, MilImage);

       /* Associació del display amb la imatge, per visualitzar-la */
        MdispSelect (MilDisplay, MilImage);

       /* Guardar la imatge */
        MbufExport (FITXER_SOR, M_TIFF, MilImage);

        printf ("\n\n Última imatge gravada a test.tiff");
        printf ("\n\n Prem <ENTER> per acabar.");
        getch();

       /* Alliberament de la memòria i dels dispositius utilitzats */
        MdispFree (MilDisplay);
        MbufFree (MilImage);
        MdigFree (MilCamera);
        MappFreeDefault (MilApplication, MilSystem, M_NULL, M_NULL,
                         M_NULL);
```

```
}
```

3.2. Components RGB.

En una imatge qualsevol tenim tres plans. Cadascun d'aquests plans conté la informació corresponent al color Vermell, Verd i Blau (RGB). Per tant, ens pot interessar obtenir alguna de les tres components RGB i així poder tractar-la amb algun càlcul.

Les llibreries MIL ens ofereixen una instrucció que ens permet separar la component que nosaltres desitgem. Per indicar quina component de color ho fem amb les constants M_RED, M_GREEN i M_BLUE. La instrucció en concret és:

```
MbufCopyColor (MilImage1, MilImageVer, M_RED);
```

Les variables MilImage1 i MilImageVer emmagatzemen la imatge font i la component vermella.

El codi de l'exemple és el següent:

```
#include <stdio.h>
#include <conio.h>
#include <mil.h>

#define FITXER_ENT "bird.mim"
#define AMPLADA 256
#define ALCADA 240

void main(void) {
       /* Variables MIL */
        MIL_ID
           MilApplication,          // Identificador de l'aplicació
           MilSystem,              // Identificador del sistema
           MilDisplay0,            // Display de la imatge original
           MilDisplay1,            // Display de la imatge tractada
           MilImage,               // Imatge original
           MilImageVer;            // Component de vermella

       /* Inicialització per treballar amb les MIL */
        MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                          M_NULL,M_NULL, M_NULL);

       /* Inicialitzar display */
        MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
```

```
                    &MilDisplay0);
          MdispAlloc (MilSystem, M_DEV1, M_DISPLAY_SETUP, M_DEFAULT,
                    &MilDisplay1);

         /* Iniciem els buffers de cada variable de les imatges */
          MbufAllocColor (MilSystem, 3, AMPLADA, ALCADA, 8+M_UNSIGNED,
                          M_IMAGE+M_PROC+M_DISP, &MilImage);
          MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                       M_IMAGE+M_PROC+M_DISP, &MilImageVer);

         /* Carreguem la imatge */
          MbufLoad (FITXER_ENT, MilImage);

         /* Seleccionen la component vermella per mostrar-la */
          MbufCopyColor (MilImage, MilImageVer, M_RED);

        /*Associació dels displays amb la imatges,per visualitzar-la */

          MdispSelect (MilDisplay0,MilImage);
          MdispSelect (MilDisplay1,MilImageVer);

          printf ("\n\n Imatge original i component Vermella.");
          printf ("\n\n Prem <ENTER> per acabar.");
          getch();



         /* Alliberament de la memòria i dels dispositius utilitzats */
          MdispFree (MilDisplay0);
          MdispFree (MilDisplay1);
          MbufFree (MilImage);
          MbufFree (MilImageVer);
          MappFreeDefault (MilApplication,MilSystem,M_NULL,M_NULL,M_NULL);
}
```

3.3. Canvi de Components (RGB → HLS).

Hi ha una altre forma de guardar la informació que té una imatge. En comptes de guardar les components de color, el que farem serà guardar les components de To, Lluminància i Saturació (HLS). Per poder realitzar modificacions sobre aquestes components, el primer que hem de fer és transformar la imatge de format RGB a HLS. A continuació seleccionen una de les components i l'operem per obtenir el resultat desitjat.

Per canviar el format de les components fem servir la següent instrucció, on MilImage0 i MilImage1 guarden la imatge RGB i HLS.

```
MimConvert (MilImage0, MilImage1, M_RGB_TO_HLS);
```

A continuació el que hem de fer és separar la component que ens interessa tractar. Com tenim tres components, hem d'indicar quina d'elles volem i ho fem amb les constants M_LUMINANCE (lluminància), M_HUE (to) i M_SATURATION (saturació). Per fer-ho tenim la següent instrucció:

```
MbufChildColor (MilImage1, M_LUMINANCE, &MilImage1Llu);
```

On MilImage1 conté la imatge HLS i MilImage1Llu conté la component de la lluminància.

Tot seguit mostrem el codi que fa servir aquestes dues instruccions i que ens permet veure la component de la lluminància i la imatge inicial.

```
#include <stdio.h>
#include <conio.h>
#include <mil.h>

#define FITXER_ENT "pilota.mim"
#define AMPLADA 382
#define ALCADA 286

void main(void) {
       /* Variables MIL */
        MIL_ID
           MilApplication,           // Identificador de l'aplicació
           MilSystem,               // Identificador del sistema
           MilDisplay0,             // Display de la imatge original
           MilDisplay1,             // Display de la imatge tractada
           MilImage0,               // Imatge original
           MilImage1,               // Imatge tractada
           MilImage1Llu;            // Component de la lluminància

       /* Inicialització per treballar amb les MIL */
        MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                          M_NULL,M_NULL, M_NULL);

       /* Inicialitzar display */
        MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
                    &MilDisplay0);
        MdispAlloc (MilSystem, M_DEV1, M_DISPLAY_SETUP, M_DEFAULT,
                    &MilDisplay1);

       /* Iniciem els buffers de cada variable de les imatges */
        MbufAllocColor (MilSystem, 3, AMPLADA, ALCADA, 8+M_UNSIGNED,
                        M_IMAGE+M_PROC+M_DISP, &MilImage0);
        MbufAllocColor (MilSystem, 3, AMPLADA, ALCADA, 8+M_UNSIGNED,
                        M_IMAGE+M_PROC+M_DISP, &MilImage1);
```

```
            MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                         M_IMAGE+M_PROC+M_DISP, &MilImage1Llu);

           /* Carreguem la imatge */
            MbufLoad (FITXER_ENT, MilImage0);

           /* Convertim la imatge carregada de RGB a HLS */
            MimConvert (MilImage0, MilImage1, M_RGB_TO_HLS);

           /*Seleccionen la component de la lluminància per mostrar-la */

            MbufCopyColor (MilImage1, MilImage1Llu, M_LUMINANCE);

         /*Associació dels displays amb la imatges, per visualitzar-la */

            MdispSelect (MilDisplay0, MilImage0);
            MdispSelect (MilDisplay1, MilImage1Llu);

            printf ("\n Imatge original i component Lluminància (HLS).");
            printf ("\n\n Prem <ENTER> per acabar.");
            getch ();
            getch ();

           /* Alliberament de la memòria i dels dispositius utilitzats */
            MdispFree (MilDisplay0);
            MdispFree (MilDisplay1);
            MbufFree (MilImage0);
            MbufFree (MilImage1);
            MbufFree (MilImage1Llu);
            MappFreeDefault(
            MilApplication,MilSystem,M_NULL,M_NULL,M_NULL);
}
```

3.4. Resta d'imatges. Operacions Aritmètiques.

Per poder detectar objectes podem fer la resta d'una imatge que només conté el fons i d'una altre que conté el fons amb uns objectes a sobre. Si fem la resta veurem que els fons pràcticament s'anul·la i només queden els objectes.

Aquesta operació de resta la fem amb la següent operació, on MilImage1 és la imatge amb els objectes, MilImageFons conté el fons que volem restar i MilImageRes emmagatzemarà el resultat.

```
MimArith (MilImage1, MilImageFons, MilImageRes, M_SUB_ABS);
```

A més a més de fer la resta, podem fer altres operacions canviant l'última constant de la funció. Podem fer sumes, restes, multiplicacions, divisions i les operacions lògiques AND, NAND, OR, XOR, NOR i XNOR.

A continuació presentem el codi de l'exemple:

```
#include <stdio.h>
#include <conio.h>
#include <mil.h>

#define FITXER_ENT1 "rogi_color.tif"
#define FITXER_ENT2 "rogi_campcolor.tif"
#define AMPLADA 384
#define ALCADA 288

void main(void) {
       /* Variables MIL */
        MIL_ID
           MilApplication,          // Identificador de l'aplicació
           MilSystem,               // Identificador del sistema
           MilDisplay0,             // Display de la imatge original
           MilDisplay1,             // Display de la imatge tractada
           MilImageFons,            // Imatge original del fons
           MilImage1,               // Imatge original amb objectes
           MilImageRes;             // Imatge resultat de la resta

       /* Inicialització per treballar amb les MIL */

        MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                          M_NULL,M_NULL, M_NULL);
```

```
        /* Inicialització del display */
         MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
                     &MilDisplay0);
         MdispAlloc (MilSystem, M_DEV1, M_DISPLAY_SETUP, M_DEFAULT,
                     &MilDisplay1);

        /* Iniciem els buffers de cada variable de les imatges */
         MbufAllocColor (MilSystem, 3, AMPLADA, ALCADA, 8+M_UNSIGNED,
                         M_IMAGE+M_PROC+M_DISP, &MilImageFons);
         MbufAllocColor (MilSystem, 3, AMPLADA, ALCADA, 8+M_UNSIGNED,
                         M_IMAGE+M_PROC+M_DISP, &MilImage1);
         MbufAllocColor (MilSystem, 3, AMPLADA, ALCADA, 8+M_UNSIGNED,
                         M_IMAGE+M_PROC+M_DISP, &MilImageRes);

        /* Carreguem les imatges */
         MbufLoad (FITXER_ENT2, MilImageFons);
         MbufLoad (FITXER_ENT1, MilImage1);

        /* Traiem fons de imatge amb objectes i guardem resultat */
         MimArith (MilImage1, MilImageFons, MilImageRes, M_SUB_ABS);



        /* Associació dels displays amb la imatges, per visualitzar-la */
         MdispSelect (MilDisplay0,MilImage1);
         MdispSelect (MilDisplay1,MilImageRes);

         printf ("\n\n Imatge original i resultat de treure el fons.");
         printf ("\n\n Prem <ENTER> per acabar.");
         getch();

        /* Alliberament de la memòria i dels dispositius utilitzats */
         MdispFree (MilDisplay0);
         MdispFree (MilDisplay1);
         MbufFree (MilImage1);
         MbufFree (MilImageFons);
         MbufFree (MilImageRes);
         MappFreeDefault (MilApplication, MilSystem, M_NULL, M_NULL,
                          M_NULL);
}
```

3.5. Binarització.

En algunes aplicacions ens pot interessar que algunes que les imatges en blanc i negre tinguin valors binaris (0 i 1). Per realitzar aquesta operació tenim la funció següent de la llibreria de les MIL. Aquesta funció ens permet fer la binarització d'una imatge a partir d'un determinat valor (LLINDAR).

```
MimBinarize (MilImage1, MilImage1, M_GREATER_OR_EQUAL, LLINDAR, M_NULL);
```

Tanmateix en aquest exemple fem servir una altre funció que ens permet fer un filtre passa baixes. Fent aquest el filtre aconseguim que desaparegui tot el soroll que tingui la imatge. Aquesta funció és la següent:

```
MimConvolve (MilImage0, MilImage1, M_SMOOTH);
```

L'exemple que il·lustra el funcionament d'aquestes dues funcions és el següent:

```
#include <stdio.h>
#include <conio.h>
#include <mil.h>

#define FITXER_ENT "titanic.mim"
#define AMPLADA 256
#define ALCADA 256

/* Definició del valor per on farem la binarització */
#define LLINDAR 128

void main (void) {
       /* Variables MIL */
        MIL_ID
           MilApplication,          // Identificador de l'aplicació
           MilSystem,               // Identificador del sistema
           MilDisplay0,             // Display de la imatge original
           MilDisplay1,             // Display de la imatge binaritzada
           MilImage0,               // Imatge
           MilImage1;               // Resultat de la binarització

       /* Inicialització per treballar amb les MIL */
        MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                          M_NULL,M_NULL, M_NULL);



       /* Inicialitzar display */
        MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
                    &MilDisplay0);
```

```
            MdispAlloc (MilSystem, M_DEV1, M_DISPLAY_SETUP, M_DEFAULT,
                        &MilDisplay1);

           /* Inicialitzar buffer */
            MbufAllocColor (MilSystem, 3, AMPLADA, ALCADA, 8+M_UNSIGNED,
                            M_IMAGE+M_PROC+M_DISP, &MilImage0);
            MbufAllocColor (MilSystem, 3, AMPLADA, ALCADA, 8+M_UNSIGNED,
                            M_IMAGE+M_PROC+M_DISP, &MilImage1);

           /* Carreguem la imatge */
            MbufLoad (FITXER_ENT, MilImage0);

           /* Associació del display amb la imatge, per visualitzar-la */
            MdispSelect (MilDisplay0, MilImage0);
            MdispSelect (MilDisplay1, MilImage1);

           /* Esborrem el soroll amb un filtre passa baixes tipus SMOOTH */
            MimConvolve (MilImage0, MilImage1, M_SMOOTH);

           /* Binaritzem imatge amb LLINDAR, fons negre i resta en blanc */
            MimBinarize (MilImage1, MilImage1, M_LESS_OR_EQUAL, LLINDAR,
                         M_NULL);

            printf ("\n\n Imatge original i binarització.");
            printf ("\n\n Prem <ENTER> per acabar.");
            getch ();

           /* Alliberament de la memòria i dels dispositius utilitzats */
            MdispFree (MilDisplay0);
            MdispFree (MilDisplay1);
            MbufFree (MilImage0);
            MbufFree (MilImage1);
            MappFreeDefault (MilApplication, MilSystem, M_NULL, M_NULL,
                             M_NULL);
}
```

3.6. Histograma.

Un histograma ens permet mesurar la freqüència d'ocurrència de cada nivell de gris definit en una imatge. Tenint en compte aquesta mesura podrem decidir com tractar aquesta imatge, per exemple, podrem decidir un valor per fer una llindarització.

La funció que farem servir és:

```
MimAllocResult (M_DEFAULT, NUM_INTENSITATS, M_HIST_LIST, &ResultatHist);

MimHistogram (MilImage, ResultatHist);

MimGetResult (ResultatHist, M_VALUE, ValorsHist);
```

Com podem veure la primera funció ens crea un buffer per tractar les dades resultat de l'histograma. A continuació generem una imatge de l'histograma i la guardem en l'espai reservat. Aquest resultat l'hem de passar a format numèric per poder tractar els valors que trobarem a la taula ValorsHist.

El codi que dona exemple a aquesta funció és:

```
#include <stdio.h>
#include <conio.h>
#include <mil.h>

#define FITXER_ENT "titanic.mim"
#define AMPLADA 256
#define ALCADA 256

/* Definició del rang d'intensitats que pot assolir l'histograma */
#define NUM_INTENSITATS 256

void main (void) {
       /* Variables MIL */
        MIL_ID
           MilApplication,           // Identificador de l'aplicació
           MilSystem,                // Identificador del sistema
           MilDisplay,               // Display de la imatge
           MilImage,                 // Imatge
           ResultatHist;             // Resultat de l'histograma

       /* Valors de l'histograma */
        long ValorsHist [NUM_INTENSITATS];

        short i;
```

```
        /* Inicialització per treballar amb les MIL */
         MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                           M_NULL,M_NULL, M_NULL);

        /* Inicialitzar display */
         MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
                     &MilDisplay);

        /* Inicialitzar buffer */
         MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                      M_IMAGE+M_PROC+M_DISP, &MilImage);

        /* Carreguem la imatge */
         MbufLoad (FITXER_ENT, MilImage);

        /* Associació del display amb la imatge, per visualitzar-la */
         MdispSelect (MilDisplay, MilImage);


        /* Reservem espai per guardar el resultat de l'histograma */
         MimAllocResult (M_DEFAULT, NUM_INTENSITATS, M_HIST_LIST,
                         &ResultatHist);

        /* Generem l'histograma */
         MimHistogram (MilImage, ResultatHist);

        /* Generem el resultat numèric de l'histograma */
         MimGetResult (ResultatHist, M_VALUE, ValorsHist);

        /* Escrivim els valors obtinguts a l'histograma */
         printf ("\n Prem <RETURN> per escriure el resultat de
                 l'histograma.");
         getch ();
         printf ("\n");
         for (i=0, i<NUM_INTENSITATS, i++) {
            printf ("%3d:%6ld\n",i,ValorsHist[i]);
            if ((i%20)==19) {
               printf ("\n Prem <RETURN> per continuar.");
               getch ();
               printf ("\n");
            }
         }

        /* Alliberem els recursos adquirits */
         MimFree (ResultatHist);
         MbufFree (MilImage);
         MdispFree (MilDisplay);
         MappFreeDefault (MilApplication, MilSystem, M_NULL, M_NULL,
                          M_NULL);
}
```

3.7. Equalització d'Histograma.

Aquest exemple ens mostra com obtenir un histograma i, a més a més, millorarem la disposició dels nivells de gris pel rang disponible. Aquests canvis faran que la nostra imatge tingui un contrast més realçat pel nostre ull sense modificar les dades.

La funció que fem servir en aquest cas és la següent:

```
MimHistrogramEqualize (MilImage0,MilImage1,M_UNIFORM,ALFA,MINIM, MAXIM);
```

On MilImage0 i MilImage1 tenen la imatge original i resultat, ALFA és un paràmetre que no farem servir en aquest exemple, MÍNIM i MÀXIM és permeten definir el rang on volem fer l'equalització.

I el codi que ens mostra el funcionament és:

```
#include <stdio.h>
#include <conio.h>
#include <mil.h>

#define FITXER_ENT "titanic.mim"
#define AMPLADA 256
#define ALCADA 256

#define MINIM 0
#define MAXIM 255
#define ALFA 1

/* Definició del rang d'intensitats que pot assolir l'histograma */
#define NUM_INTENSITATS 256

void main (void) {
       /* Variables MIL */
        MIL_ID
           MilApplication,           // Identificador de l'aplicació
           MilSystem,                // Identificador del sistema
           MilDisplay0,              // Display de la imatge
           MilDisplay1,              // Display de la imatge
           MilImage0,                // Imatge
           MilImage1;                // Imatge Resultat

       /* Inicialització per treballar amb les MIL */
        MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                          M_NULL,M_NULL, M_NULL);

       /* Inicialitzar display */
        MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
                    &MilDisplay0);
```

```
        MdispAlloc (MilSystem, M_DEV1, M_DISPLAY_SETUP, M_DEFAULT,
                    &MilDisplay1);

       /* Inicialitzar buffer */
        MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                        M_IMAGE+M_PROC+M_DISP, &MilImage0);
        MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                        M_IMAGE+M_PROC+M_DISP, &MilImage1);

       /* Carreguem la imatge */
        MbufLoad (FITXER_ENT, MilImage0);

       /* Associació del display amb la imatge, per visualitzar-la */
        MdispSelect (MilDisplay0, MilImage0);



/* Generem l'equalització d'histograma */
        MimHistogramEqualize (MilImage0, MilImage1, M_UNIFORM, ALFA,
                              MINIM, MAXIM);

        printf ("\n Prem <RETURN> per veure el resultat de
                 l'equalització.");
        getch ();

       /* Mostrem el resultat */
        MdispSelect (MilDisplay1, MilImage1);

        printf ("\n Prem <RETURN> per acabar.");
        getch ();

       /* Alliberem els recursos adquirits */
        MbufFree (MilImage0);
        MbufFree (MilImage1);
        MdispFree (MilDisplay0);
        MdispFree (MilDisplay1);
        MappFreeDefault (MilApplication, MilSystem, M_NULL, M_NULL,
                         M_NULL);
}
```

3.8. Filtres (I).

En l'exemple 3.5 hem vist com podíem fer un filtre passa baixes, però això és una mica limitat perquè la llibreria de les MIL ens permet treballar amb altres filtres i/o crear-los. En aquest apartat aprendrem una altre forma d'utilitzar els filtres. El filtre que hem vist fins ara ens permet reduir el soroll de tipus Gaussià o sorolls d'alta freqüència sistemàtics. I el que veurem ara ens permetrà esborrar el soroll de tipus 'sal i pebre' ("salt-and-pepper").

Aquest filtre ve realitzat per la funció MimRank() i en concret pel paràmetre M_MEDIAN. I funciona amb els següents paràmetres, on MilImage0 i MilImage1 emmagatzemen la imatge original i la resultat, M_3X3_RECT ens indica el tipus de màscara.

```
MimRank (MilImage0, MilImage1, M_3X3_RECT, M_MEDIAN, M_GRAYSCALE);
```

El codi de l'exemple és el següent:

```
#include <stdio.h>
#include <conio.h>
#include <mil.h>

#define FITXER_ENT "soroll.mim"
#define AMPLADA 640
#define ALCADA 480

void main (void) {
       /* Variables MIL */
        MIL_ID
           MilApplication,            // Identificador de l'aplicació
           MilSystem,                 // Identificador del sistema
           MilDisplay0,               // Display de la imatge
           MilDisplay1,               // Display de la imatge
           MilImage0,                 // Imatge
           MilImage1;                 // Resultat del filtre

       /* Inicialització per treballar amb les MIL */
        MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                          M_NULL,M_NULL, M_NULL);

       /* Inicialitzar display */
        MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
                    &MilDisplay0);
        MdispAlloc (MilSystem, M_DEV1, M_DISPLAY_SETUP, M_DEFAULT,
                    &MilDisplay1);

       /* Inicialitzar buffer */
        MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                     M_IMAGE+M_PROC+M_DISP, &MilImage0);
        MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
```

```
                        M_IMAGE+M_PROC+M_DISP, &MilImage1);

        /* Carreguem la imatge */
         MbufLoad (FITXER_ENT, MilImage0);

        /* Associació del display amb la imatge, per visualitzar-la */
         MdispSelect (MilDisplay0, MilImage0);
         MdispSelect (MilDisplay1, MilImage1);

        /* Apliquem el filtre sobre la imatge original */
         MimRank (MilImage0,MilImage1,M_3X3_RECT, M_MEDIAN, M_GRAYSCALE);

         printf ("\n Imatge original i filtrada amb un filtre de la
                  mediana.");
         printf ("\n Prem <RETURN> per continuar.");
         getch ();

        /* Alliberem els recursos adquirits */
         MbufFree (MilImage0);
         MbufFree (MilImage1);
         MdispFree (MilDisplay0);
         MdispFree (MilDisplay1);
         MappFreeDefault (MilApplication, MilSystem, M_NULL, M_NULL,
                          M_NULL);
}
```

3.9. Filtres (II). Màscares.

Una manera de crear els filtres és fent servir una finestra on cada posició conté el pes per realitzar una mitja ponderada o altres operacions. Tot seguit fem la convolució entre els punts que diu la finestra.

Per exemple podem tenir un filtre passa altes si fem servir la següent finestra:

-1	-1	-1
-1	8	-1
-1	-1	-1

Per obtenir el resultat hem de passar aquesta finestra per cada punt de la imatge i fer les operacions que ens indica. És a dir, multiplicar per vuit el punt a tractar i restar-li tots els seus veïns.

Aquests filtres també es poden fer amb les funcions que ens dona la llibreria de les MIL. Tenim diferents funcions que ens permeten introduir la finestra i d'altres que ens diuen com s'han de tractar els veïns alhora de fer la convolució.

La finestra que hem vist anteriorment es diu que està normalitzada perquè la suma de tots els valors donen zero. En canvi, els valors que farem servir en el següent exemple no estan normalitzats i, per tant, farem servir una altre funció que ens els normalitzarà.

Les instruccions que fem servir són les següents:

```
MbufAlloc2d (M_DEFAULT, ALCADA_FINESTRA, AMPLADA_FINESTRA,
             PROFUNDITAT_FINESTRA+M_UNSIGNED, M_KERNEL, &MilFinestra);

MbufPut (MilFinestra, Finestra);

MbufControlNeighborhood (MilFinestra, M_NORMALITZATION_FACTOR, 16);

MimConvolve (MilImage0, MilImage1, MilFinestra);
```

Amb la primera, el que fem és preparar espai per emmagatzemar la finestra. A continuació emmagatzemem la informació de la finestra. Com abans hem dit, la finestra no està normalitzada i, per tant l'hem de normalitzar amb la funció que es troba a continuació. Per últim, realitzem la convolució.

A continuació tenim el codi de l'exemple que llustra aquestes operacions:

```
#include <stdio.h>
#include <conio.h>
#include <mil.h>

#define FITXER_ENT "wafer.mim"
#define AMPLADA 512
#define ALCADA 480

/* Informació del filtre */
#define AMPLADA_FINESTRA 3
#define ALCADA_FINESTRA 3
#define PROFUNDITAT_FINESTRA 8

/* Informació de la taula del filtre que volem realitzar */
unsigned char Mascara_convolucio[ALCADA_FINESTRA][AMPLADA_FINESTRA]=
{{1,2,1},
{2,4,2},
{1,2,1}};

void main (void) {
       /* Variables MIL */
        MIL_ID
           MilApplication,           // Identificador de l'aplicació
           MilSystem,                // Identificador del sistema
           MilDisplay0,              // Display de la imatge
           MilDisplay1,              // Display de la imatge
           MilImage0,                // Imatge
           MilImage1,                // Resultat del filtre
           MilFinestra;              // Finestra del filtre

       /* Inicialització per treballar amb les MIL */
        MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                          M_NULL,M_NULL, M_NULL);

       /* Inicialitzar display */
        MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
                    &MilDisplay0);
        MdispAlloc (MilSystem, M_DEV1, M_DISPLAY_SETUP, M_DEFAULT,
                    &MilDisplay1);

       /* Inicialitzar buffer */
        MbufAllocColor (MilSystem, 3, AMPLADA, ALCADA, 8+M_UNSIGNED,
                        M_IMAGE+M_PROC+M_DISP, &MilImage0);
        MbufAllocColor (MilSystem, 3, AMPLADA, ALCADA, 8+M_UNSIGNED,
                        M_IMAGE+M_PROC+M_DISP, &MilImage1);

       /* Carreguem la imatge */
```

```
        MbufLoad (FITXER_ENT, MilImage0);

       /* Associació del display amb la imatge, per visualitzar-la */
        MdispSelect (MilDisplay, MilImage0);

        printf ("\n Prem <RETURN> per continuar amb el filtre.");
        getch ();

       /* Situem la finestra del filtre en un buffer */
        MbufAlloc2d (M_DEFAULT, ALCADA_FINESTRA, AMPLADA_FINESTRA,
                     PROFUNDITAT_FINESTRA+M_UNSIGNED, M_KERNEL,
                     &MilFinestra);

       /* Introduïm la finestra en el buffer */
        MbufPut (MilFinestra, Mascara_convolucio);

       /* Introduïm un valor per normalitzar el filtre */
        MbufControlNeighborhood (MilFinestra, M_NORMALIZATION_FACTOR,
                                 16);

       /* Realitzem convolució per obtenir resultat d'aplicar filtre */
        MimConvolve (MilImage0, MilImage1, MilFinestra);

       /* Associació del display amb la imatge, per visualitzar-la */
        MdispSelect (MilDisplay, MilImage1);

        printf ("\n Prem <RETURN> per acabar.");
        getch ();

       /* Alliberem els recursos adquirits */
        MbufFree (MilImage0);
        MbufFree (MilImage1);
        MbufFree (MilFinestra);
        MdispFree (MilDisplay0);
        MdispFree (MilDisplay1);
        MappFreeDefault (MilApplication, MilSystem, M_NULL, M_NULL,
                         M_NULL);
}
```

3.10. Millora i detecció de les vores.

En l'exemple anterior hem vist la funció MimConvolve, i en concret una de les seves opcions. A continuació veurem altres opcions que estan orientades a la millora i detecció de les vores (línies que separen dues zones de nivell de gris molt diferent).

Podem millorar les vores si fem servir alguna de les dues següents constants:

- M_SHARPEN
- M_SHARPEN2

Aquesta operació pot ser que no produeixi els resultats desitjats perquè també augmenta el soroll que te la imatge. Tanmateix, podem obtenir el mateix resultat si fem servir detecció de vores, en concret el de Laplace, i aquest resultat el sumem a la imatge original.

A més a més, d'aquestes operacions podem realitzar detecció de vores fen servir les següents constants:

- M_HORIZ_EDGE: detecció horitzontal de vores
- M_VERT_EDGE: detecció vertical de vores
- M_LAPLACIAN_EDGE: detecció de vores Laplacià de tipus 1
- M_LAPLACIAN_EDGE2: detecció de vores Laplacià de tipus 2
- M_EDGE_DETECT: detecció de vores pel gradient de tipus 1
- M_EDGE_DETECT2: detecció de vores pel gradient de tipus 2

La instrucció que farem servir és la següent, on MilImage és la imatge original i MilImage1 el resultat de l'operació. La constant que ve a continuació pot ser qualsevol de les anteriorment esmentada.

```
MimConvolve (MilImage0, MilImage1, M_HORIZ_EDGE);
```

El codi que llustra aquest exemple el tenim a continuació:

```
#include <conio.h>
#include <stdio.h>
#include <mil.h>

#define FITXER_ENT "wafer.mim"
#define AMPLADA 512
#define ALCADA 480

void main (void) {
```

```
        /* Variables MIL */
         MIL_ID
            MilApplication,           // Identificador de l'aplicació
            MilSystem,                // Identificador del sistema
            MilDisplay0,              // Display de la imatge
            MilDisplay1,              // Display de la imatge
            MilImage0,                // Imatge
            MilImage1;                // Resultat del filtre

        /* Inicialització per treballar amb les MIL */
         MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                           M_NULL,M_NULL, M_NULL);

        /* Inicialitzar display */
         MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
                     &MilDisplay0);
         MdispAlloc (MilSystem, M_DEV1, M_DISPLAY_SETUP, M_DEFAULT,
                     &MilDisplay1);

        /* Inicialitzar buffer */
         MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                      M_IMAGE+M_PROC+M_DISP, &MilImage0);
         MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                      M_IMAGE+M_PROC+M_DISP, &MilImage1);


        /* Carreguem la imatge */
         MbufLoad (FITXER_ENT, MilImage0);

        /* Associació del display amb la imatge, per visualitzar-la */
         MdispSelect (MilDisplay0, MilImage0);

         printf ("\n Prem <RETURN> per continuar amb la millora de vores
                  horitzontals.");
         getch ();

        /* Realitzem convolució per obtenir resultat d'aplicar millora*/
         MimConvolve (MilImage0, MilImage1, M_HORIZ_EDGE);

        /* Associació del display amb la imatge, per visualitzar-la */
         MdispSelect (MilDisplay1, MilImage1);

         printf ("\n Prem <RETURN> per acabar.");
         getch ();

        /* Alliberem els recursos adquirits */
         MbufFree (MilImage0);
         MbufFree (MilImage1);
         MdispFree (MilDisplay0);
         MdispFree (MilDisplay1);
         MappFreeDefault (MilApplication, MilSystem, M_NULL, M_NULL,
                          M_NULL);
}
```

3.11. Adquisició d'imatges amb una càmera.

En aquest exemple veurem com emmagatzemar en disc una imatge que hem aconseguit d'una càmera. Per tant, el primer que haurem de fer és iniciar la càmera indicant el sistema, el número, el format de les imatges (en concret el PAL), les opcions i el punter al dispositiu. A continuació hem d'indicar en quin canal està connectada la càmera, en aquest exemple en vídeo compost (RCA). I així, les instruccions utilitzades són:

```
MdigAlloc (MilSystem, M_DEV0, "M_PAL", M_DEFAULT, &MilCamera);
MdigChannel (MilCamera, M_RCA);
```

I el codi de l'exemple és el següent:

```
#include <conio.h>
#include <stdio.h>
#include <mil.h>

#define NOM_FITXER "test.tif"
#define AMPLADA 384
#define ALCADA 288

void main (void) {
       /* Variables MIL */
        MIL_ID
           MilApplication,           // Identificador de l'aplicació
           MilSystem,                // Identificador del sistema
           MilDisplay,               // Display de la imatge
           MilCamera,                // Identificador de la camera
           MilImage;                 // Imatge

       /* Inicialització per treballar amb les MIL */
        MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                          M_NULL,M_NULL, M_NULL);

       /* Inicialitzar display */
        MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
                    &MilDisplay);

       /* Inicialitzar la càmera */
        MdigAlloc (MilSystem, M_DEV0, "M_PAL", M_DEFAULT, &MilCamera);
        MdigChannel (MilCamera, M_RCA);

       /* Inicialitzar buffer */
        MbufAllocColor (MilSystem, 3, AMPLADA, ALCADA, 8+M_UNSIGNED,
                        M_IMAGE+M_GRAB+M_DISP, &MilImage);

       /* Gravar la imatge */
        MdigGrab (MilCamera, MilImage);

       /* Associació del display amb la imatge, per visualitzar-la */
```

```
            MdispSelect (MilDisplay, MilImage);

           /* Guardar la imatge */
            MbufExport (NOM_FITXER, M_TIFF, MilImage);

            printf ("\n\n Ultima imatge gravada a test.tif");
            printf ("\n\n Prem <ENTER> per acabar.");
            getch ();

           /* Alliberament de la memòria i dels dispositius utilitzats */
            MdispFree (MilDisplay);
            MbufFree (MilImage);
            MdigFree (MilCamera);
            MappFreeDefault (MilApplication, MilSystem, M_NULL, M_NULL,
                             M_NULL);
}
```

3.12. Altres operacions.

Erosió: aquesta operació ens permet treure capes dels objectes o partícules, extraient pixels estranys i petites partícules de la imatge fent servir una finestra de 3x3. Aquesta operació la podem realitzar amb la instrucció *MimErode ()* i ens permet treballar amb dos modes:

- M_BINARY: cada pixel que no te tots els seus veïns de color blanc (1) el fa negre (0).
- M_GRAYSCALE: cada pixel és reemplaçat pel mínim valor dels seus veïns.

Per tant, la instrucció que farem servir te la següent forma:

```
MimErode (MilImageFont, MilImageDest, NumIte, Mode);
```

On MilImageFont i MilImageDest són les imatges font i destí; NumIte és el número de iteracions que executarem; i Mode poden ser qualsevol de dues anteriors esmentades.

Dilatació: aquesta operació ens permet posar capes als objectes o partícules, fent servir una finestra de 3x3. Pot fer que objectes o partícules erosionades tornin a la seu tamany original, encara que no a la seva forma. En aquest cas, com abans, tenim dos modes de funcionament:

- M_BINARY: si un pixel te cap veí que sigui blanc (1) farà que aquest sigui blanc.
- M_GRAYSCALE: cada pixel és reemplaçat pel màxim dels seus veïns.

Aquesta instrucció te un funcionament igual a l'anterior:

```
MimDilate (MilImageFont, MilImageDest, NumIte, Mode);
```

En aquesta operació, i en l'anterior, iterar equival a realitzar les operacions amb una finestra més gran. És a dir, iterar dues vegades equival a fer una finestra de 5x5. Iterar *i* vegades equival a fer una finestra de (1+(2**i*))x(1+(2**i*)).

El següent exemple ens permet veure els resultats de les dues operacions esmentades:

```
#include <stdio.h>
#include <conio.h>
#include <mil.h>

#define FITXER_ENT "cell.mim"
#define AMPLADA 512
#define ALCADA 480

void main (void) {
        /* Variables MIL */
         MIL_ID
            MilApplication,      // Identificador de l'aplicació
            MilSystem,           // Identificador del sistema
            MilDisplay,          // Display de la imatge original
            MilDisplayEro,       // Display de la imatge Erosionada
            MilDisplayDil,       // Display de la imatge Dilatada
           MilImage,             // Imatge original
           MilImageEro,          // Imatge Erosionada
           MilImageDil;          // Imatge Dilatada

        /* Inicialització per treballar amb les MIL */
         MappAllocDefault (M_SETUP, &MilApplication, &MilSystem,
                           M_NULL,M_NULL, M_NULL);

        /* Inicialitzar display */
         MdispAlloc (MilSystem, M_DEV0, M_DISPLAY_SETUP, M_DEFAULT,
                     &MilDisplay);

         MdispAlloc (MilSystem, M_DEV1, M_DISPLAY_SETUP, M_DEFAULT,
                     &MilDisplayEro);
         MdispAlloc (MilSystem, M_DEV2, M_DISPLAY_SETUP, M_DEFAULT,
                     &MilDisplayDil);

        /* Inicialitzar buffer */
         MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                      M_IMAGE+M_PROC+M_DISP, &MilImage);
         MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                      M_IMAGE+M_PROC+M_DISP, &MilImageEro);
         MbufAlloc2d (MilSystem, AMPLADA, ALCADA, 8+M_UNSIGNED,
                      M_IMAGE+M_PROC+M_DISP, &MilImageDil);

        /* Carreguem la imatge */
```

```
            MbufLoad (FITXER_ENT, MilImage);

           /* Associació del display amb la imatge, per visualitzar-la */
            MdispSelect (MilDisplay, MilImage);

            printf ("\n Prem <RETURN> per veure el resultat de l'erosió.");
            getch ();

           /* Realitzem l'erosió sobre la imatge original */
            MimErode (MilImage, MilImageEro, 1, M_GRAYSCALE);

           /* Associació del display amb la imatge, per visualitzar-la */
            MdispSelect (MilDisplayEro, MilImageEro);

            printf ("\n Prem <RETURN> per veure el resultat de la
                     dilatació.");
            getch ();

           /* Realitzem la dilatació sobre la imatge erosionada */
            MimDilate (MilImageEro, MilImageDil, 1, M_GRAYSCALE);

           /* Associació del display amb la imatge, per visualitzar-la */
            MdispSelect (MilDisplayDil, MilImageDil);

            printf ("\n Prem <RETURN> per acabar.");
            getch ();

           /* Alliberem els recursos adquirits */
            MbufFree (MilImage);
            MbufFree (MilImageEro);
            MbufFree (MilImageDil);
            MdispFree (MilDisplay);
            MdispFree (MilDisplayEro);
            MdispFree (MilDisplayDil);
            MappFreeDefault (MilApplication, MilSystem, M_NULL, M_NULL,
                             M_NULL);
}
```